MTConnect Standard

Part 3 – Streams, Events, and Samples

Version 1.0.1

Prepared for: MTConnect Institute
Prepared by: William Sobel
Prepared on: October 2, 2009

MTConnect Specification

Table of Contents

Table of Figures

1 Overview

MTConnect is a standard based on an open protocol for data integration. MTConnect is not intended to replace the functionality of existing products, but it strives to enhance the data acquisition capabilities of devices and applications and move toward a plug-and-play environment to reduce the cost of integration.

MTConnect is built upon the most prevalent standards in the manufacturing and software industry, maximizing the number of tools available for its implementation and providing the highest level of interoperability with other standards and tools in these industries.

To facilitate this level of interoperability, a number of objectives are being met. Foremost is the ability to transfer data via a standard protocol which includes:

- A device identity (i.e. model number, serial number, calibration data, etc.).
- The identity of all the independent components of the device.
- Possibly a device's design characteristics (i.e. axis length, maximum speeds, device thresholds, etc.).
- Most importantly, data captured in real or near-real-time (i.e. current speed, position data, temperature data, program block, etc.) by a device that can be utilized by other devices or applications (e.g. utilized by maintenance diagnostic systems, management production information systems, CAM products, etc.).

The types of data that may need to be addressed in MTConnect could include:

- Physical and actual device design data
- Measurement or calibration data
- Near-real-time data from the device

To accommodate the vast amount of different types of devices and information that may come into play, MTConnect will provide a common high-level vocabulary and structure.

The first version of MTConnect will focus on a limited set of the characteristics mentioned above that were selected based on the fact that they can have an immediate affect on the efficiency of operations.

1.1 MTConnect Document Structure

The MTConnect specification is subdivided using the following scheme:

Part 1: Overview and Protocol
Part 2: Components and Data Items
Part 3: Streams, Events and Samples

Extensions to the standard will be made according to this scheme and new sections will be added as new areas are addressed. Documents will be named as follows: MTC_Part_<Number>_<Description>.doc. All documents will be developed in Microsoft® Word format and released in Adobe® PDF format. For example, this document is MTC_Part_1_Overview.doc.

2 Purpose of This Document

This document is intended to:

- define the MTConnect standard;
- specify the requirements for compliance with the MTConnect standard;
- provide engineers with sufficient information to implement *Agents* for their devices;
- provide developers with the necessary guidelines to use the standard to develop applications.

The third part of the standard covers the data returned from a current or sample request (for more information on the requests, see Part 1). Part 2 covered what data is available; this section covers the values of the data representing the state of the machine. The values and the descriptive information are separated do reduce the amount of redundant information and reduce the network bandwidth used by the protocol.

The information is broken down into two general types. The first is events that represent information that has finite state changes like controller modes and samples that are continuously changing like axis positions. This section also covers the vocabulary and format of every piece of data that can be retrieved from the machine.

2.1 Terminology

Adapter An optional software component that connects the Agent to the Device.

Agent A process that implements the MTConnect specification, acting as an interface to the device.

Alarm An alarm indicates an event that requires attention and indicates a deviation from normal operation.

Application A process or set of processes that access the MTConnect *Agent* to perform some task.

Attribute A part of an element that provides additional information about that element. For example, the `name` element of the `Device` is given as `<Device` **`name=”mill-1”`**`>...</Device>`

CDATA The text in a simple content element. For example, *`This is some text`*, in `<mt:Alarm ...>This is some text</mt:Alarm>`.

Component A part of a device that can have sub-components and data items. A component is a basic building block of a device.

Controlled Vocabulary The value of an element or attribute is limited to a restricted set of possibilities. Examples of controlled vocabularies are country codes: US, JP, CA, FR, DE, etc…

Current A snapshot request to the *Agent* to retrieve the current values of all the data items specified in the path parameter. If no path parameter is given, then the values for all components are provided.

Data Item A data item provides the descriptive information regarding something that can be collected by the *Agent*.

Device A piece of equipment capable of performing an operation. A device is composed of a set of components that provide data to the application. The device is a separate entity with at least one Controller managing its operation.

Discovery Discovery is a service that allows the application to locate *Agents* for devices in the manufacturing environment. The discovery service is also referred to as the *Name Service*.

Element An XML element is the central building block of any XML Document. For example, in MTConnect the Device element is specified as <**`Device`** `>...</`**`Device`**>

Event An event represents a change in state that occurs at a point in time. Note: An event does not occur at predefined frequencies.

HTTP Hyper-Text Transport Protocol. The protocol used by all web browsers and web applications.

Instance When used in software engineering, the word *instance* is used to define a single physical example of that type. In object-oriented models, there is the class that describes the thing and the instance that is an example of that thing.

LDAP Lightweight Directory Access Protocol, better known as Active Directory in Microsoft Windows. This protocol provides resource location and contact information in a hierarchal structure.

MIME Multipurpose Internet Mail Extensions. A format used for encoding multipart mail and http content with separate sections separated by a fixed boundary.

Probe A request to determine the configuration and reporting capabilities of the device.

REST REpresentational State Transfer. A software architecture where the client and server move through a series of state transitions based solely on the request from the client and the response from the server.

Results A general term for the `Samples` and `Events` contained in a `ComponentStream` as a response from a `sample` or `current` request.

Sample A sample is a data point from within a continuous series of data points. An example of a Sample is the position of an axis.

Socket When used concerning interprocess communication, it refers to a connection between two end-points (usually processes). Socket communication most often uses TCP/IP as the underlying protocol.

Stream A collection of events and samples organized by devices and components.

Service An application that provides necessary functionality.

Tag Used to reference an instance of an XML element.

TCP/IP TCP/IP is the most prevalent stream-based protocol for interprocess communication. It is based on the IP stack (Internet Protocol) and provides the flow-control and reliable transmission layer on top of the IP routing infrastructure.

URI Universal Resource Identifier. This is the official name for a web address as seen in the address bar of a browser.

UUID Universally unique identifier.

XPath XPath is a language for addressing parts of an XML Document. See the XPath specification for more information. http://www.w3.org/TR/xpath

XML Extensible Markup Language. http://www.w3.org/XML/

XML Schema The definition of the XML structure and vocabularies used in the XML Document.

XML Document An instance of an XML Schema which has a single root element and conforms to the XML specification and schema.

2.2 XML Terminology

In the document there will be references to XML constructs, including elements, attributes, CDATA, and more. XML consists of a hierarchy of elements. The elements can contain sub-elements, CDATA, or both. For this specification, however, an element never contains mixed content or both sub-elements and CDATA. Attributes are additional information associated with an *element*. The textual representation of an element is referred to as a *tag*. In the example:

```
<Foo name="bob">Ack!</Foo>
```

an *element* consists of a named opening and closing tag. In the above example, `<Foo...>` is referred to as the opening tag and `</Foo>` is referred to as the closing tag. The text `Ack!` in between the opening and closing tags is called the `CDATA`. `CDATA` can be restricted to certain formats, patterns, or words. In the document when it refers to an element having CDATA, it indicates that the element has no sub-elements and only contains data.

When one looks at an XML Document there are two parts. The first part is typically referred to as an XML declaration and is only a single line. It looks something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
```

This line indicates the XML version being used and the character encoding. Though it is possible to leave this line off, it is usually considered good form to include this line in the beginning of the document. The second part contains the XML document and consists of the rest of the document.

Every XML Document contains one and only one root element. In the case of MTConnect, it is the MTConnectDevices, MTConnectStreams, or MTConnectError element. When these root elements are used in the examples, you will sometimes notice that it is prefixed with mt: as in mt:MTConnectDevices. The mt: is what is referred to as a namespace. In XML, to allow for multiple XML Schemas to be used within the same XML Document, a namespace will indicate which XML Schema is in effect for this section of the document. This convention allows for multiple XML Schemas to be used within the same XML Document, even if they have the same element names. The namespace is optional and is only required if multiple schemas are required.

An *attribute* is additional data that can be included in each XML element. For example, in the following MTConnect DataItem, there are several attributes describing the data item:

```
<DataItem name="Xpos" type="POSITION" subType="ACTUAL" category="SAMPLE" />
```

The name, type, subType, and category are attributes of the element. Each attribute can only occur once within an element declaration, and it can either be required or optional.

An element can have any number of sub-elements. The XML Schema specifies which sub-elements and how many times a given sub-element can occur. Here's an example:

```
<TopLevel>
  <FirstLevel>
    <SecondLevel>
      <ThirdLevel name="first"></ThirdLevel>
      <ThirdLevel name="second"></ThirdLevel>
    </SecondLevel>
  </FirstLevel>
</TopLevel>
```

In the above example, the FirstLevel has a sub-element SecondLevel which in turn has two sub-elements, ThirdLevel, with different names. Each level is an element and its children are its sub-elements and so forth.

An XML Document can be validated. The most basic check is to make sure it is well-formed, meaning that each element has a closing tag, as in <foo>...</foo> and the document does not contain any illegal characters (<>) when not specifying a tag. If the closing </foo> was left off or an extra > was in the document, the document would not be well-formed and may be rejected by the receiver. The document can also be validated against a schema to ensure it is valid. This second level of analysis checks to make sure that required elements and attributes are present and only occur the correct number of times. A valid document must be well-formed.

All MTConnect documents must be valid and conform to the XML Schema provided along with this specification. The schema will be versioned along with this specification. The greatest possible care will be taken to make sure that the schema is backward compatible.

For more information, visit the w3c website for the XML Standards documentation: http://www.w3.org/XML/

2.3 Markup Conventions

MTConnect follows industry conventions on tag format and notations when developing the XML schema. The general guidelines are as follows:

1. All tag names will be specified in Pascal case (first letter of each word is capitalized). For example: <ComponentEvents />
2. Attribute names will also be camel case, similar to Pascal case, but the first letter will be lower case. For example: <MyElement attributeName=”bob”/>
3. All values that are part of a limited or controlled vocabulary will be in upper case. For example: `ON`, `OFF`, `ACTUAL`, etc…
4. Dates and times will follow the W3C ISO 8601 format with arbitrary fractions of a second allowed. Refer to the following specification for details: http://www.w3.org/TR/NOTE-datetime The format will be YYYY-MM-DDThh:mm:ss.ffff, for example 2007-09-13T13:01.213415. The accuracy and number of fractional digits of the timestamp is determined by the capabilities of the device collecting the data. All times will be given in UTC (GMT).
5. Element names will be spelled-out and abbreviations will be avoided. The one exception is the word `identifier` that will be abbreviated `Id`. For example: `SequenceNumber` will be used instead of `SeqNum`.

2.4 Document Conventions

The following documentation conventions will be used in the text:

- The word **MUST** is used to indicate provisions that are mandatory. Any deviation from those provisions will not be permitted.
- The word **SHOULD** is used to indicate a provision that is recommended but the exclusion of which will not invalidate the implementation.
- The word **MAY** will be used to indicate provisions that are optional and are up to the implementor to decide if they are relevant to their device.

In the tables where elements are described, the Occurrence column indicates if the attribute or sub-elements are required by the specification.

For attributes:

1. If the Occurrence is 1, the attribute **MUST** be provided.
2. If the Occurrence is 0..1, the attribute **MAY** be provided, and at most one occurrence of the attribute may be given.

For elements:

1. If the Occurrence is 1, the element **MUST** be provided.
2. If the Occurrence is 0..1, the element **MAY** be provided, and at most one occurrence of the element may be given.
3. If the Occurrence is 1..INF, one or more elements **MUST** be provided.

4. If the Occurrence is a number, e.g. 2, exactly that number of elements **MUST** be provided.

Font styles used:

Code samples as well as any XML elements or attributes will always be given in `fixed width fonts`. References to other *Documents* or *Sections* will be presented in italics.

2.5 Units

MTConnect will adopt the units common to most standards specifications for exchanging data items. This will allow for greatest interoperability with other specifications. It is assumed that all MTConnect *Agents* will be responsible for converting the units from the native device units.

Property	Symbol	Unit
Angle	°	decimal degrees
Angular Acceleration	$°/s^2$	degree per second square
Angular Velocity	°/s	degrees per second
Elapsed time	s	seconds with fractions
Force	N	newtons
Length	mm	millimeters
Linear Acceleration	mm/s^2	millimeter per second square
Linear Velocity	mm/s	millimeters per second
Mass	kg	kilograms
Spindle Speed	rev/min	revolutions per minute
Temperature	°C	degree Celsius

Additional units will be added as needed. The decision to require the *Agent* to convert to the standard simplifies the applications and will provide greater interoperability and accuracy.

2.6 Referenced Standards and Specifications

A large number of specifications are being used to normalize and harmonize the schema and the vocabulary (names of tags and attributes) specified in MTConnect *(See Bibliography for complete references).*

3 Streams, Samples and Events

The MTConnect *Agent* collects data from various sources and delivers it to applications in response to `sample` or `current` requests. (See *Protocol* section.) All the data are collected into streams and organized by device and then by componentA component stream has two parts: `Samples` and `Events`. `Samples` are point-in-time readings from a component reporting what the value is at that instant. For an example, refer to the Device in Figure 2 below.

An `Event` changes state to a limited set of values. It is assumed that an event remains at a state until the next event occurs; it cannot have any intermediate values between the reported values. Alarms are classified as events. The following are examples of `Events`: `Block`, `Code`, `Execution`, `PowerStatus`, etc.

If two adjacent samples for the same component and data item have the same value, the second sample **MUST NOT** be sent to the client application and does not need to be retained by the MTConnect *Agent*. This will greatly reduce the amount of information sent to the application. The application can always assume that if the sample is not present, it has the previous value. If the application needs the present value, it can always ask for the `current` values (see *Protocol).*

3.1 Streams

A `Streams` element is the high level container for all device streams. It serves no other purpose than to have `DeviceStream` sub-elements. There **MUST** be no attributes or elements within this element.

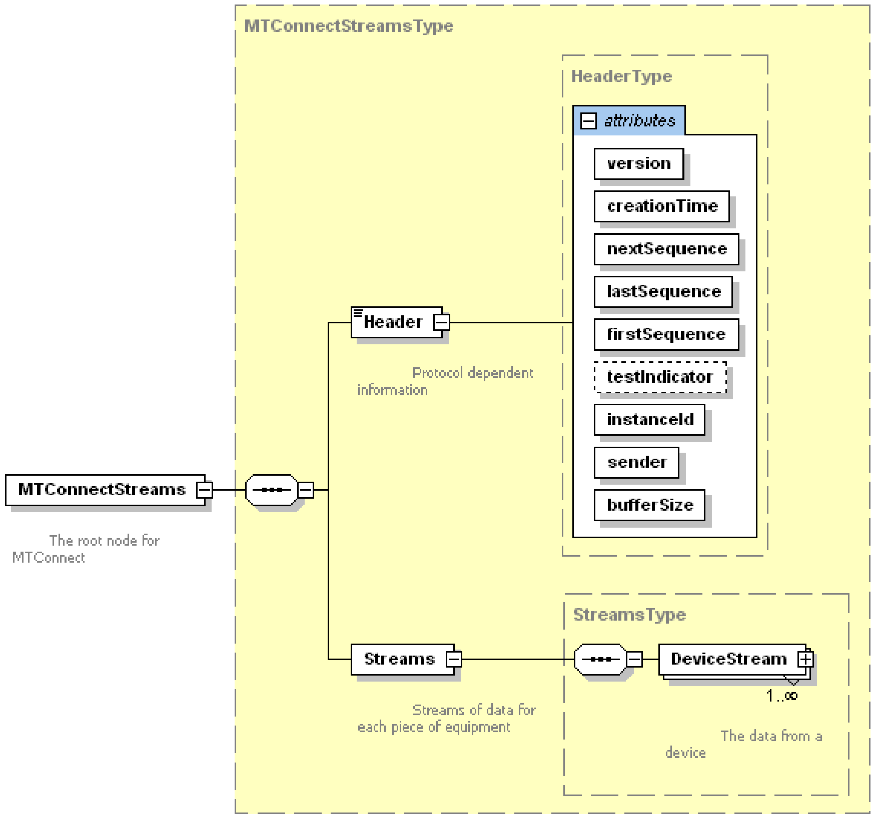

Figure 1: Streams Schema Diagram

Elements	Description	Occurrence
`DeviceStream`	The stream of samples and events for each device.	1..INF

3.2 Structure

The following diagram illustrates the structure of the streams with some samples and events at the lowest level:

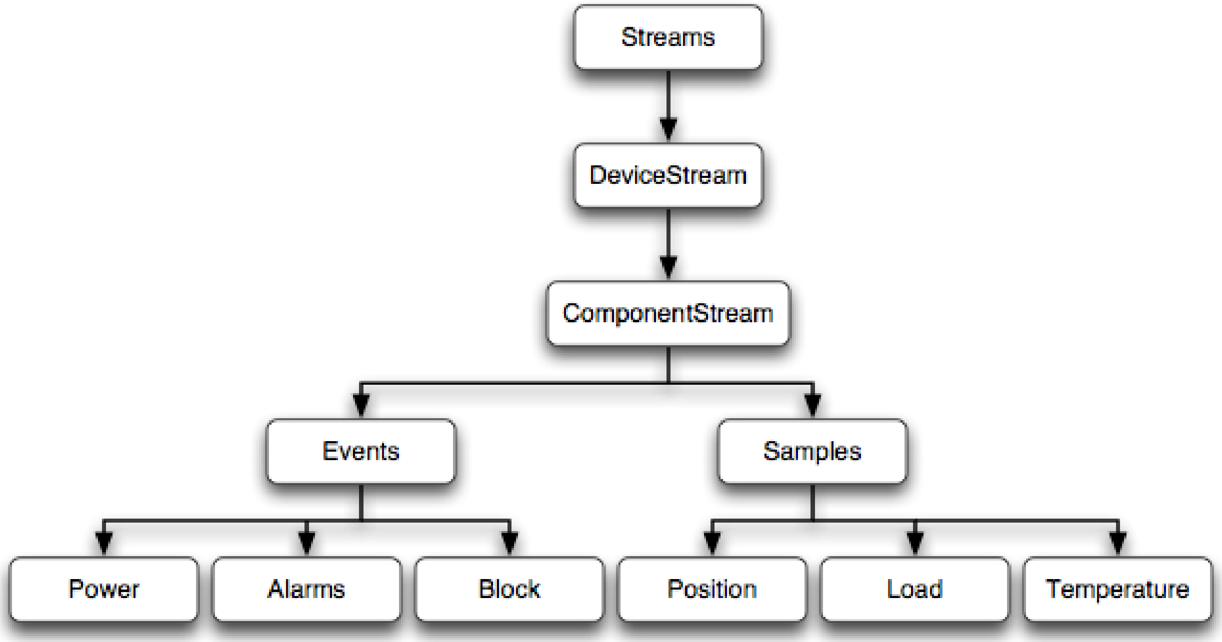

Figure 2: Streams Example Structure

A `Stream` **MUST** have at least one `DeviceStream` and the `DeviceStream` **MAY** have one or more `ComponentStream` elements, depending on whether there are events or samples available for the component. If there are no `ComponentStream` elements, then no data will be delivered for this request.

Below is an example XML Document response for an *Agent* with two devices, mill-1 and mill-2. The data is reported in two separate device streams. The sequence numbers is unique across the two devices. The applications **MUST NOT** assume that the event and sample sequence numbers are strictly in sequence. The sequence numbers **MAY** skip due to filtering.

```
<?xml version="1.0" encoding="UTF-8"?>
<MTConnectStreams ...>
  <Header .../>
  <Streams>
    <DeviceStream uuid="1" name="mill-1">
      <ComponentStream componentId="2" name="power" component="Power">
        <Events>
          <PowerStatus name="power" dataItemId="9" sequence="30055111"
     timestamp="2008-07-07T14:27:59.591">ON</PowerStatus>
        </Events>
        </ComponentStream>
      </DeviceStream>
      <DeviceStream uuid="2" name="mill-2">
        <ComponentStream componentId="3" name="power" component="Power">
          <Events>
```

```
              <PowerStatus name="power" dataItemId="10" sequence="52162"
    timestamp="2008-06-11T10:17:33.291">ON</PowerStatus>
            </Events>
          </ComponentStream>
        </DeviceStream>
      </Streams>
    </MTConnectStreams>
```

3.3 DeviceStream

A `DeviceStream` is created to hold the device-specific information so it does not need to be repeated for every event and sample. This is done to reduce the size of each event and sample so they only carry the information that is being reported. A `DeviceStream` **MAY** contain one or more `ComponentStream` elements. If the request is valid and there are no events or samples that match the criteria, an empty `DeviceStream` element **MUST** be created to indicate that the device exists, but there was no data available.

3.3.1 DeviceStream Attributes

Attributes	Description	Occurrence
name	The device's name	1
uuid	The device's unique identifier	1

3.3.2 DeviceStream Elements

Element	Description	Occurrence
ComponentStream	One component's stream for each component with data	0..INF

3.4 ComponentStream

A `ComponentStream` is similar to the `DeviceStream`. It contains the information specific to the component within the `Device`. The `uuid` only needs to be specified if the Component has a `uuid` assigned.

3.4.1 ComponentStream Attributes

Attribute	Description	Occurrence
name	This components name within the device	1
component	The element name for the component	1
uuid	The component's unique identifier	0..1
componentId	Corresponds to the id attribute of the component in the probe request (Refer Probe in Part 1).	1

The Elements of the `ComponentStream` classify the data into `Events` and `Samples`. *(The classification is discussed below)*. The `ComponentStream` **MUST NOT** be empty. It **MUST** include an `Events` and/or a `Samples` element.

3.4.2 `ComponentStream` Elements

Element	Description	Occurrence
`Events`	The events for this component stream	0..1
`Samples`	The samples for this component	0..1

3.5 `Samples`

The `Samples` element must contain at least one `Sample` element. This element acts only as a container for all the `Samples` to provide a logical structure to the XML Document.

Element	Description	Occurrence
`Sample`	The subtype of `Sample` for this component stream	1..INF

3.6 `Sample`

A `Sample` is an abstract type. This means there will never be an actual element called `Sample`, but any element that is a sub-type of `Sample` can be used in place of `Sample`. Examples of sample sub-types are `Position`, `Load`, and `Angle`. Sample types **MUST** have numeric values.

3.6.1 Sample attributes:

Attribute	Description	Occurrence
`name`	The name **MUST** match the name of the `DataItem` this sample is associated with.	1
`sequence`	The sequence number of this sample. This value **MUST** have a maximum value of 2^63-1 and **MUST** be stored in a signed 64 bit integer.	1
`timestamp`	The timestamp of the sample.	1
`dataItemID`	The id attribute of the corresponding data retrieved in the probe request.	1

A sample **MUST** contain `CDATA` as the content between the element tags. A position is formatted like this:

```
1. <Position sequence="112" timestamp="2007-08-09T12:32:45.1232" name="Xabs"
      dataItemId="10">123.3333</Position>
```

In this example the `123.3333` is the `CDATA` for the position. All the `CDATA` in a sample is typed, meaning that it can be validated using an XML parser. This restricts the format of the values to a specific pattern.

3.6.2 Sample Elements

`Acceleration` The acceleration of the component MUST always be reported in `MILLIMETER/SECOND^2`. An acceleration **MUST** have a numeric value.

`Amperage` The current in an electrical circuit. The amperage **MUST** have a numeric value and **MUST** be reported in `AMPS`.

`Angle` An angle **MUST** always be reported in `DEGREE` and **MUST** always have a numeric CDATA value as a floating point number.

`AngularAcceleration` The angular acceleration of the component as measured in `DEGREE/SECOND^2`. An acceleration **MUST** have a numeric value.

`AngularVelocity` A angular velocity represents the rate of change in angle. An angular velocity **MUST** always be reported in `DEGREE/SECOND` and **MUST** always have a numeric CDATA value as a floating point number.

`AxisFeedrate` Axis Feedrate is defined as the rate of motion of the feed axis of the tool relative to the workpiece[1]. An axis feedrate **MUST** always be reported in `MILLIMETER/SECOND` or `PERCENT` for override and **MUST** always have a numeric CDATA value as a floating point number.

`PathFeedrate` Path Feedrate is defined as the rate of motion of the feed path of the tool relative to the workpiece[2]. A path feedrate **MUST** always be reported in `MILLIMETER/SECOND` or `PERCENT` for override and **MUST** always have a numeric CDATA value as a floating point number.

`Frequency` The rate at which a component is vibrating. The frequency **MUST** have a numeric value and **MUST** be reported in `HERTZ`.

`Displacement` The displacement as measured from zero to peak. The displacement **MUST** have a value reported in `MILLIMETER`.

`GlobalPosition` The global position is the three space coordinate of the tool. A global position **MUST** always be reported in `MILLIMETER` and **MUST** always have a numeric CDATA value as three floating point numbers (x, y, and z). Position **MUST** always be given in absolute coordinates.

[1] From ASME B5.54 - 2005
[2] From ASME B5.54 - 2005

`Load` The load on a component. The load **MUST** always be reported in `NEWTON` and **MUST** always have a numeric CDATA value as a floating point number.

`Position` A position represents the location along a linear axis. A position **MUST** always be reported in `MILLIMETER` and **MUST** always have a numeric CDATA value as a floating point number. Position **MUST** always be given in absolute coordinates.

`Pressure` The pressure on a component. The pressure **MUST** be a numeric value and **MUST** be provided in `PASCALS`.

`SpindleSpeed` The rate of rotation of a machine spindle [3]. A spindle speed **MUST** always be reported in `REVOLUTION/MINUTE` and **MUST** always have a numeric CDATA value as a floating point number.

`Temperature` Temperature **MUST** always be reported in degrees `CELSIUS` and **MUST** always have a numeric CDATA value as a floating point number.

`Torque` The torque of the component **MUST** be reported in SI units of `NEWTON_METER` and **MUST** have a numeric CDATA value as a floating point number.

`Velocity` A velocity represents the rate of change in position along one or more axis. When given as a Sample for the `Axes` component, it represents the magnitude of the velocity vector for all given axis, similar to a path feedrate. A velocity **MUST** always be reported in `MILLIMETER/SECOND` and **MUST** always have a numeric CDATA value as a floating point number.

`Volts` The potential difference as measured across an electrical circuit. The voltage **MUST** have a numeric value and **MUST** be reported in `VOLTS`.

`Watts` The electrical power (volt-amps) of an electrical circuit. The watts **MUST** have a numeric value and **MUST** be reported in `WATTS`.

3.6.3 Extensibility

Additional sample types can be added by extending the `Sample` type in the XML schema. The samples presented here are the official sample types that will be supported by all MTConnect *Agents*. Any non-sanctioned extensions will not be guaranteed to have consistency across implementations.

3.7 Events

The `Events` element must contain at least one `Event` element. This element acts only as a container for all the `Events` to provide a logical structure to the XML Document.

Element	Description	Occurrence

[3] From ASME B5.54 - 2005

Element	Description	Occurrence
Event	The subtype of Event for this component stream	1..INF

3.8 Event

A Event is an abstract type. This means there will never be an actual element called Event, but any element that is a sub-type of Event can be used in place of Sample. Examples of event sub-types are Block, Execution, and Line. Events types have values in any format.

Attribute	Description	Occurrence
name	The name **MUST** match the name of the DataItem this event is associated with	1
sequence	The sequence number of this event. This value **MUST** have a maximum value of 2^63-1 and **MUST** be stored in a signed 64 bit integer.	1
timestamp	The time-stamp of the event	1
dataItemID	The id attribute of the corresponding data retrieved in the probe request.	1

An event is similar to a sample, but its values are going to be changing with unpredictable frequency. Events do not have intermediate values. When a power status transitions from OFF to ON, there is no intermediate state that can be inferred. Therefore, most events have a controlled vocabulary as their content.

An event does not add any additional attributes or elements to the Sample. It is a placeholder in the schema type hierarchy for elements that are events. This relationship will be enforced by the schema.

3.8.1 Event Elements

Block A Block of code is a command being executed by the Controller. The Block **MUST** include the entire command with all the parameters.

Code The code is just the G, M, or NC code being executed. The Code **MUST** only contain the simplest form of the executing command.

ControllerMode The Mode of the Controller. The CDATA **MUST** be one of the following:

Value	Description
AUTOMATIC	The controller is configured to automatically execute a program.
MANUAL	The controller is under manual control by the operator.

Value	Description
MANUAL_DATA_INPUT	The operator can enter operations for the controller to perform. There is no current program being executed.

Direction A Direction indicates the direction of rotation. The CDATA **MUST** be either CLOCKWISE or COUNTER_CLOCKWISE.

Execution The Execution state of the Controller. The CDATA **MUST** be one of the following:

Value	Description
READY	The controller is ready to execute. It is currently idle.
ACTIVE	The controller is actively executing an instruction.
INTERRUPTED	The operator or the program has paused execution and is waiting to be continued.
STOPPED	The controller has been stopped.

Line The current line number of the program being executed. The CDATA **MUST** be a numeric value.

PartCount The number of parts produced. This will not be counted by the agent and **MUST** only be supplied if the controller provides the count.

PowerStatus Power status **MUST** be either ON or OFF.

Program The name of the program executing in the controller. This is usually the name of the file containing the program instructions.

3.9 Alarms

The Alarm event adds some additional fields to the standard Event schema. The following additional attributes are used for the alarm:

Attribute	Description	Occurrence
code	The type of alarm. This is a high level classification for all codes.	1
severity	The severity of the alarm, currently we have CRITICAL, ERROR, WARNING, or INFORMATION.	1
nativeCode	The native code for the piece of equipment. This is the way the alarm is represented on the component.	1

Attribute	Description	Occurrence
`state`	Either `INSTANT`, `ACTIVE` or `CLEARED`. When the Alarm occurs, it will be created with an `ACTIVE` state. Once it has been addressed, the state will be changed to `CLEARED`. An `INSTANT` alarm does not need to be cleared.	1
`lang`	An optional attribute that specifies language of the alarm text. Refer to IETF RFC 4646 (http://www.ietf.org/rfc/rfc4646.txt) or successor for a full definition of the values for this attribute.	0..1

The `code` can have one of the following values:

Enumeration	Description
`CRASH`	A spindle crashed
`JAM`	A component jammed.
`FAILURE`	The component failed.
`FAULT`	A fault occurred on the component.
`STALLED`	The component has stalled and cannot move.
`OVERLOAD`	The component is overloaded.
`ESTOP`	The ESTOP button was pressed.
`MATERIAL`	There is a problem with the material.
`MESSAGE`	A system message.
`OTHER`	The alarm is not in any of the above categories.

The CDATA of the Alarm is the human-readable text from the component that raised the alarm. The device should specify this text so it can be logged.

4 Annotated XML Examples

4.1 Example of a `current` Request

The sample was generated with the following request:

```
http://10.1.23.5/LinuxCNC/sample?path=//Controller|//Power
```

The response is as follows:

```
1. <MTConnectStreams xmlns:m="urn:mtconnect.com:MTConnectStreams:0.9"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:mtconnect.com:MTConnectStreams:0.9"
xsi:schemaLocation="urn:mtconnect.com:MTConnectStreams:0.9
/schemas/MTConnectStreams.xsd">
2.    <Header sender="10.1.23.5" bufferSize="100000" creationTime="2008-07-
      07T23:22:40-07:00" nextSequence="31088439" version="0.9"
      instanceId="1214527986"/>
3.      <Streams>
```

Events are grouped by equipment:

```
4.        <DeviceStream uuid="linux-01" name="LinuxCNC">
```

All the events are then grouped by components. The path includes the most relevant parts of the xpath with only the `Components` containers removed here for brevity. The only element that **MUST** be removed is `Components`. The `name` selector makes the component unique within the path:

```
5.    <ComponentStream componentId="2" name="power" component="Power">
6.            <Events>
7.              <PowerStatus name="power" dataItemId="9" sequence="30055111"
      timestamp="2008-07-10T10:27:59.591">ON</PowerStatus>
8.           </Events>
9.         </ComponentStream>
```

The control execution is now idle:

```
10.         <ComponentStream componentId="8" name="Controller"
      component="Controller">
11.           <Events>
12.             <Execution name="execution" dataItemId="22"
      sequence="38148653" timestamp="2008-07-10T12:34:00.615">IDLE</Execution>
```

The execution unit is now running:

```
            <Execution name="execution" dataItemId="22" sequence="38148753" timestamp="2008-07-10T12:35:00.615">EXECUTING </Execution>
          </Events>
        </ComponentStream>
      </DeviceStream>
    </Streams>
</MTConnectStreams>
```

5 Bibliography

1. Engineering Industries Association. *EIA Standard - EIA-274-D*, Interchangeable Variable, Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically Controlled Machines. Washington, D.C. 1979.

2. ISO TC 184/SC4/WG3 N1089. *ISO/DIS 10303-238*: Industrial automation systems and integration Product data representation and exchange Part 238: Application Protocols: Application interpreted model for computerized numerical controllers. Geneva, Switzerland, 2004.

3. International Organization for Standardization. *ISO 14649*: Industrial automation systems and integration – Physical device control – Data model for computerized numerical controllers – Part 10: General process data. Geneva, Switzerland, 2004.

4. International Organization for Standardization. *ISO 14649*: Industrial automation systems and integration – Physical device control – Data model for computerized numerical controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.

5. International Organization for Standardization. *ISO 6983/1* – Numerical Control of machines – Program format and definition of address words – Part 1: Data format for positioning, line and contouring control systems. Geneva, Switzerland, 1982.

6. Electronic Industries Association. *ANSI/EIA-494-B-1992*, 32 Bit Binary CL (BCL) and 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines. Washington, D.C. 1992.

7. National Aerospace Standard. *Uniform Cutting Tests* - NAS Series: Metal Cutting Equipment Specifications. Washington, D.C. 1969.

8. International Organization for Standardization. *ISO 10303-11*: 1994, Industrial automation systems and integration Product data representation and exchange Part 11: Description methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.

9. International Organization for Standardization. *ISO 10303-21*: 1996, Industrial automation systems and integration -- Product data representation and exchange -- Part 21: Implementation methods: Clear text encoding of the exchange structure. Geneva, Switzerland, 1996.

10. H.L. Horton, F.D. Jones, and E. Oberg. *Machinery's handbook*. Industrial Press, Inc. New York, 1984.

11. International Organization for Standardization. *ISO 841-2001: Industrial automation systems and integration - Numerical control of machines - Coordinate systems and motion nomenclature.* Geneva, Switzerland, 2001.

12. *ASME B5.59-2 Version 9c: Data Specification for Properties of Machine Tools for Milling and Turning. 2005.*

13. *ASME/ANSI B5.54: Methods for Performance Evaluation of Computer Numerically Controlled Lathes and Turning Centers. 2005.*

14. OPC Foundation. *OPC Unified Architecture Specification, Part 1: Concepts Version 1.00. July 28, 2006.*