MTConnect® Standard

Part 2 – Device Information Model

Version 1.3.0

Prepared for: MTConnect Institute
Prepared by: John Turner
Prepared on: September 30, 2014

MTConnect® Specification and Materials

If you intend to adopt or implement an MTConnect® Specification or Material in a product, whether hardware, software or firmware, which complies with an MTConnect® Specification, you shall agree to the MTConnect® Specification Implementer License Agreement ("Implementer License") or to the MTConnect® Intellectual Property Policy and Agreement ("IP Policy"). The Implementer License and IP Policy each sets forth the license terms and other terms of use for MTConnect® Implementers to adopt or implement the MTConnect® Specifications, including certain license rights covering necessary patent claims for that purpose. These materials can be found at www.MTConnect.org, or by contacting Helina Heliu at mailto:hheliu@amt.com.

MTConnect® Institute and AMT have no responsibility to identify patents, patent claims or patent applications which may relate to or be required to implement a Specification, or to determine the legal validity or scope of any such patent claims brought to their attention. Each MTConnect® Implementer is responsible for securing its own licenses or rights to any patent or other intellectual property rights that may be necessary for such use, and neither AMT nor MTConnect® Institute have any obligation to secure any such rights.

This Material and all MTConnect® Specifications and Materials are provided "as is" and MTConnect® Institute and AMT, and each of their respective members, officers, affiliates, sponsors and agents, make no representation or warranty of any kind relating to these materials or to any implementation of the MTConnect® Specifications or Materials in any product, including, without limitation, any expressed or implied warranty of non-infringement, merchantability, or fitness for particular purpose, or of the accuracy, reliability, or completeness of information contained herein. In no event shall MTConnect® Institute or AMT be liable to any user or implementer of MTConnect® Specifications or Materials for the cost of procuring substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, indirect, special or punitive damages or other direct damages, whether under contract, tort, warranty or otherwise, arising in any way out of access, use or inability to use the MTConnect® Specification or other MTConnect® Materials, whether or not they had advance notice of the possibility of such damage.

Table of Contents

Table of Figures

1 Purpose of This Document

This document, *Part 2 Device Information Model* of the MTConnect® Standard, defines the rules and terminology to be used by designers to describe the function and operation of a device and to define the data that is provided by an MTConnect Agent from a device. The Device Information Model also defines the structure for the XML document that is returned from an MTConnect Agent in response to a `Probe` request.

In the MTConnect Standard, a device typically represents a single piece of equipment (i.e. machine, robot, etc.). It can also represent any logical grouping of pieces of equipment that operate together to perform a function.

Note: See *Part 3 Streams* of the MTConnect Standard for details on the XML documents constructed using the Streams Information Model which are returned from an MTConnect Agent in response to a `Sample` or `Current` request.

2 Terminology

Refer to *Section 2* of *Part 1 Overview and Protocol* for a dictionary of terms used in the MTConnect Standard.

3 Device Information Model

The Device Information Model is an XML data model that is comprised of two primary types of XML Elements –Structural Elements and Data Elements.

In the MTConnect Standard, Structural Elements are defined as XML Elements that describe the physical and logical parts and sub-parts of a device (*Section 4* of this document).

Likewise, Data Elements are defined as XML Elements that describe data that can be collected from a device (*Section 5* of this document).

Together, the Structural Elements and Data Elements form the information that is provided in a MTConnect Device XML document that allows a client software application to interpret the data in that document and to correlate that data back into the same meaning, value, and context that it had at the original source device.

Note: The MTConnect Standard also defines the information model for `Assets`. An Asset is something that is associated with the manufacturing process that is not a component of a device, can be removed without detriment to the function of the device, and can be associated with other devices during their lifecycle. See *Part 4 Assets* of the MTConnect Standard, for more details on `Assets`.

4 Structural Elements for a Device

There are several types of Structural Elements defined to describe a device – each is an XML Element and together they provide the structure used to organize information about a device. Some of these Structural Elements **MUST** always appear in the XML document for a device, while others are optional and **MAY** be used, as required, to provide additional context or definition to a device.

The first, or highest level, Structural Element in the Device Information Model is `Devices.` `Devices` is a container type XML element. `Devices` provides the structure for organizing data from one or multiple devices into a single XML document and **MUST** always appear in an XML document for a device.

`Device` is the next Structural Element in the Device Information Model. `Device` is also a container type XML element. `Device` is used to organize information representing a single piece of equipment or it can represent any logical grouping of pieces of equipment that operate together to perform a unique function. One or more `Device` element(s) **MUST** always appear in the XML document describing a device(s).

`Components` is the next Structural Element in the Device Information Model. `Components` is also a container type XML element. `Components` is used to organize information representing each of the physical or logical parts of a device.

The `Components` container is comprised of one or more `Component` type XML Elements. The `Components` element **MAY** or **MAY NOT** appear in the XML document describing a device.

`Component` is the next level of Structural Element in the Device Information Model. `Component` is an abstract type XML element. As such, the `Component` XML element will never appear in the XML document describing a device - only the different `Component` Types defined in Section 5 will appear in the XML document.

Each `Component` is a container type XML element used to organize lower level Structural Elements or Data Elements associated with the `Component.` If lower level Structural Elements are described, these elements are by definition child `Component` elements of a parent `Component.` At this next level, the child `Component` elements are grouped into an XML container called `Components.`

This lower level `Components` container is comprised of one or more child `Component` XML elements representing the sub-parts of the parent `Component.` Just like the parent `Component` element, the child Component element is an abstract type XML element and will never appear in the XML document – only the different child `Component` types will appear.

This parent-child relationship can continue to any depth required to fully define a device. For clarity, the MTConnect Standard calls these lower level child `Component` elements *`Subcomponent`* elements.

The following example is an XML document structure that demonstrates the relationship between a parent `Component` and the child *`Subcomponent`*:

```
<Devices>
  <Device>
    <Components>
      <Axes  (Component)>
        <Components>
          <Linear  (Subcomponent)>
            < Components)>
              <Etc. (Subcomponent)>
```

The following XML Tree demonstrates the various Structural Elements for a device and the relationship between these elements.

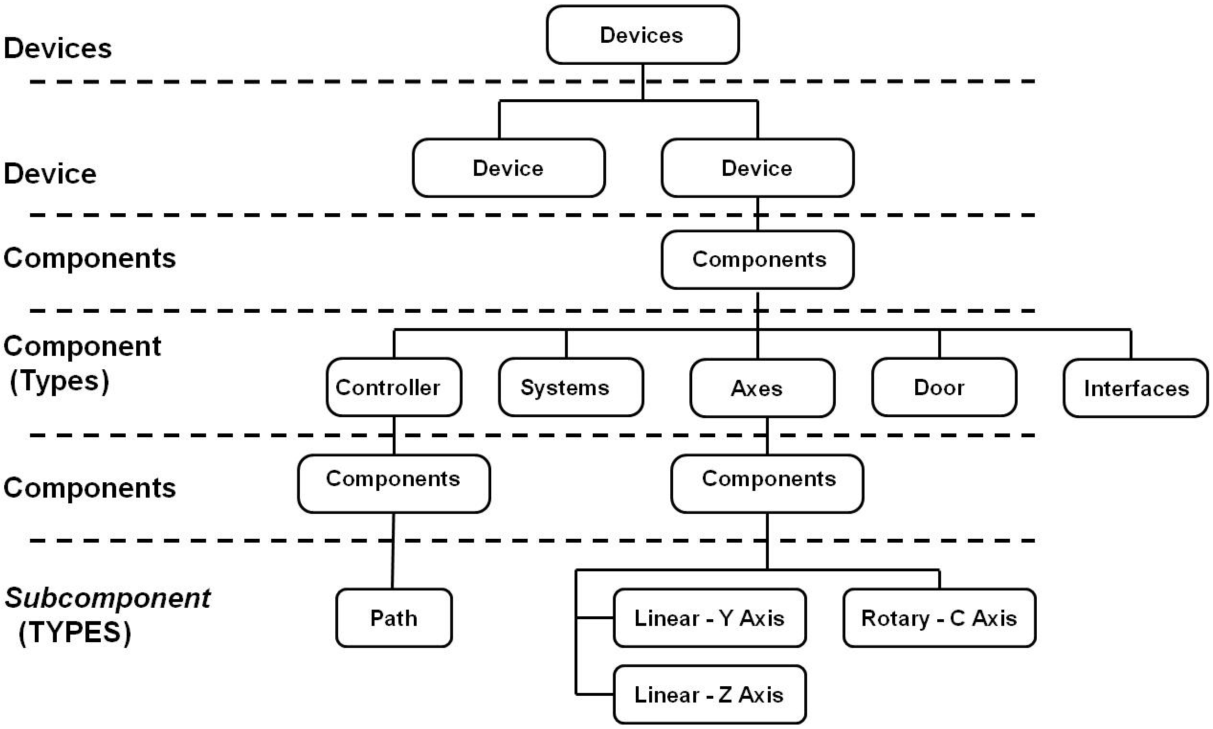

Figure 1: Example Device Structural Elements

4.1 Devices

The Devices XML Element is the top level container in the XML document provided for any device. Devices **MUST** contain only Device elements. Devices **MUST** contain at least one Device element, but **MAY** contain multiple Device elements. Data Elements **MAY NOT** be directly associated with the Devices container.

Elements	Description	Occurrence
Devices	The root XML element for the XML document provided for a device.	1

4.2 Device

Device is an XML container type element that holds all the Structural XML elements and Data XML elements associated with a device. Data Elements **MAY** be directly associated with the Device container. Device **MUST** have the EVENT category data item AVAILABILITY that indicates if this device is available to provide information.

In the Device Information Model, Device is a unique type of Structural XML element. Device carries all of the properties of a Component (see Section 4.3). Additionally, Device **MUST** have a unique identifier attribute (uuid) that identifies the device and it **SHOULD** not be changed over time. It **MUST** also only appear once in any XML document. All Structural XML elements and Data XML elements associated with a device are therefore uniquely identified through their association with the Device container.

Elements	Description	Occurrence
Device	The primary container element of each device. Device is contained within the top level Devices container. There **MAY** be multiple Device elements in an XML document.	1..INF

Note: Some pieces of equipment may not be integral to a parent device. These pieces of equipment may function independently or produce data that is not relevant to a parent device. An example would be a temperature sensor installed in a plant to monitor the ambient air temperature. In such a case, these individual pieces of equipment, if they singularly or together perform a unique function, **MAY** be modeled in an MTConnect XML document as a Device. When modeled as a Device, these pieces of equipment **MUST** provide all of the data and capabilities defined for a Device.

It is also possible for a piece of equipment to be defined as a `Component` of a parent device and simultaneously as an independent `Device`; communicating data associated with the parent `Device` incorporated into that device's data set and independently communicating additional data in a separate data set using its own identity (`uuid`). An example would be a vibration monitoring system that itself is defined as a `Device` reporting its own information and some of the data from this system is also reported in the data set for the piece of equipment that is being monitored.

4.2.1 XML Schema Structure for a Device

The following XML tree represents the structure of the `Device` XML Element showing the attributes defined for `Device` and the sub-elements that may be associated with the `Device`.

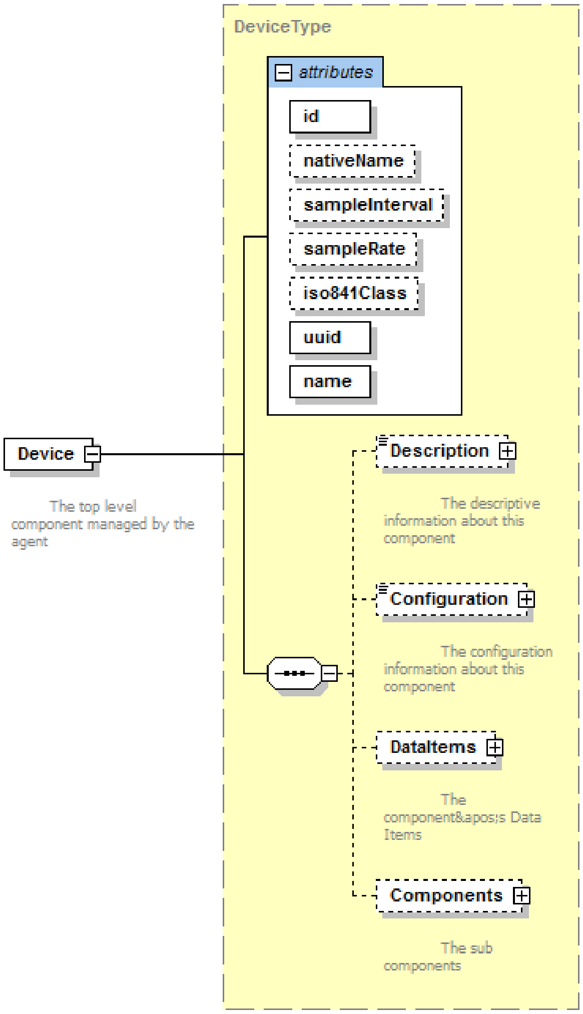
DeviceType
attributes
id
nativeName
sampleInterval
sampleRate
iso841Class
uuid
name
Device
The top level component managed by the agent
Description
The descriptive information about this component
Configuration
The configuration information about this component
DataItems
The component&apos;s Data Items
Components
The sub components

Figure 2: `Device` Schema Diagram

4.2.2 Attributes for Device

The following table defines the attributes that may be used to provide additional information for a Device type element.

Attribute	Description	Occurrence
iso841Class	DEPRECATED in Release 1.1.0	
uuid	A unique identifier that will only refer to this Device. For example, this may be the manufacturer's code and the serial number. The uuid should be alphanumeric and not exceeding 255 characters. An NMTOKEN XML type.	1*
name	The name of the Device. This name should be unique within the XML document to allow for easier data integration. An NMTOKEN XML type.	1
nativeName	The name the device manufacturer assigned to this Device. If the native name is not provided, it **MUST** be the name.	0..1
id	The unique identifier for this Device in the document. An id **MUST** be unique across all the id attributes in the document. An XML ID-type.	1
sampleRate	DEPRECATED IN REL. 1.2 (REPLACED BY sampleInterval)	
sampleInterval	The interval in milliseconds between the completion of the reading of one sample of data from a device until the beginning of the next sampling of that data. This is the number of milliseconds between data captures. If the sample interval is smaller than one millisecond, the number can be represented as a floating point number. For example, an interval of 100 microseconds would be 0.1.	0..1**

Notes: * The uuid **MUST** be provided for the Device. It is optional for other Structural XML elements – Component and *Subcomponent*.

** The sampleInterval is used to aid a client software application in interpolating values provided by some Data Elements. This is the desired sample interval and may vary depending on the capabilities of the device.

4.2.3 Sub-Elements for Device

The following table lists the sub-elements defined to provide additional information for a Device. These sub-elements are organized in the Device container.

Element	Description	Occurrence
Description	An XML element that can contain any descriptive content. This can contain configuration information and manufacturer specific details.	0..1
Configuration	An XML element that can contain descriptive content defining the configuration information for a Device.	0..1
Components	A container for Component XML Elements associated with this Device.	0..INF
DataItems	A container for the Data XML Elements (See Details in *Section 5* of this document) provided by this Device. The data items define the measured values to be reported by this Device.	1..INF*

Notes: * DataItems **MUST** be provided since every device **MUST** report AVAILABILITY.

4.3 Components

Components is an XML container that provides structure for the physical and logical sub-elements of a device. Components contains one or more Component XML Elements.

Elements	Description	Occurrence
Components	XML Container consisting of one or more types of Component XML Elements. Only one Components container MAY appear for a Device element.	0..1

4.4 Component

A `Component` XML Element defines the structure of the physical or logical parts of a device and the association of the data supplied from that device to the specific part of the device to which it applies. `Component` is an abstract type XML element and will never appear directly in the MTConnect XML document. As an abstract type XML element, `Component` will be replaced in the XML document by specific component types. XML elements representing `Component` are described in Section 5 and include elements such as `Axes`, `Controller`, `Door`, etc.

Elements	Description	Occurrence
`Component`	An abstract XML Element. Replaced in the XML document by types of `Component` elements representing physical and logical parts of the `Device`. There can be multiple types of `Component` XML Elements in the document.	1..INF

4.4.1 XML Schema Structure for `Component`

The following XML tree represents the structure of a `Component` XML element showing the attributes defined for `Component` and the sub-elements that may be associated with `Component` type XML elements.

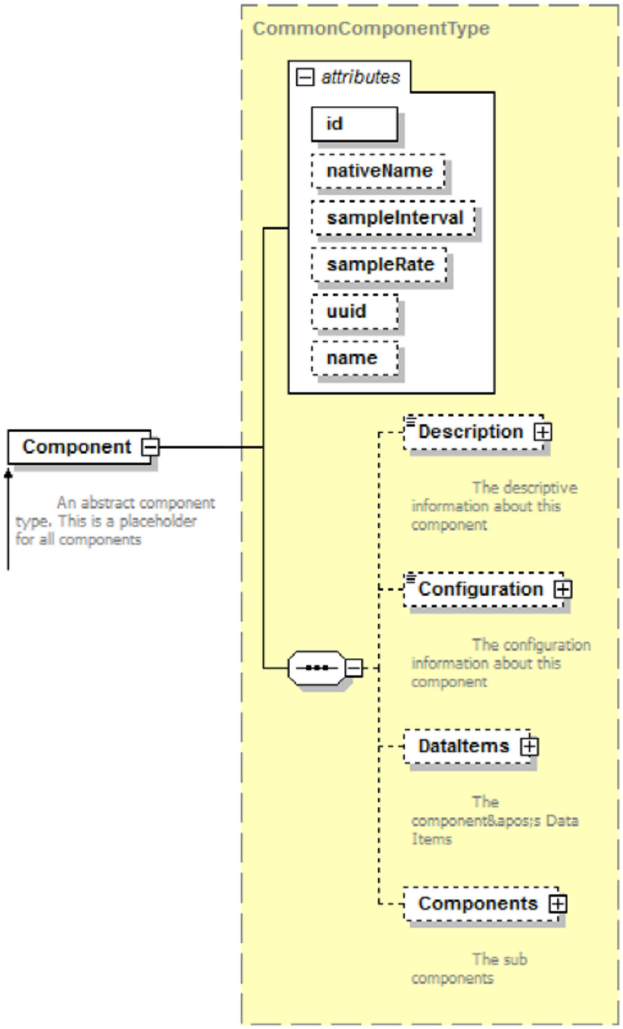
CommonComponentType
attributes
id
nativeName
sampleInterval
sampleRate
uuid
name
Component
An abstract component type. This is a placeholder for all components
Description
The descriptive information about this component
Configuration
The configuration information about this component
DataItems
The component&apos;s Data Items
Components
The sub components

Figure 3: `Component` Schema

4.4.2 Attributes for Component

The following table defines the attributes that may be used to provide additional information for a Component type XML element.

Attribute	Description	Occurrence
uuid	A unique identifier that will only refer to this Component. For example, this can be the manufacturer's code or the serial number. The uuid should be alphanumeric and not exceeding 255 characters. An NMTOKEN XML type.	0..1*
name	The name of the Component. name is an optional attribute. If provided, name **MUST** be unique within a type of Component or subComponent. It is recommended that duplicate names **SHOULD NOT** occur within a Device. An NMTOKEN XML type.	0..1
nativeName	The name the device manufacturer assigned to the Component. If the native name is not provided it **MUST** be the name.	0..1
id	The unique identifier for this Component in the document. An id **MUST** be unique across all the id attributes in the document. An XML ID-type.	1
sampleRate	DEPRECATED IN REL. 1.2 (REPLACED BY sampleInterval)	
sampleInterval	The interval in milliseconds between the completion of the reading of one sample of data from a component until the beginning of the next sampling of that data. This is the number of milliseconds between data captures. If the sample interval is smaller than one millisecond, the number can be represented as a floating point number. For example, an interval of 100 microseconds would be 0.1.	0..1**

Notes: * While the uuid **MUST** be provided for the Device element, it is optional for Component and *Subcomponent* elements.

** The sampleInterval is used to aid a client software application in interpolating values provided by some Data Elements. This is the desired sample interval and may vary depending on the capabilities of the device.

4.4.3 Sub-Elements of Component

The following table lists the sub-elements defined to provide additional information for a Component type XML Element.

Element	Description	Occurrence
Description	An element that can contain any descriptive content. This can contain information about the Component and manufacturer specific details.	0..1
Configuration	An element that can contain descriptive content defining the configuration information for a Component.	0..1
Components	A container for lower level Component XML Elements associated with this parent Component. These lower level elements in this container are defined as *Subcomponent* elements.	0..INF*
DataItems	A container for the Data XML Elements (defined below) provided that are directly related to this Component. The data items define the measured values to be reported that are related to this Component.	0..INF*

Notes: *At least one of Components or DataItems **MUST** be provided.

4.4.3.1 Description for Component

The following XML tree represents the structure of the Description XML sub-element showing the attributes defined for Description.

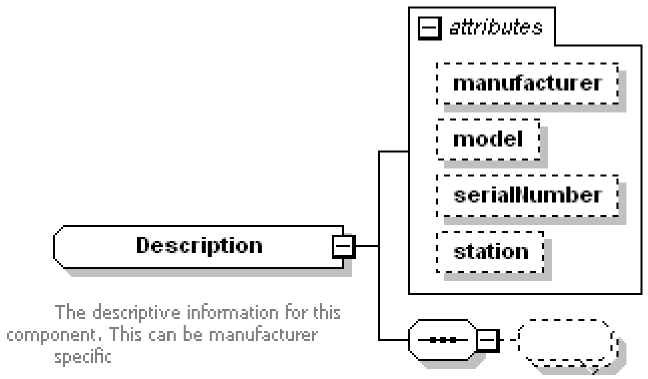
attributes
manufacturer
model
serialNumber
station
Description
The descriptive information for this component. This can be manufacturer specific

Figure 4: Component Description Schema

The following table lists the attributes defined for the Description XML sub-element.

Attribute	Description	Occurrence
manufacturer	The name of the manufacturer of the Component	0..1
model	The model description of the Component	0..1
serialNumber	The component's serial number	0..1
station	The station where the Component is located when a component is part of a manufacturing unit or cell with multiple stations that share the same physical controller.	0..1

The CDATA of Description is any additional descriptive information the implementer chooses to include regarding the Component. An example of a Description is as follows:

```
<Description manufacturer="Example Co" serialNumber="A124FFF"
  station="2"> Example Co Simulated Vertical 3 Axis Machining center.>
</Description>
```

The information can be provided for any component. For example, an electrical power sensor can be defined as follows:

```
<Description manufacturer="Example Co"
   serialNumber="EXCO-TT-099PP-XXXX"> Advanced Pulse watt-hour transducer
   with pulse output>
</Description>
```

4.4.3.2 Configuration for Component

The Configuration XML element contains descriptive information about a Component. Configuration **MAY** include any manufacturer's information, calibration data, maintenance information, or any other information or data relative to the Component.

Not all Component types support Configuration. When Configuration is supported, details on the schema for Configuration will be included in the applicable sections of the MTConnect standard.

Element	Description	Occurrence
Configuration	An XML element that can contain descriptive content defining the configuration information for a Component.	0..1

Configuration data for a Component is structured in the Device Information Model as shown below. AbstractConfiguration is an abstract type XML element. It will never appear in the XML document for a device. When Configuration is supported for a Component type, that configuration will appear in the XML document. Currently, Sensor is the only component type that supports Configuration.

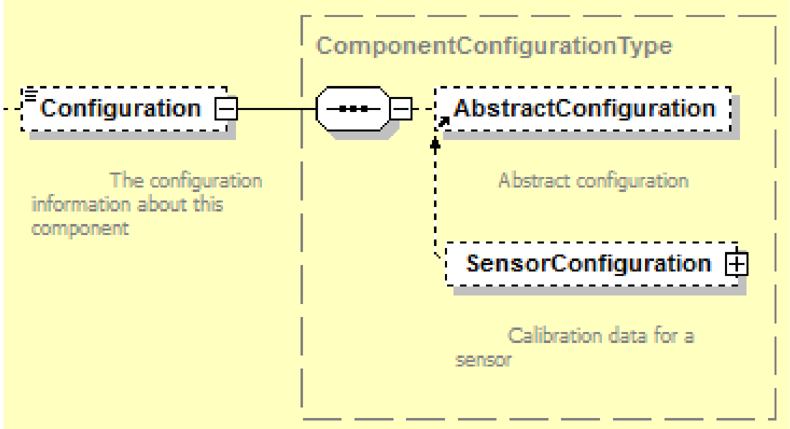
ComponentConfigurationType
Configuration
The configuration information about this component
AbstractConfiguration
Abstract configuration
SensorConfiguration
Calibration data for a sensor

Figure 5: Component Configuration Schema

4.4.3.3 Components for Component

Components is an XML container used to organize information representing the physical and logical sub-parts of a parent Component.

Components provides the ability to add lower level sub-parts to a higher level Component. These lower level elements can add more clarity and granularity to the physical or logical structure of a device and the data being retrieved from the device.

A Component may also have sub-types. For example Axes has the sub-types Linear and Rotary. These sub-types are also defined as a Component within the Components container.

These lower level sub-parts of a Component are called *Subcomponent* elements within the MTConnect Standard to more clearly define the relationship between the parent Component and its associated child sub-elements (*Subcomponent* elements). *Subcomponent* elements use the same XML structure as Component. See *Section 4.4.1* of this document for details on the structure for Component.

`Components` contains one or more of the child *Subcomponent* type XML Elements.

Element	Description	Occurrence
`Components`	An XML container comprised of one or more `Component` type XML elements (*`Subcomponent`* elements).	0..1

The `Components`-`Component`-`Components`-*`Subcomponent`*-`Components` structure can be expanded as required to provide the level of detail required to describe the sub-parts of a device and to provide the level of granularity and context required for the data provided from the device.

A parent `Component` and the child sub-elements (*`Subcomponent`*) are represented in a XML document as follows:

```
<Devices>
  <Device>
    <Components>
      <Axes(Component)>
        <Components>
          <Linear (Subcomponent)>
            < Components>
              <Etc.(Subcomponent)>
```

4.4.3.4 DataItems for Component

`DataItems` is an XML container that provides structure for the Data Elements collected from a device that are associated with each `Component` in the XML document describing a device.

See Section 6.1 of this document for details on the `DataItems` XML Element.

Element	Description	Occurrence
`DataItems`	XML Container consisting of one or more Data Elements. Only one `DataItems` container **MAY** appear for a `Component` element.	0..1

5 Component and *Subcomponent* Type Structural Elements

Component and *Subcomponent* Structural Elements define physical or logical parts (and sub-parts) of a device that provide additional granularity and more precise definition for the structure of the device. They also provide the association of the data supplied from that device to the specific part of the device to which it applies.

As described in Section 4 above, Component and *Subcomponent* are both abstract type Structural Elements within the Device Data Model and will never appear directly in the MTConnect XML document. As abstract type XML elements, Component and *Subcomponent* will be replaced in the XML document by specific Component and *Subcomponent* types defined below.

The following table defines the top-level Component types available to describe a device.

Top Level Components	Description
Axes	Structural Elements that perform linear or rotational motion associated with a Device.
Controller	The intelligent or computational part of a Device which monitors and calculates information
Systems	Structural Elements describing the major sub-systems that provide services to a Device
Door	Mechanisms or closures that can cover access portals into a Device.
Sensor	Signal processing unit of a measurement sub-system within a Device.
Stock	The material to which work is applied in a machine or piece of equipment to produce parts.
Interfaces	The information used to coordinate actions and activity between devices or sub-systems and a device.

Note: As the MTConnect Standard evolves, more Component types and associated *Subcomponent* types will be added to support new devices and/or new parts of devices.

5.1 Axes

Axes provides the information for Structural Elements that perform linear or rotational motion for the Device.

Axes is an XML container that organizes Structural Elements representing individual axes into *Subcomponent* types of Linear and Rotary based on the type of motion performed by each axis. Axes **MUST** contain at least one Linear or one Rotary axis.

A Linear axis represents the movement of a physical device, or a portion of a device, in a straight line. Movement may be in either a positive or negative direction. Linear type axes **MUST** be named X, Y, Z; with numbers appended for additional axes in the same plane. Additional linear axes are often referred to as U, V, and W. However, MTConnect defines the secondary axes to X, Y, and Z as X2, Y2, and Z2.

A Rotary axis represents any non-linear or rotary movement of a physical device, or a portion of a device. Rotary type axes **MUST** be named A, B, and C and rotate around the X, Y, and Z axes respectively. As with the Linear axes, a number **MUST** be appended for additional axes in the same plane (C, C2, C3, C4, …).

An axis whose function is to provide rotary motion may function as a continuous rotation (SPINDLE mode), continuous-path contour rotary motion (CONTOUR mode), or positioning (INDEX mode) to discrete rotary positions. As such, a rotary axis **MUST** specify a subType attribute of SPINDLE, INDEX, or CONTOUR.

The following diagram defines the relationship between the Axes container and the individual Axis type Structural Elements.

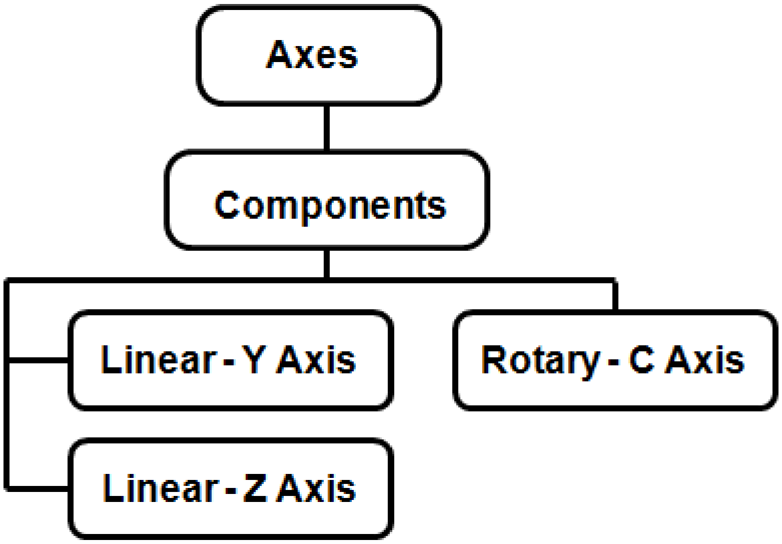

Figure 6: Axes Example With Two Linear Axes and One Rotary Axis

5.1.1 Chuck

Chuck represents a mechanism that holds a part or stock material in place. It may also represent a mechanism that holds any other item in place within a device. The operation of a Chuck is represented by Chuck_State. The value of Chuck_State **MAY** be OPEN, CLOSED, or UNLATCHED.

5.2 Controller

Controller represents an intelligent part of a Device which monitors and calculates information that alters the operating conditions of the Device and the other Component and *Subcomponent* elements of the Device. Typical types of controllers for a piece of equipment are CNC (Computer Numerical Control), PAC (Programmable Automation Control), IPC (Industrialized Computer), or an IC (Imbedded Computer).

Controller provides information regarding the execution of a control program(s), the mode of operation of the device, and fault information regarding the operation of the device.

Note: MTConnect *Version 1.1.0* and later implementations **SHOULD** use a *Subcomponent* called Path to represent an individual tool path and Execution state (see Path). When the machine is capable of executing more than one simultaneous program, the implementation **MUST** specify each Path type *Subcomponent*.

5.2.1 Path

Path represents the information for an independent operation or function within a Controller. Typically, Path represents a set of Axes, one or more Program elements, and the data associated with the motion of a control point as it moves through space. However, it **MAY** represent any independent function within a Controller that has unique data associated with that function.

If the controller is capable of performing more than one independent operation or function simultaneously, a Path component **MUST** be used to organize the data associated with each independent operation or function.

5.3 Power (DEPRECATED in Rel. 1.1)

NOTE: Power as an indication of a device's ability to provide data was changed to an Event category DataItem called AVAILABILITY in Release 1.1. Also, electrical current and power consumption **MUST** be represented by the Electric system, see *Section 5.5.5* of this document for more information.

5.4 Door

Door represents a mechanical mechanism or closure that can cover an access portal into a piece of equipment. The closure can be opened or closed to allow or restrict access to other parts of the equipment. Door **MUST** have a DataItem called DOOR_STATE to indicate if the closure is OPEN, CLOSED, or UNLATCHED. A device may contain multiple door type components.

5.5 Systems

Systems is an XML container that provides structure for the information describing functional sub-systems of a Device.

Many pieces of equipment have functional sub-systems that perform as discrete operating modules of the equipment or provide services to support the operation of the equipment. These sub-systems are comprised of many parts that are not easily deconstructed into lower level parts. Since these sub-systems operate as a functional unit, they are represented in the Device Information Model as a unit and identified by the function or service provided to the equipment.

Systems contains one or more *Subcomponent* type XML Element(s) representing each of the sub-systems of the Device.

5.5.1 Hydraulic

Hydraulic represents a system comprised of all the parts involved in moving and distributing pressurized liquid for the purpose of delivering a source of power to specific types of actuators.

5.5.2 Pneumatic

Pneumatic represents a system comprised of all the parts involved in moving and distributing pressurized gas regardless of purpose or activity.

5.5.3 Coolant

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

5.5.4 Lubrication

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

5.5.5 Electric

Electric represents the main power supply or generator for the device. The electric system will provide all the data with regard to current, voltage, and frequency that apply to the Device as a functional unit. Data regarding electric power that is specific to a Component or *Subcomponent* will be reported as a DataItem for that specific Component or *Subcomponent*.

5.6 Actuator

`Actuator` describes a device for moving or controlling a mechanism or system. It takes energy, usually transported by air, electric current, or liquid and converts it into some kind of motion.

`Actuator` is a unique Structural Element since it may function, and be modeled, as either a primary `Component` of a `Device` or it may be a *`Subcomponent`* of a parent `Component`.

5.7 Sensor

`Sensor` is a XML Element that represents a measurement device. `Sensor` is a unique Structural Element since it may function, and be modeled, as either a primary `Component` of a `Device` or it may be a *`Subcomponent`* of a parent `Component`.

5.8 Stock

`Stock` is a Structural Element that represents the material that is used in a manufacturing process and to which work is applied in a machine or piece of equipment to produce parts.

`Stock` may be either a continuous piece of material from which multiple parts may be produced or it may be a discrete piece of material that will produce a part or a set of parts.

5.9 Interfaces

`Interfaces` is a `Component` type Structural Element in the Device Information Model. `Interfaces` is used to organize the information provided by a device that supports integration with other pieces of equipment that are associated with that `Device`. As such, `Interfaces` represents the inter-device communication information used to coordinate the operation between a `Device` and other associated pieces of equipment.

`Interfaces` is also a container type XML element. As a container, it organizes the information used to coordinate the operation between the `Device` and each one of the associated pieces of equipment into separate sets of information. Each set of information is defined as an `Interface`.

`Interface` is an abstract type Structural Element within the Device Data Model and will never appear directly in the MTConnect XML document. As an abstract type XML element, `Interface` will be replaced in the XML document by specific `Interface` types defined below.

Each `Interface` type contains two types of Data Elements - `DataItem` elements that are unique for that type of `Interface` and represent the state of the `Interface` (detailed in *Section 7.2.1* of this document) and any other `DataItem` elements available from the device that may be needed to coordinate the operation with the associated piece of equipment.

In addition to DataItem elements, an Interface may have an additional XML element type called References.   An Interface may require data and state information from other Component and *Subcomponent* Structural Elements which has already been defined elsewhere in the XML document.   To avoid duplication of this data and state information, References provides a method to include the data from other Structural Elements to also be included in the set of information provided for an Interface.   See *Section 6.2.5* of this document for more information on References.

An Interface is represented in a XML document as follows:

```
<Devices>
 <Device>
  <Components>
   <Interfaces(Component)>
    <Components>
     <Interface Type(Subcomponent)>
      < Components>
        <Etc.(Subcomponent)>
```

5.9.1 Interface Types

The data exchanged between a Device and various types of associated equipment will differ based on the functions to be performed by each piece of equipment.   The information required by a specific type of equipment will be defined by an Interface type XML element.

An initial list of Interface types are defined below.

Note: Additional Interface types will be defined in future releases of the MTConnect Standard.

5.9.1.1 BarFeederInterface

The set of information used to coordinate the operations between a device and a Bar Feeder. Bar Feeder is a piece of equipment that pushes bar stock (long cylindrical pieces of material) into machine piece of equipment – most typically a lathe or turning center. As each part is machined, a cutting tool creates a final cut to separate the part from the bar stock and the feeder then feeds the bar for the next part to be produced, allowing for continual operation of the machine.   The bar feeder controls the length of material and the type of material fed, if there is the ability to load more than one type of material, into the machine for each part to be produced.

5.9.1.2 MaterialHandlerInterface

The set of information used to coordinate the operations between a device and an associated piece of equipment used to automatically handle various types of materials or services associated with the device. A material handler is a piece of equipment capable of providing any one, or more, of a variety of support services for a machine (Device). These services can include loading and/or unloading material, loading/unloading tooling, inspection/testing, cleaning, etc.

A robot is a common example of a material handler.

5.9.1.3 DoorInterface

The set of information used to coordinate the operations between two devices, one of which controls the operation of a door which provides access to a piece of equipment. This interface will reference a specific Door component and **MUST** report the Door_State of the door.

5.9.1.4 ChuckInterface

The set of information used to coordinate the operations between two devices, one of which controls the operation of a chuck. This interface will be reference a specific Chuck component and **MUST** report the Chuck_State of the chuck.

6 Data Elements for a Device

In the Device Information Model, Data Elements are XML Elements that describe data that can be collected from a device and are associated with `Device,Component`, or *`Subcomponent`* Structural Elements.

There are two types of Data Elements defined to organize the data collected from a device. These are `DataItems` and `DataItem`.

Each Data Element should be modeled in the XML document such that it is aligned directly with the Structural Element that the specific data is most closely associated.

The first, or highest level, Data Element defined in the Device Information Model is `DatatItems. DataItems` is a container type XML element. `DataItems` provides the structure for organizing data from a device and associates that data to the Structural Element that it applies.

The `DataItems` container is comprised of one or more `DataItem` type XML Elements. The `DataItems` element **MAY** or **MAY NOT** appear for each Structural Element in the XML document describing a device; depending on whether data is being collected for that specific Structural Element.

`DataItem` is the other Data Element defined in the Device Information Model. `DataItem` represents a piece of information that **MAY** represent either a numeric value or a health status for a device or a *Subcomponent* of a device. `DataItem` provides a detailed description for each piece of data that is collected from a device; the type of data being collected, an array of optional attributes that further defines that data, and the value of the data.

`DataItem` is an abstract type XML element. As such, the `DataItem` XML element will never appear in the XML Document. Only the different DataItem Types defined in Section 7 will appear in the XML document describing a device.

The following XML Tree demonstrates the relationship between Data Elements (`DataItem`) and the various Structural Elements in the Device Information Model.

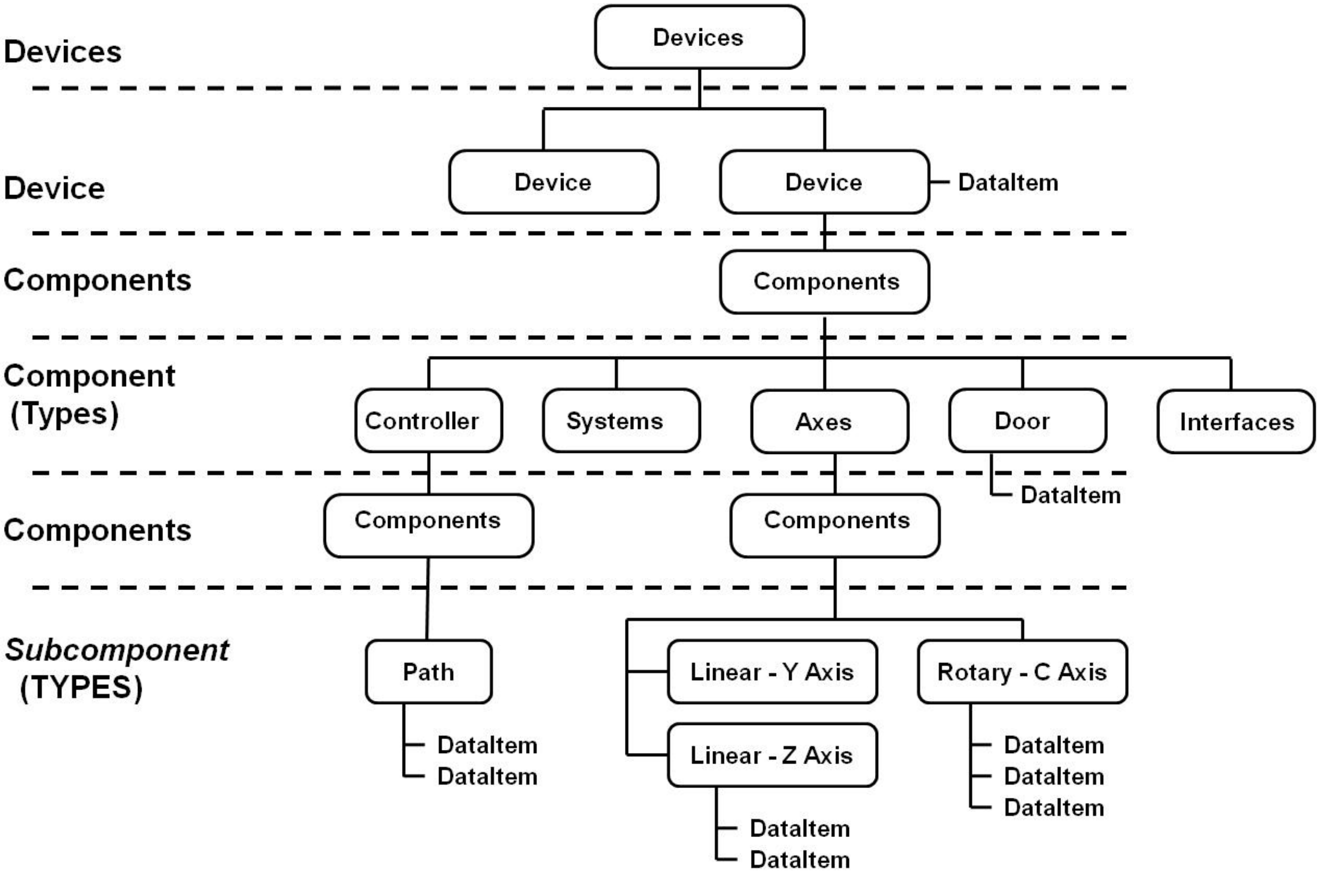

Figure 7: Example Device Data Elements (`DataItem`)

6.1 DataItems

The `DataItems` XML Element is the top level container for the Data Elements associated with a `Device`, `Component`, or *`Subcomponent`*. `DataItems` MUST contain only `DataItem` type elements. `DataItems` **MUST** contain at least one `DataItem` type element, but **MAY** contain multiple `DataItem` type elements.

Elements	Description	Occurrence
`DataItems`	XML Container consisting of one or more types of `DataItem` XML Elements. Only one `DataItems` container **MUST** appear for each Structural Element in the XML document.	0..1

6.2 DataItem

A DataItem XML Element represents each piece of data that **MAY** be collected by an MTConnect Agent from a device. DataItem provides a detailed description for each piece of data that is collected from a device - the type of data being collected, an array of optional attributes that further defines that data, and the value of the data.

DataItem is an abstract type XML element and will never appear directly in the MTConnect XML document. As an abstract type XML element, DataItem will be replaced in the XML document by specific data item types. XML elements representing DataItem will include elements such as Temperature, Pressure, Velocity, etc.

Elements	Description	Occurrence
DataItem	An abstract XML Element. Replaced in the XML document by Elements representing various types of DataItem XML Elements. There can be multiple types of DataItem XML Elements in the document.	1..INF

6.2.1 XML Schema Structure for `DataItem`

The following XML tree represents the structure of a `DataItem` XML element showing the attributes defined for `DataItem` and the sub-elements that may be associated with `DataItem` type XML elements.

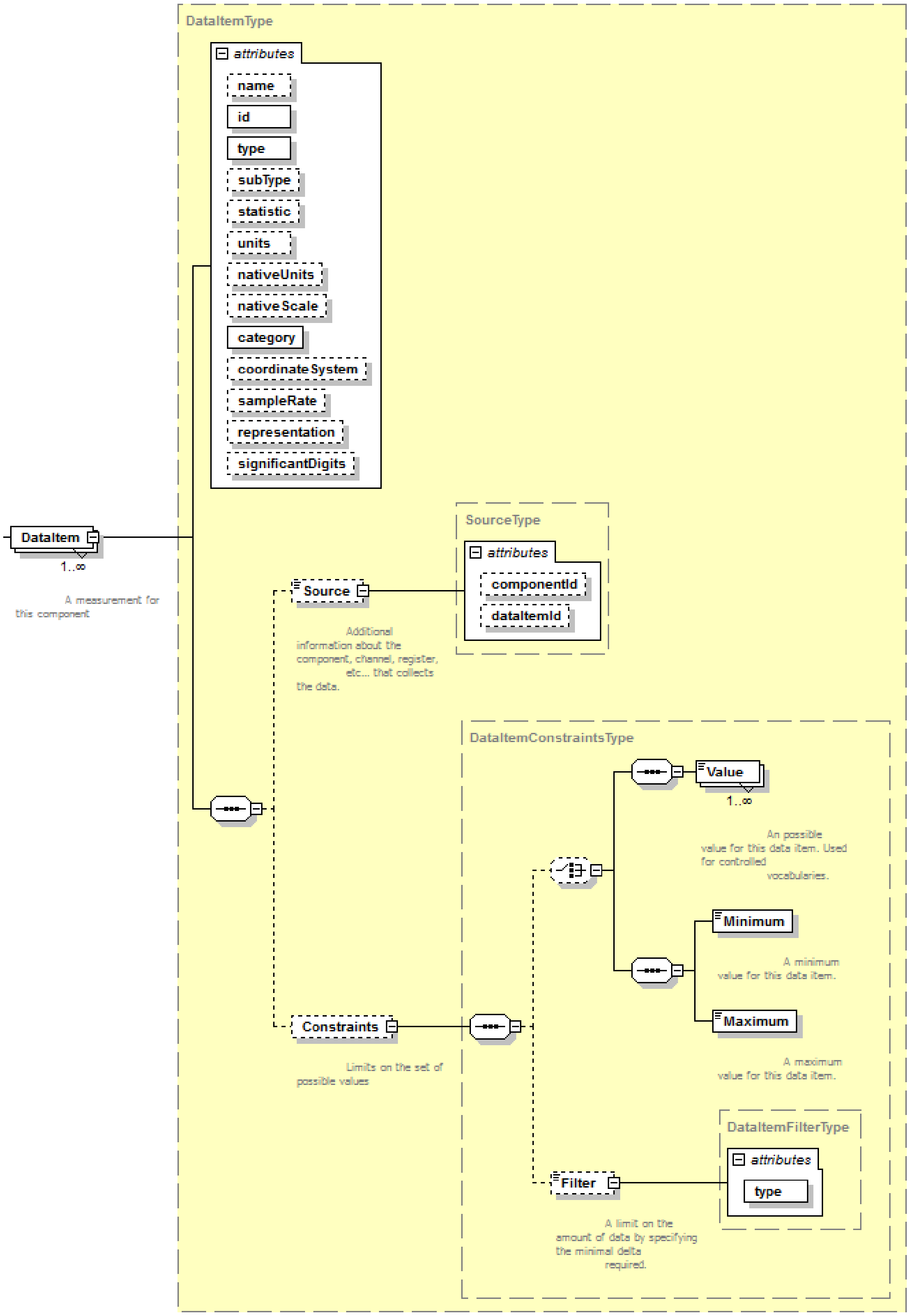
DataItemType
attributes
name
id
type
subType
statistic
units
nativeUnits
nativeScale
category
coordinateSystem
sampleRate
representation
significantDigits
DataItem
1..∞
A measurement for this component
SourceType
attributes
componentId
dataItemId
Source
Additional information about the component, channel, register, etc... that collects the data.
DataItemConstraintsType
Value
1..∞
An possible value for this data item. Used for controlled vocabularies.
Minimum
A minimum value for this data item.
Maximum
A maximum value for this data item.
Constraints
Limits on the set of possible values
DataItemFilterType
attributes
type
Filter
A limit on the amount of data by specifying the minimal delta required.

Figure 8: `DataItem` Schema Diagram

6.2.2 Attributes for a `DataItem`

The following table lists the attributes defined to provide information for a `DataItem` type XML Element.

`DataItem` **MUST** specify the `type` of data being collected, the `id` of the `DataItem`, and the `category` of the `DataItem`.

Attribute	Description	Occurrence
`id`	The unique identifier for this `DataItem`. The `id` attribute **MUST** be unique across the entire document including the ids for components. An XML ID-type.	1
`name`	The `name` of the `DataItem`. A `name` is provided as an additional human readable identifier for this `DataItem` in addition to the `id`. It is not required and will be implementation dependent. An NMTOKEN XML type.	0..1
`category`	Specifies the kind of information provided by a data item. Each category of information will provide similar characteristics in its representation. The available options are `SAMPLE`, `EVENT`, or `CONDITION`.	1
`type`	The `type` of data being measured. Examples of types are `POSITION`, `VELOCITY`, `ANGLE`, `BLOCK`, `ROTARY_VELOCITY`, etc.	1
`subType`	A sub-categorization of the data item type. For example, the Sub-types of `POSITION` can be `ACTUAL` or `COMMANDED`. Not all types have subTypes and they can be optional.	0..1
`statistic`	Data calculated specific to a `DataItem`. Examples of `statistic` are `AVERAGE`, `MINIMUM`, `MAXIMUM`, `ROOT_MEAN_SQUARE`, `RANGE`, `MEDIAN`, `MODE`, and `STANDARD_DEVIATION`.	0..1
`representation`	Data consisting of multiple data points or samples or a file presented as a single `DataItem`. Each representation will have a unique format defined for each `representation`. Examples of `representation` are `VALUE`, `TIME_SERIES`, `DISCRETE`, `MP3`, `WAV`, etc. Initially, the representation for `TIME_SERIES`, `DISCRETE`, and `VALUE` are defined. If a `representation` is not specified, it **MUST** be determined to be `VALUE`.	0..1

Attribute	Description	Occurrence
units	Units **MUST** be present for all DataItem elements in the SAMPLE category. If the data represented by a DataItem is a numeric value, except for line number and count, the units **MUST** be specified.	0..1
nativeUnits	The native units used by the Component. These units will be converted before they are delivered to the application.	0..1
nativeScale	The multiplier for the native units. The received data **MAY** be divided by this value before conversion. If provided, the value **MUST** be numeric.	0..1
significantDigits	The number of significant digits in the reported value. This is used by applications to determine accuracy of values. This **SHOULD** be specified for all numeric values.	0..1
sampleRate	The rate at which successive samples of a DataItem are recorded. sampleRate is expressed in terms of samples per second. If the sampleRate is smaller than one, the number can be represented as a floating point number. For example, a rate 1 per 10 seconds would be 0.1	0..1**
coordinateSystem	The coordinate system being used. The available values for coordinateSystem are WORK and MACHINE.	0..1

6.2.2.1 id for a DataItem

Each DataItem **MUST** be identified with an identifier (id). The id attribute **MUST** be unique across the entire XML document for a device, including the ids for all Structural Elements. This unique id provides the information required by a client software application to identify each piece of data and correlate that data to its original meaning or function at the source device.

For example, an XML document may provide three different pieces of data representing the position of the axes on a machine (x axis position, y axis position, and z axis position). All three may be modeled in the XML document as Position type data items for the Axes components. The unique id allows the client software application to distinguish the data for each of the axes.

6.2.2.2 name for a DataItem

name is provided as an additional human readable identifier for a DataItem. It is not required and is implementation dependent

6.2.2.3 category for a DataItem

Many DataItem types provide two forms of data - a value (reported as either a SAMPLE or EVENT category) and a health status (reported as a CONDITION category). Therefore, each occurrence of a DataItem in the XML document **MUST** report a category attribute. This category attribute provides the information required by a client software application to determine the specific meaning of the data provided.

Each piece of data provided by a device **MUST** be identified with one of the following:

SAMPLE A SAMPLE is the reading of the value of a continuously variable or analog data value. A continuous value can be measured at any point-in-time and will always produce a result. An example of a continuous data value is the position of the Linear X Axis.

The data provided for a SAMPLE category data item is always a floating point number or integers that have an infinite number of possible values. This is different from a state or discrete type data item that has a limited number of possible values. A data item of category SAMPLE **MUST** also provide the units attribute.

EVENT An EVENT is a data value representing a discrete piece of information from the device. EVENT does not have intermediate values that vary over time, as does SAMPLE. An EVENT is information that, when provided at any specific point in time, represents the current state of the device.

There are two types of EVENT: those representing state, with two or more discrete values; and those representing messages that contain plain text data.

An example of a state type EVENT is the value of the data item DOOR_STATE which can be OPEN, UNLATCHED, or CLOSED. (Note: No other values are valid to represent the value of DOOR_STATE.)

An example of a message type EVENT is the value for a data item PROGRAM. The value representing PROGRAM can be any valid string of characters.

CONDITION A CONDITION is a data item that communicates information about the health of a device and its ability to function. A valid value for a data item in the category CONDITION can be one of UNAVAILABLE, NORMAL, WARNING, or FAULT.

A data item of category CONDITION **MAY** report multiple values (CONDITION) at one time; whereas a DataItem of category SAMPLE or EVENT can only have a single value at any one point in time.

6.2.2.4 type and subType for a DataItem

type specifies the kind of information that is represented by the data item. Typical values for type include POSITION, VOLTAGE, CURRENT, PROGRAM, LINE, etc. type **MUST** be specified for every data item.

A data item **MAY** further qualify the data being provided by specifying a subType. subType is required for certain data item types. For example, POSITION has the subType of ACTUAL and COMMANDED. These are represented by two separate and different DataItem Type XML elements.

Section 7 of this document provides a detailed listing of the data item types and sub-types defined for each category of data item available for a device– SAMPLE, EVENT, and CONDITION.

6.2.2.5 statistic for a DataItem

Data reported by a device is normally provided as its original measured value or it may be scaled (see nativeScale below) to provide more meaning to the device or a software application. Some data types may be further processed by the device using a statistical calculation like average, mean, or square root and summary data resulting from this processing is provided. In this case, the statistic attribute **MAY** be used to indicate how the data has been processed.

statistic may be reported for any SAMPLE type DataItem. All statistic data is reported in the standard units of the DataItem.

statistic data is always the result of a calculation using data that has been measured over a specified period of time.

The value of statistic may be periodically reset. When a device reports a DataItem with a value that is a statistic, the information provided in the XML document for that piece of data **MUST** include an additional attribute called duration. The attribute duration defines the period of time over which the statistic has been calculated. Refer to Part 3, Streams, of the MTConnect Standard for more information about duration.

The following are the types of statistic defined for a DataItem.

Statistic	Description
AVERAGE	Mathematical Average value calculated for the DataItem during the calculation period.
KURTOSIS	A measure of the “peakedness” of a probability distribution; i.e., the shape of the distribution curve.
MAXIMUM	Maximum or peak value recorded for the DataItem during the calculation period.
MEDIAN	The middle number of a series of numbers.
MINIMUM	Minimum value recorded for the DataItem during the calculation period.

Statistic	Description
MODE	The number in a series of numbers that occurs most often.
RANGE	Difference between the Maximum and Minimum value of a DataItem during the calculation period. Also represents Peak-to-Peak measurement in a waveform.
ROOT_MEAN_SQUARE	Mathematical Root Mean Value (RMS) value calculated for the DataItem during the calculation period.
STANDARD_DEVIATION	Statistical Standard Deviation value calculated for the DataItem during the calculation period.

6.2.2.6 representation for a DataItem

Some data types provide data that may consist of a series of values or a file of data, not a single value. Other data types provide data that may require additional information so that the data may be correctly understood by a client software application.

When such data is provided, the representation attribute **MUST** be used to define the format for the data provided.

The types of representation defined are provided in the table below.

Note: See Part 3, Streams, of the MTConnect Standard for more information on the structure and format of each representation.

Representation	Description
VALUE	The measured value of a SAMPLE. If no representation is specified for a DataItem, the representation **MUST** be determined to be VALUE.
TIME_SERIES	A series of sampled data. The data is collected for a specified number of samples and each SAMPLE is collected with a fixed period.
DISCRETE	A data type where each discrete occurrence of the data may have the same value as the previous occurrence of the data. There is no reported state change between occurrences of the data. In this case, duplicate occurrences of the same data value **SHOULD NOT** be suppressed. Examples of a DISCRETE data type would be a Parts Counter that reports the completion of each part, versus the accumulation of parts. Also, Message does not typically have a reset state and may re-occur each time a specific message is triggered.

6.2.2.7 units for a DataItem

The following table lists the units that are defined as the standard unit of measure for each type of DataItem.

Units	Description
AMPERE	Amps
CELSIUS	Degrees Celsius
COUNT	A counted event
DECIBEL	Sound Level
DEGREE	Angle in degrees
DEGREE/SECOND	Angular degrees per second
DEGREE/SECOND^2	Angular acceleration in degrees per second squared
HERTZ	Frequency measured in cycles per second
JOULE	A measurement of energy.
KILOGRAM	Kilograms
LITER	Liters
LITER/SECOND	Liters per second
MICRO_RADIAN	Measurement of Tilt
MILLIMETER	Millimeters
MILLIMETER/SECOND	Millimeters per second
MILLIMETER/SECOND^2	Acceleration in millimeters per second squared
MILLIMETER_3D	A point in space identified by X, Y, and Z positions and represented by a space delimited set of numbers each expressed in millimeters.
NEWTON	Force in Newtons
NEWTON_METER	Torque, a unit for force times distance.
OHM	Measure of Electrical Resistance
PASCAL	Pressure in Newtons per square meter
PASCAL_SECOND	Measurement of Viscosity
PERCENT	Percentage
PH	A measure of the acidity or alkalinity of a solution
REVOLUTION/MINUTE	Revolutions per minute
SECOND	A measurement of time.
SIEMENS/METER	A measurement of Electrical Conductivity
VOLT	Volts

Units	Description
VOLT_AMPERE	Volt-Ampere (VA)
VOLT_AMPERE_REACTIVE	Volt-Ampere Reactive (VAR)
WATT	Watts
WATT_SECOND	Measurement of electrical energy, equal to one Joule

6.2.2.8 nativeUnits for a DataItem

The nativeUnits attribute provides additional information about the original measured value for a piece of data reported by a device. nativeUnits **MAY** be specified to provide additional information about the data if the units of the measured value supplied by the device differs from the value provided for that data when converted to standard units.

The following table defines the nativeUnits currently supported by the Device Information Model:

Native Units	Description
CENTIPOISE	A measure of Viscosity
DEGREE/MINUTE	Rotational velocity in degrees per minute
FAHRENHEIT	Temperature in Fahrenheit
FOOT	Feet
FOOT/MINUTE	Feet per minute
FOOT/SECOND	Feet per second
FOOT/SECOND^2	Acceleration in feet per second squared
FOOT_3D	A point in space identified by X, Y, and Z positions and represented by a space delimited set of numbers each expressed in feet.
GALLON/MINUTE	Gallons per minute.
INCH	Inches
INCH/MINUTE	Inches per minute
INCH/SECOND	Inches per second
INCH/SECOND^2	Acceleration in inches per second squared
INCH_3D	A point in space identified by X, Y, and Z positions and represented by a space delimited set of numbers each expressed in inches.
INCH_POUND	A measure of torque in inch pounds.
KELVIN	A measurement of temperature
KILOWATT	A measurement in kilowatt.
KILOWATT_HOUR	Kilowatt hours which is 3.6 mega joules.

Native Units	Description
LITER	Measurement of volume of a fluid
LITER/MINUTE	Measurement of rate of flow of a fluid
MILLIMETER/MINUTE	Velocity in millimeters per minute
POUND	US pounds
POUND/INCH^2	Pressure in pounds per square inch (PSI).
RADIAN	Angle in radians
RADIAN/SECOND	Velocity in radians per second
RADIAN/SECOND^2	Rotational acceleration in radian per second squared
RADIAN/MINUTE	Velocity in radians per minute.
REVOLUTION/SECOND	Rotational velocity in revolution per second
OTHER	Unsupported units

6.2.2.9 nativeScale for a DataItem

The units of measure for some values at the source device may be different from the nativeUnits defined in 6.2.2.8 above. In the cases where the units of measure uses a different weighting or range than is provided by nativeUnits,the nativeScale attribute can be used to define the original units of measure.

As an example, a velocity measured in units of 100 ft/min can be represented as nativeUnits=“FEET/MINUTE” and nativeScale=“100”.

6.2.2.10 significantDigits for a DataItem

significantDigits is used to specify the level of accuracy (number of significant digits) for the value provided for a DataItem.

significantDigits is used by a client software application to determine accuracy of values provided in the XML document for a DataItem.

significantDigits attribute is not required for a DataItem, but it is recommended and **SHOULD** be used for any DataItem reporting a numeric value.

6.2.2.11 sampleRate for a DataItem

The value for some data types provided by a device may be collected at the device or reported by the device at specific intervals of time. When such data is provided, the sampleRate defines the rate at which successive samples of data are recorded.

The sampleRate attribute provides the information required by a client software application to interpret the data and the sampling time relationship between successive values reported for the data.

sampleRate is expressed in terms of samples per second. If the sample rate is smaller than one, the number can be represented as a floating point number. For example, a rate 1 per 10 seconds would be 0.1

6.2.2.12 coordinateSystem for a DataItem

The values reported by a device for some types of data will be in reference to a specific positioning measurement system used by the device. The coordinateSystem attribute **MAY** be used to specify the coordinate system used to measure the reported value.

The coordinateSystem attribute is used by a client software application to interpret the spacial relationship between values reported by a device.

If coordinateSystem is not provided, all values representing positional data for Axes **MUST** be interpreted using the MACHINE coordinate system and all values representing positional data for Path **MUST** be interpreted using the WORK coordinate system

The following table defines the types of coordinateSystem currently supported by the Device Information Model:

Coordinate System	Description
MACHINE	An unchangeable coordinate system that has machine zero as its origin.
WORK	The coordinate system that represents the working area for a particular workpiece whose origin is shifted within the MACHINE coordinate system. If the WORK coordinates are not currently defined in the device, the MACHINE coordinates will be used.

6.2.3 Sub-Elements for a DataItem

The following table lists the sub-elements defined to provide additional information for a DataItem type XML Element.

Element	Description	Occurrence
Source	Source is an XML element that identifies the Component, *Subcomponent*, or DataItem representing the part of the device from which a measured value originates.	0..1
Constraints	The set of possible values that can be assigned to this DataItem.	0..1

6.2.3.1 Source for a DataItem

Source identifies the physical part of a device where the data represented by the DataItem is originally measured.

As an example, data related to a servo motor on an Axes component may actually originate from a measurement made in the controller.

The following XML tree represents the structure of the Source XML sub-element element showing the attributes defined for Source.

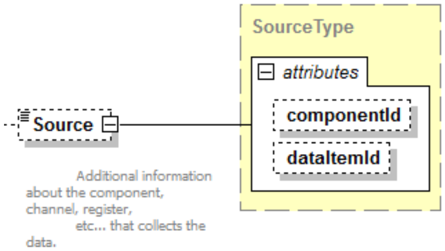
SourceType
attributes
componentId
dataItemId
Source
Additional information about the component, channel, register, etc... that collects the data.

Figure 9: Source Schema Diagram

6.2.3.1.1 Attributes for Source

The following table identifies the attributes available to identify Source for a measured value:

Attribute	Description	Occurrence
componentID	The id attribute of the Component that represents the physical part of a device where the data represented by the DataItem is actually measured.	0..1
dataItemID	The id attribute of the DataItem that represents the originally measured value of the data referenced by this DataItem.	0..1

6.2.3.2 Constraints for a DataItem

For some types of DataItem elements, the value(s) for the data provided for the DataItem **MAY** be restricted to specific values or a range of values.

Constraints provides a way to define the allowable value(s) or the upper and lower limits for the range of values that can be reported for the data by an MTConnect Agent in response to a Current or Sample request. Constraints also provides a means to suppress multiple occurrences of data values where the change in value is below a threshold defined by a Filter attribute. This is effective to reduce the amount of data generated by a "noisy" data source.

The following XML tree represents the structure of the `Constraints` XML element and the sub-elements defined for `Constraints`.

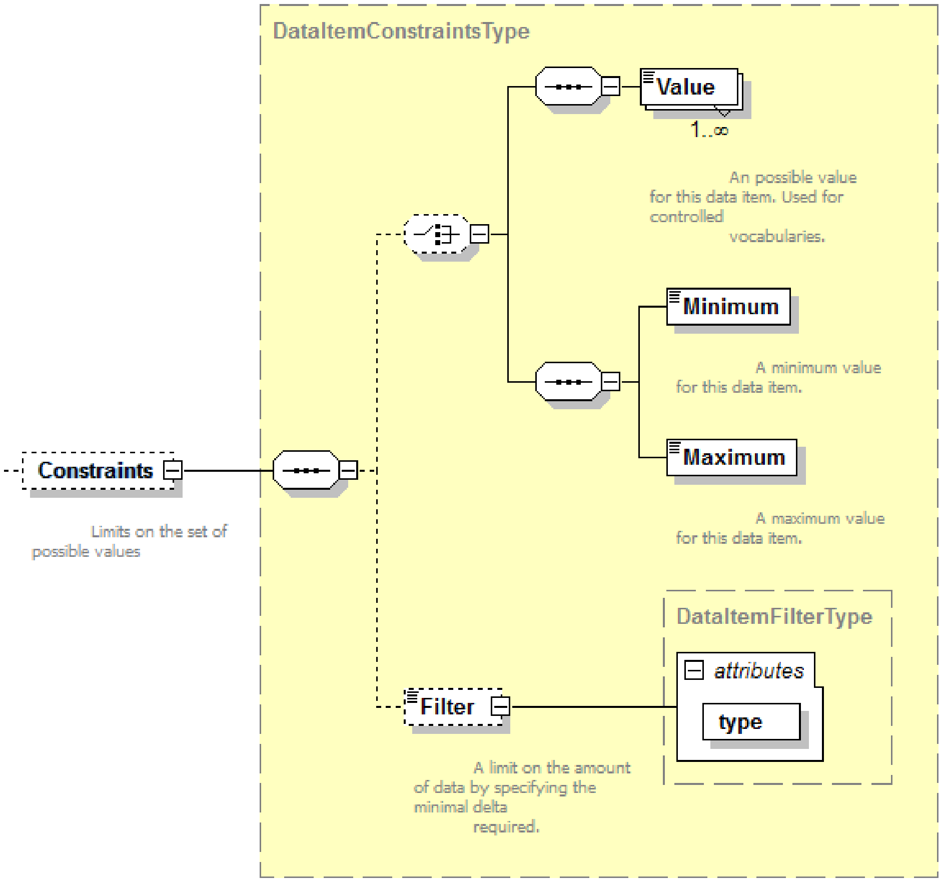
DataItemConstraintsType
Value
1..∞
An possible value for this data item. Used for controlled vocabularies.
Minimum
A minimum value for this data item.
Maximum
A maximum value for this data item.
Constraints
Limits on the set of possible values
DataItemFilterType
attributes
type
Filter
A limit on the amount of data by specifying the minimal delta required.

Figure 10: `Constraints` Schema

The following table identifies the sub-elements available to identify `Constraints` for a measured value:

Element	Description	Occurrence
`Value`	A Data Element that defines a valid value for the data provided for a `DataItem`. When the data reported for a `DataItem` is a descriptive type of data (not numeric data), then `Value` **MAY** be used to define a valid desriptor for the `DataItem`. Multiple `Value` Data Elements may be defined for any `DataItem` and each valid value **MUST** be defined by a `Value` Data Element. If there is only one `Value` Data Element defined for a `DataItem`, the value will be constant and cannot change. In the case of a constant value, the value is not required to be supplied in the XML document provided by an MTConnect Agent in response to a `Current` or `Sample` request.	0..INF
`Maximum`	If data reported for a `DataItem` is a range of numeric values, then the value reported **MAY** be bounded with an upper limit defined by this constraint.	0..1
`Minimum`	If the data reported for a `DataItem` is a range of numeric values, the value reported **MAY** be bounded with a lower limit defined by this constraint.	0..1
`Filter`	If the data reported for a `DataItem` is a numeric value, a new value **MUST NOT** be reported if the change from the last reported value is less than the delta given as the CDATA of this element. `Filter` is an abstract type XML element. As such, `Filter` will never appear in the XML document, but will be replaced by a `Filter` type. The only currently supported `Filter` type is `MINIMUM_DELTA`. The `CDATA` **MUST** be an absolute value using the same Units as the reported data. Additional filter types **MAY** be supported in the future.	0..1

6.2.4 Example Schema Structure for DataItem

The following sample XML type document structure shows how Structural Elements and Data Elements are combined to represent a typical machine with rotary and linear axes and a controller.

```
MTConnectDevices
   Devices
      Device
         Components
            Axes
               Rotary [C]
                  DataItems
                     DataItem [Cvel]
                        Constraints SPINDLE
               Linear  [X]
                  DataItems
                     DataItem [Xpos]
               Linear  [Y]
                  DataItems
                     DataItem [Ypos]
               Linear  [Z]
                  DataItems
                     DataItem [Zpos]
            Controller
               Path
                  DataItems
                     DataItem [mode]
                     DataItem [execution]
```

6.3 References

References is an XML Data Element that may be modeled as part of an Interface type Structural Element, e.g. BarFeederInterface or MaterialHandlerInterface. References provides an efficient method of organizing data required by an Interface where that data is associated with other Structural Elements and is already defined elsewhere in the XML document.

References is also a container type XML element. As a container, it is used to organize each of the pieces of data belonging to other Structural Elements which are required by an Interface.

The References container is comprised of one or more Reference XML Elements.

6.4 Reference

A Reference XML Element acts as a pointer to information that is associated with other Structural Elements and provides a copy of the value of that information as part of the data set provided for an Interface.

The following is an example of the use of the Reference XML Element:

> *The data set for the DoorInterface component must include the value of the DOOR_STATE data element from the Door component. If the Reference XML Element were not used, it would be necessary to either duplicate the DOOR_STATE data element as part of the DoorInterface component or violate the structure of the XML data model defined in Section 6 by moving the DOOR_STATE data element from the Door component to the DoorInterface component. Reference provides a means to provide a copy of the value of the DOOR_STATE data element from the Door component to be included in the data set provided for the DoorInterface component.*

6.4.1 XML Schema Structure for a Reference

The following XML tree represents the structure of an Interface XML element showing the Reference sub-elements that may be associated with an Interface.

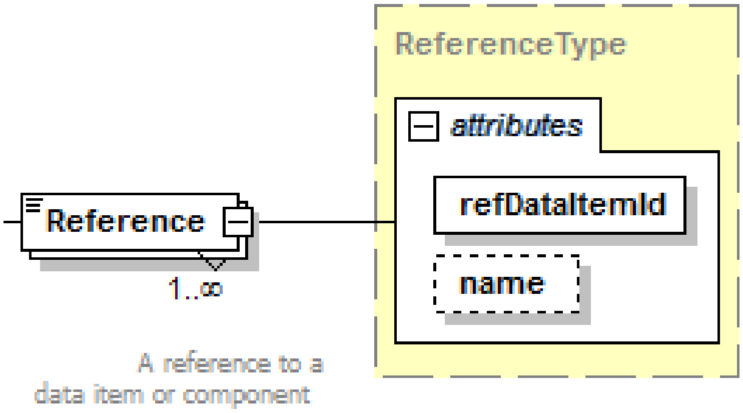

Figure 5: Reference Schema

The following table lists the attributes defined for the Reference XML sub-element.

Attribute	Description	Occurrence
name	An optional name foe the data element to provide a human readable identifier of the reference.	0..1
dataItemId	The id attribute of the DataItem that represents the originally measured value of the data provided by the Interface.	1

7 DataItem Types

As described in *Section 5* of this document, DataItem is an abstract type XML Element. As such, DataItem will be replaced in the XML document by specific DataItem types.

In the MTConnect Standard, DataItem types are grouped into categories based on the type of information that they describe. These categories are:

SAMPLE A SAMPLE is the reading of the value of a continuously variable or analog data value.

EVENT An EVENT is a data value representing a discrete piece of information from the device. The data provided may be a numeric value or text.

There are two types of EVENT: those representing state, with two or more discrete values, and those representing messages (text).

CONDITION A CONDITION communicates information about the health of a device and its ability to function.

Many DataItem types provide two forms of data - a value (reported as either a SAMPLE or EVENT) and a health status (reported as a CONDITION). These DataItem types and the data that they represent **MAY** be defined in more than one category.

The following sections define the DataItem types that are available in each of the above categories.

7.1 DataItem Types for SAMPLE Category

DataItem types in the SAMPLE Category report data representing a continuously changing or analog data value. This data can be measured at any point-in-time and will always produce a result. The data provided may be a scalar floating point number or integers that have an infinite number of possible values. All possible numeric data values **MUST** be considered valid unless the valid values are restricted by Constraints Data Elements. The units attribute **MUST** be defined and reported for each DataItem in this category.

The table below defines the following for each of the DataItem types defined for the SAMPLE category:

- type attribute (**bold text**)
- subType attribute, if applicable. (indented in normal text)
- units attribute defining the standard unit of measure for the reported values

Data Item type/subType	Description	Units
ACCELERATION	Rate of change of velocity	MILLIMETER/SECOND^2
ACCUMULATED_TIME	The measurement of accumulated time for an activity or event	SECOND
ANGULAR_ACCELERATION	Rate of change of angular velocity.	DEGREE/SECOND^2
ANGULAR_VELOCITY	Rate of change of angular position.	DEGREE/SECOND
AMPERAGE	The measurement of electrical current	AMPERE
ALTERNATING	The measurement of alternating current. If not specified further in statistic, defaults to RMS current	AMPERE
DIRECT	The measurement of DC current	AMPERE
ANGLE	The measurement of angular position	DEGREE
ACTUAL	The actual angular position as read from the physical component.	DEGREE
COMMANDED	A calculated value for angular position computed by the Controller type component	DEGREE
AXIS_FEEDRATE	The feedrate of a linear axis.	MILLIMETER/SECOND
ACTUAL	The measured value of the feedrate of a linear axis.	MILLIMETER/SECOND
COMMANDED	The feedrate of a linear axis as specified by the Controller type Component. The COMMANDED feedrate is a calculated value that includes adjustments and overrides.	MILLIMETER/SECOND
JOG	The feedrate specified by a logic or motion program, by a pre-set value, or set by a switch as the feedrate for a linear axis when operating in a manual state or method (jogging).	MILLIMETER/SECOND
PROGRAMMED	The feedrate specified by a logic or motion program or set by a switch for a linear axis.	MILLIMETER/SECOND

Data Item type/subType	Description	Units
RAPID	The feedrate specified by a logic or motion program, by a pre-set value, or set by a switch as the feedrate for a linear axis when operating in a rapid positioning mode.	MILLIMETER/SECOND
~~OVERRIDE~~	~~The operator's overridden value. Percent of commanded~~. Deprecated in Rel. 1.3. See EVENT Type DataItems.	~~PERCENT~~
CLOCK_TIME	The value provided by a timing device at a specific point in time. CLOCK_TIME **MUST** be reported in W3C ISO 8601 format.	YYYY-MM-DDThh:mm:ss.ffff
CONCENTRATION	Percentage of one component within a mixture of components	PERCENT
CONDUCTIVITY	The ability of a material to conduct electricity	SIEMENS/METER
DISPLACEMENT	The change in position of an object	MILLIMETER
ELECTRICAL_ENERGY	The measurement of electrical energy consumption by a component	WATT_SECOND
FILL_LEVEL	The measurement of the amount of a substance remaining compared to the planned maximum amount of that substance	PERCENT
FLOW	The rate of flow of a fluid	LITER/SECOND
FREQUENCY	The measurement of the number of occurrences of a repeating event per unit time	HERTZ
~~**GLOBAL_POSITION**~~	DEPRECATED in Rel. 1.1	

Data Item type/subType	Description	Units
~~**LEVEL**~~	DEPRECATED in Rel. 1.2 See FILL_LEVEL	
LENGTH	The length of an object	MILLIMETER
STANDARD	The standard or original length of an object	MILLIMETER
REMAINING	The remaining total length of an object.	MILLIMETER
USEABLE	The remaining useable length of an object.	MILLIMETER
LINEAR_FORCE	The measure of the push or pull introduced by an actuator or exerted on an object	NEWTON
LOAD	The measurement of the actual versus the standard rating of a device	PERCENT
MASS	The measurement of the mass of an object(s) or an amount of material	KILOGRAM
PATH_FEEDRATE	The feedrate for the axes associated with a Path component - may represent a single axis or the coordinated movement of multiple axes – a vector.	MILLIMETER/SECOND
ACTUAL	The measured value of the feedrate of the axes associated with a Path component.	MILLIMETER/SECOND
COMMANDED	The feedrate as specified by the Controller type component for the axes associated with a Path component. The COMMANDED feedrate is a calculated value that includes adjustments and overrides.	MILLIMETER/SECOND
JOG	The feedrate specified by a logic or motion program, by a pre-set value, or set by a switch as the feedrate for the axes associated with a Path when operating in a manual state or method (jogging).	MILLIMETER/SECOND

Data Item type/subType	Description	Units
PROGRAMMED	The feedrate specified by a logic or motion program or set by a switch as the feedrate for the axes associated with a Path.	MILLIMETER/SECOND
RAPID	The feedrate specified by a logic or motion program, by a pre-set value, or set by a switch as the feedrate for the axes associated with a Path when operating in a rapid positioning mode.	MILLIMETER/SECOND
~~OVERRIDE~~	~~The operator's overridden value. Percent of commanded.~~ Deprecated in Rel. 1.3. See EVENT Type DataItems.	~~PERCENT~~
PATH_POSITION	The current program control point or program coordinate in WORK coordinates. The coordinate system will revert to MACHINE coordinates if WORK coordinates are not available.	MILLIMETER_3D
ACTUAL	The position of the Component as read from the device.	MILLIMETER_3D
COMMANDED	The position computed by the Controller type Component	MILLIMETER_3D
TARGET	The desired end position for a movement or a series of movements. Multiple discrete movements may need to be completed to achieve the final TARGET position.	MILLIMETER_3D
PROBE	The position provided by a probe	MILLIMETER_3D
PH	The measure of the acidity or alkalinity.	PH

Data Item type/subType	Description	Units
POSITION	The position of the COMPONENT. Defaults to MACHINE coordinates. When POSITION type data is provided representing a measured value for the physical axes of the device, this data **MUST** be given in MACHINE coordinates. When POSITION type data is provided representing a logical or calculated location on the device, this data **MUST** be given in WORK coordinates and is associated with the PATH element of the CONTROLLER.	MILLIMETER
ACTUAL	The physical position of the COMPONENT.	MILLIMETER
COMMANDED	A position calculated by the Controller type Component for a discrete movement.	MILLIMETER
TARGET	The desired end position of a Component resulting from a movement or a series of movements. Multiple discrete movements may need to be completed to achieve the final TARGET position.	MILLIMETER
POWER_FACTOR	The measurement of the ratio of real power flowing to a load to the apparent power in that AC circuit.	PERCENT
PRESSURE	The force per unit area exerted by a gas or liquid	PASCAL
RESISTANCE	The measurement of the degree to which an object opposes an electric current through it	OHM
ROTARY_VELOCITY	The rotational speed of a rotary axis.	REVOLUTION/MINUTE
ACTUAL	The measured value of rotational speed that the rotary axis is spinning.	REVOLUTION/MINUTE

Data Item type/subType	Description	Units
COMMANDED	The rotational speed as specified by the Controller type Component. The COMMANDED velocity is a calculated value that includes adjustments and overrides.	REVOLUTION/MINUTE
PROGRAMMED	The rotational velocity specified by a logic or motion program or set by a switch	REVOLUTION/MINUTE
~~OVERRIDE~~	~~The operator's overridden value. Percent of commanded.~~ Deprecated in Rel. 1.3. See EVENT Type DataItems.	~~PERCENT~~
SOUND_LEVEL	Measurement of a sound level or sound pressure level relative to atmospheric pressure	DECIBEL
NO_SCALE	No weighting factor on the frequency scale	DECIBEL
A_SCALE	A Scale weighting factor. This is the default weighting factor if no factor is specified	DECIBEL
B_SCALE	B Scale weighting factor	DECIBEL
C_SCALE	C Scale weighting factor	DECIBEL
D_SCALE	D Scale weighting factor	DECIBEL
~~**SPINDLE_SPEED**~~	DEPRECATED in REL 1.2. Replaced by ROTARY_VELOCITY	
~~ACTUAL~~	~~The rotational speed of a rotary axis. ROTARY_MODE **MUST** be SPINDLE.~~	~~REVOLUTION/MINUTE~~
~~COMMANDED~~	~~The rotational speed the as specified by the Controller type Component.~~	~~REVOLUTION/MINUTE~~
~~OVERRIDE~~	~~The operator's overridden value. Percent of commanded.~~	~~PERCENT~~
STRAIN	The amount of deformation per unit length of an object when a load is applied.	PERCENT
TEMPERATURE	The measurement of temperature	CELSIUS
TILT	A measurement of angular displacement	MICRO_RADIAN
TORQUE	The turning force exerted on an object or by an object	NEWTON_METER

Data Item type/subType	Description	Units
VOLT_AMPERE	The measure of the apparent power in an electrical circuit, equal to the product of root-mean-square (RMS) voltage and RMS current’ (commonly referred to as VA)	VOLT_AMPERE
VOLT_AMPERE_REACTIVE	The measurement of reactive power in an AC electrical circuit (commonly referred to as VAR)	VOLT_AMPERE_REACTIVE
VELOCITY	The rate of change of position.	MILLIMETER/SECOND
VISCOSITY	A measurement of a fluid’s resistance to flow	PASCAL_SECOND
VOLTAGE	The measurement of electrical potential between two points	VOLT
ALTERNATING	The measurement of alternating voltage. If not specified further in statistic, defaults to RMS voltage	VOLT
DIRECT	The measurement of DC voltage	VOLT
WATTAGE	The measurement of power consumed or dissipated by an electrical circuit or device	WATT

7.2 DataItem Types for EVENT Category

DataItem Types in the EVENT category represent a discrete piece of information from a device. EVENT does not have intermediate values that vary over time, as does SAMPLE. An EVENT is information that, when provided at any specific point in time, represents the current state of the device.

There are two types of EVENT: those representing state, with two or more discrete values; and those representing messages that contain plain text data.

The table below defines the following for each of the DataItem types defined for the EVENT Category:

- type attribute (**bold text**)
- subType attribute, if applicable (indented in normal text)
- Allowable values for the State(s) represented by the DataItem. (All CAPS)

Note: DataItem types in the EVENT category do not have any units since these values for the data are not scalar.

833

Data Item type/subType	Description
ACTUATOR_STATE	The state of an Actuator. State **MUST** be ACTIVE or INACTIVE.
ALARM	DEPRECATED: Replaced with CONDITION category. *Rel. 1.1*.
ACTIVE_AXES	The set of axes currently associated with a Path and the Controller Structural Elements. If this DataItem is not provided, it will be assumed that all axes are currently associated with the Controller Structural Element and with an individual Path. The value will be a space delimited set of axes names.
AVAILABILITY	Represents the ability of a Structural Element to communicate. This **MUST** be provided for a Device Element and **MAY** be provided for any other Structural Element type element. State **MUST** be AVAILABLE or UNAVAILABLE.
AXIS_COUPLING	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 valid States are: TANDEM, SYNCHRONOUS, MASTER, and SLAVE. The coupling **MUST** be viewed from the perspective of the axis. Therefore a MASTER coupling indicates that this axis is the master for the COUPLED_AXES.

Data Item type/subType	Description
AXIS_FEEDRATE_OVERRIDE	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 states of operation of an axis are currently defined as PROGRAMMED, JOG, and RAPID.
JOG	The value of a signal or calculation issued to adjust the feedrate of an individual linear type axis when that axis is being operated in a manual state or method (jogging). When the JOG subtype of AXIS_FEEDRATE_OVERRIDE is applied, the resulting commanded feedrate for the axis is limited to the value of the original JOG subtype of the AXIS_FEEDRATE multiplied by the value of the JOG subtype of AXIS_FEEDRATE_OVERRIDE.
PROGRAMMED	The value of a signal or calculation issued to adjust the feedrate of an individual linear type axis that has been specified by a logic or motion program or set by a switch. When the PROGRAMMED subtype of AXIS_FEEDRATE_OVERRIDE is applied, the resulting commanded feedrate for the axis is limited to the value of the original PROGRAMMED subtype of the AXIS_FEEDRATE multiplied by the value of the PROGRAMMED subtype of AXIS_FEEDRATE_OVERRIDE.
RAPID	The value of a signal or calculation issued to adjust the feedrate of an individual linear type axis that is operating in a rapid positioning mode. When the RAPID subtype of AXIS_FEEDRATE_OVERRIDE is applied, the resulting commanded feedrate for the axis is limited to the value of the original RAPID subtype of the AXIS_FEEDRATE multiplied by the value of the RAPID subtype of AXIS_FEEDRATE_OVERRIDE.

AXIS_INTERLOCK	An indicator of the state of the axis lockout function when power has been removed and the axis is allowed to move freely. The values **MUST** be ACTIVE or INACTIVE.
AXIS_STATE	An indicator of the controlled state of an Axis *Subcomponent*. The value **MUST** be on of HOME, TRAVEL, PARKED, or STOPPED.
BLOCK	The block of code being executed. BLOCK contains the entire expression for a line of program code.
CHUCK_INTERLOCK	An indication of the state of an interlock function or control logic state intended to prevent the associated CHUCK component from being operated. The values **MUST** be ACTIVE or INACTIVE.
MANUAL_UNCLAMP	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. The values **MUST** be ACTIVE or INACTIVE. When MANUAL_UNCLAMP is ACTIVE, it is expected that a chuck cannot be unclamped until MANUAL_UNCLAMP is set to INACTIVE.
CHUCK_STATE	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 device. The value **MUST** be one of OPEN, CLOSED, or UNLATCHED.
~~CODE~~	DEPRECATED. Rel 1.1.
CONTROLLER_MODE	The current mode of the Controller. The value MUST be one of AUTOMATIC, MANUAL, MANUAL_DATA_INPUT, SEMI_AUTOMATIC, or EDIT
COUPLED_AXES	Refers to the set of associated axes. The value will be a space delimited set of axes names.
DIRECTION	The direction of motion. A subType **MUST** always be specified.
ROTARY	The rotational direction of a rotary device using the right hand rule convention. State **MUST** be CLOCKWISE or COUNTER_CLOCKWISE
LINEAR	The direction of motion of a linear device. State **MUST** be POSTIVE or NEGATIVE
DOOR_STATE	The opened or closed state of the door. State **MUST** be OPEN, UNLATCHED, or CLOSED.

Data Item type/subType	Description
END_OF_BAR	An indication of whether the end of a piece of bar stock being feed by a bar feeder has been reached. The value **MUST** be expressed as a Boolean state of YES or NO.
PRIMARY	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. If no sub-type is specified, PRIMARY **MUST** be the default End_of_Bar indication.
AUXILIARY	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.
EMERGENCY_STOP	The current state of the emergency stop signal. State **MUST** be ARMED (the circuit is complete and the device is allowed to operate) or TRIGGERED (the circuit is open and the device MUST cease operation).
EXECUTION	The execution status of the Controller. State **MUST** be READY, ACTIVE, INTERRUPTED, FEED_HOLD, STOPPED, OPTIONAL_STOP, PROGRAM_STOPPED, or PROGRAM_COMPLETED.

Data Item type/subType	Description
FUNCTIONAL_MODE	The current intended production status of the device or component. Typically, the FUNCTIONAL_MODE **SHOULD** be modeled as a data item for the Device Element, but **MAY** be modeled for any Structural Element in the XML document. The value **MUST** be PRODUCTION, SETUP, TEARDOWN, MAINTENANCE, or PROCESS_DEVELOPMENT.
INTERFACE_STATE	The current functional or operational state of an Interface type element indicating whether the interface is active or not currently functioning. The values **MUST** be ENABLED or DISABLED. When the INTERFACE_STATE is DISABLED, the state of all other data elements associated with that Interface **MUST** be set to NOT_READY.
LINE	The current line of code being executed. The data will be an alpha numeric value representing the line number of the current line of code being executed.
MAXIMUM	The maximum line number of the code being executed.
MINIMUM	The minimum line number of the code being executed.
MESSAGE	Any text string

Data Item type/subType	Description
OPERATOR_ID	The identifier of the person currently responsible for operating the device.
PALLET_ID	The identifier for the pallet currently in use. The data **MUST** be any text string.
PART_COUNT	The current count of parts produced as represented by the Controller. The data **MUST** be an integer value.
ALL	The count of all the parts produced. If the subtype is not given, this is the default.
GOOD	Indicates the count of correct parts made.
BAD	Indicates the count of incorrect parts produced.
TARGET	Indicates the number of parts that are projected or planned to be produced
REMAINING	The number of parts remaining in stock or to be produced.
PART_ID	An identifier of the current part in the device. The data **MUST** be any text string.
PATH_FEEDRATE_OVERRIDE	The value of a signal or calculation issued to adjust the feedrate for the axes associated with a Path component - may represent a single axis or the coordinated movement of multiple axes. The value provided for PATH_FEEDRATE_OVERRIDE is expressed as a percentage of the designated feedrate for the path. When PATH_FEEDRATE_OVERRIDE is applied, the resulting commanded feedrate for the path is limited to the value of the original feedrate multiplied by the value of the PATH_FEEDRATE_OVERRIDE. There MAY be different subtypes of PATH_FEEDRATE_OVERRIDE , each representing an override value for a designated subtype of feedrate depending on the state of operation of the path. The states of operation of a path are currently defined as PROGRAMMED, JOG, and RAPID.
JOG	The value of a signal or calculation issued to adjust the feedrate of the axes associated with a Path component when the axes (axis) are being operated in a manual mode or method (jogging). When the JOG subtype of PATH_FEEDRATE_OVERRIDE is applied, the resulting commanded feedrate for the axes(axis) associated with the path are limited to the value of the original JOG subtype of the PATH_FEEDRATE multiplied by the value of the JOG subtype of PATH_FEEDRATE_OVERRIDE.

Data Item type/subType	Description
PROGRAMMED	The value of a signal or calculation issued to adjust the feedrate of the axes associated with a Path component when the axes (axis) are operating as specified by a logic or motion program or set by a switch. When the PROGRAMMED subtype of PATH_FEEDRATE_OVERRIDE is applied, the resulting commanded feedrate for the axes(axis) associated with the path are limited to the value of the original PROGRAMMED subtype of the PATH_FEEDRATE multiplied by the value of the PROGRAMMED subtype of PATH_FEEDRATE_OVERRIDE.
RAPID	The value of a signal or calculation issued to adjust the feedrate of the axes associated with a Path component when the axes (axis) are being operated in a rapid positioning mode or method (rapid). When the RAPID subtype of PATH_FEEDRATE_OVERRIDE is applied, the resulting commanded feedrate for the axes(axis) associated with the path are limited to the value of the original RAPID subtype of the PATH_FEEDRATE multiplied by the value of the RAPID subtype of PATH_FEEDRATE_OVERRIDE.
PATH_MODE	The operational mode for this Path. State **MUST** be INDEPENDENT, MASTER, SYNCHRONOUS, or MIRROR. The default value **MUST** be INDEPENDENT if PATH_MODE is not specified.
POWER_STATE	The indication of the status of the source of energy for a Structural Element to allow it to perform its intended function and the state of an enabling signal providing permission for the Structural Element to perform its functions. State **MUST** be ON or OFF. DEPRECATION WARNING: **MAY** be deprecated in the future.
LINE	The state of the power source for the Structural Element.
CONTROL	The state of the enabling signal or control logic that enables or disables the function or operation of the Structural Element.
POWER_STATUS	DEPRECATED. *Rel. 1.1.*
PROGRAM	The name of the program being executed by the Controller component. The data **MUST** be any text string.

PROGRAM_EDIT	An indication of the Controller component's program editing mode. On many controls, a program can be edited while another program is currently being executed. The value MUST be: ACTIVE: The controller is in the program edit mode. READY: The controller is capable of entering the program edit mode and no function is inhibiting a change of mode. NOT_READY: A function is inhibiting the controller from entering the program edit mode.
PROGRAM_EDIT_NAME	The name of the program being edited. This is used in conjunction with PROGRAM_EDIT when in ACTIVE state. The data **MUST** be any text string.
PROGRAM_COMMENT	A comment or non-executable statement in the control program. The data **MUST** be any text string.
PROGRAM_HEADER	The non-executable header section of the control program. The data **MUST** be any text string.
ROTARY_MODE	The mode for a Rotary type axis. State **MUST** be SPINDLE, INDEX, or CONTOUR.
ROTARY_VELOCITY_OVERRIDE	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.
~~TOOL_ID~~	DEPRECATED in Rel. 1.2. See Tool_ASSET_ID. ~~The identifier of the tool currently in use for a given Path~~
TOOL_ASSET_ID	The identifier of an individual tool asset. The data **MUST** be any text string.
TOOL_NUMBER	The identifier of a tool provided by the device controller. The data **MUST** be any text string.
WORKHOLDING_ID	The identifier for the workholding currently in use. The data **MUST** be any text string.

7.2.1 EVENT Category DataItem Types Specific for Interface

MTConnect provides the means to read information from a piece of equipment, but it does not provide a mechanism for one piece of equipment to request another piece of equipment to perform a task. To enable the coordination of actions between two pieces of equipment, special data types have been defined to provide information from a piece of equipment that indicates that it has a requirement for a service or services to be performed by a second piece of equipment. As an example, a robot could indicate to a machine that it would like to have a door opened so that the robot could extract a part from the machine.

These data types are in the `EVENT` category and are modeled in the XML schema as part of an `Interface` type *Subcomponent*. However, they have functions and properties that differ from other data types in the category.

Many of the data types supporting each of these services are paired to describe two distinct actions – one to request the action to be performed and a second to reverse the action or to return to the original state. For example, a `DoorInterface` will have two actions `OPEN_DOOR` and `CLOSE_DOOR`. To enable the coordination between the two pieces of equipment, each data type **MUST** also specify a sub-type of `REQUEST` or `RESPONSE`. Data provided by the piece of equipment that requires a service to be performed will have the sub-type `REQUEST`. Data provided by the piece of equipment providing the service will have the sub-type `RESPONSE`. Together, the information provided by these data types form the basis for the coordination between the two pieces of equipment defined as the `Interface`.

The value provided in the CDATA for each DataItem type is constrained and **MUST** be either `UNAVAILABLE`, `READY`, `ACTIVE`, `NOT_READY`, or `FAIL`.

The following table provides the data types currently defined for the services supported by an Interface element:

DataItem type/subType	Description
MATERIAL_FEED	Service to load or feed material or product to a piece of equipment from a continuous or bulk source
MATERIAL_CHANGE	Service to request a change in the type of material or product being loaded or fed to a piece of equipment.
MATERIAL_RETRACT	Service to request that material or product be removed or retracted from a piece of equipment.
PART_CHANGE	Service to request that the type of part or product being made by a piece of equipment be changed to a different part or product type. Coupled with PART_ID to indicate the part or product type.
MATERIAL_LOAD	Service to request for a piece of material or product be loaded to a piece of equipment.
MATERIAL_UNLOAD	Service to request for a piece of material or product be unloaded from a piece of equipment.
OPEN_DOOR	Service to request another piece of equipment to open a door.
CLOSE_DOOR	Service to request another piece of equipment to close a door.
OPEN_CHUCK	Service to request another piece of equipment to open a chuck.
CLOSE_CHUCK	Service to request another piece of equipment to close a chuck.

7.3 DataItem Types for CONDITION Category

DataItem Types in the CONDITION category report data representing a Structural Element’s status or ability to operate. CONDITION is reported differently than SAMPLE or EVENT. CONDITION **MUST** be reported as NORMAL, WARNING, FAULT, or UNAVAILABLE.

All DataItem types in the SAMPLE category **MAY** have associated CONDITION states. These data items report continuously variable or analog data values. CONDITION states indicate whether the value reported for the data item is within an expected range (NORMAL) or the value is unexpected or out of tolerance for the data item (WARNING or FAULT).

Additionally, CONDITION **MAY** be further defined to indicate whether the reported value is above or below the expected range. These differences are defined by the qualifier attribute. As an example, CONDITION for an AMPERAGE type DataItem may differentiate between a HIGH amperage and a LOW amperage. See Part 3, Section 3.11 of the MTConnect Standard for more information on the qualifier attribute.

For these data items, there are five possible CONDITION states:

```
FAULT,LOW
WARNING,LOW
NORMAL
WARNING,HIGH
FAULT,HIGH
```

Some DataItem types in the EVENT category **MAY** have associated CONDITION states.

Additional CONDITION types are provided to represent the health and fault status of Structural Elements. Additionally, these CONDITION types are unlike other data item types since they **MAY** have multiple concurrently active values at any point in time. CONDITION states reported as WARNING or FAULT provide the information associated with the CONDITION state in the CDATA contained in the dataitem.

The table below defines these additional DataItem types that provide the health and fault status of Structural Elements.

DataItem type	Description
ACTUATOR	An actuator's status.
CHUCK_INTERLOCK	An indication of the operational condition of the interlock function for an electronically controller chuck.
COMMUNICATIONS	A communications failure indicator.
DATA_RANGE	Information provided is outside of expected value range
DIRECTION	An indication of a fault associated with the direction of motion of a Structural Element
END_OF_BAR	An indication that the end of a piece of bar stock has been reached.
HARDWARE	The hardware subsystem of the Structural Element's operation condition.
INTERFACE_STATE	An indication of the operation condition of an Interface.
LOGIC_PROGRAM	An error occurred in the logic program or PLC (programmable logic controller).
MOTION_PROGRAM	An error occurred in the motion program.
SYSTEM	A CONDITION representing something that is not the operator, program, or hardware.

8 Sensor

Sensor is a XML Element that has some unique properties from other element types. It can represent either a measurement device or the data providing the value of a measurement.

A sensor is typically comprised of two major components – the *sensing element* (provides a signal or measured value) and the *sensor unit* (signal processing, conversion, and communications). In MTConnect, the *sensor unit* is modeled as a Component or *Subcomponent* called Sensor. The *sensing element* or measured value is modeled as a DataItem (See *Section 7* of this document for more information on DataItem elements).

Example: A pressure transducer could be modeled as a Sensor (Component) with a name = *Pressure Transducer B* and its measured value could be modeled as a DataItem of type PRESSURE.

When modeled as a Component or *Subcomponent*, Sensor **MUST NOT** be modeled in the plural. Sensor will always refer to a single *sensor unit*. Multiple Sensor elements may be modeled in the XML document for a Device. Each *sensor unit* may have multiple *sensing elements*; each representing the data for a variety of measured values.

When modeled as a DataItem element, Sensor is an abstract type component that provides measurement data related to a Device, Component, or Subcomponent element. As such, the Sensor XML element will never appear in the XML document describing a specific measured value - only the different data types defined in Section 7 will appear in the XML document representing the specific type of measurement provided.

While Sensor may be modeled in the XML schema in different ways, it will always be modeled to associate the information contained in Sensor with the Structural XML Element to which the measurement device and the data provided by that device is most closely associated.

8.1 Sensor data

The most basic implementation of a *sensing element* is the providing of a measured value associated with a Component or *Subcomponent* which is the Sensor data. An example would be the measured value of the Temperature of the spindle (Rotary Axis C). This would be represented as a DataItem called Temperature that is associated with the Rotary Axis C as follows (See Section 7 for more information on data types):

```
<Components>
  <Axes
    <Components>
      <Rotary id="c" name="C">
        <DataItems>
          <DataItem type="TEMPERATURE" id="ctemp" category="SAMPLE"
                name="Stemp" units="DEGREE"/>
        </DataItems>
      </Rotary>
    </Components>
  </Axes>
</Components>
```

A sensor may measure values associated with any `Component`, *`Subcomponent`*, or `Device`. Some examples of how sensor data may be modeled are represented in Figure 9 below:

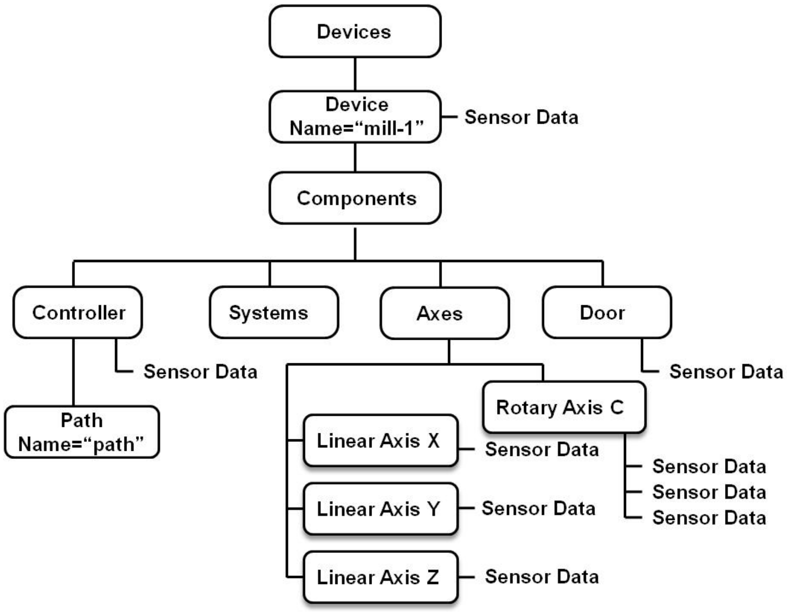

Figure 12: Sensor Data Associations

8.2 Sensor Unit

Sensing element(s) are most typically connected to a *sensor unit*. The *sensor unit* provides additional information concerning the *sensing element(s)*.

Typical functions of the *sensor unit* include:

- convert low level signals from the *sensing elements* into data that can be used by other devices. (Example: Convert a non-linear millivolt signal from a temperature sensor into a scaled temperature value that can be transmitted to another device.)

- process *sensing element* data into calculated values. (Example: temperature sensor data is converted into calculated values of average temperature, maximum temperature, minimum temperature, etc.)

- provide calibration and configuration information associated with each *sensing element*

- monitor the health and integrity of the *sensing elements* and the *sensor unit*. (Example: The *sensor unit* may provide diagnostics on each *sensing element* (e.g. open wire detection) and itself (e.g. measure internal temperature of the *sensor unit*).

The *sensor unit* is modeled in the XML schema as a `Component` called `Sensor.` `Sensor` **SHOULD** be modeled in the XML schema so that the `Sensor` is represented as part of the `Component` to which it is most closely associated.

Sensor, when representing a *senor unit*, may be associated with any Component, Subcomponent, or Device. Some examples of where a *sensor unit* may be modeled are represented in Figure 10 below:

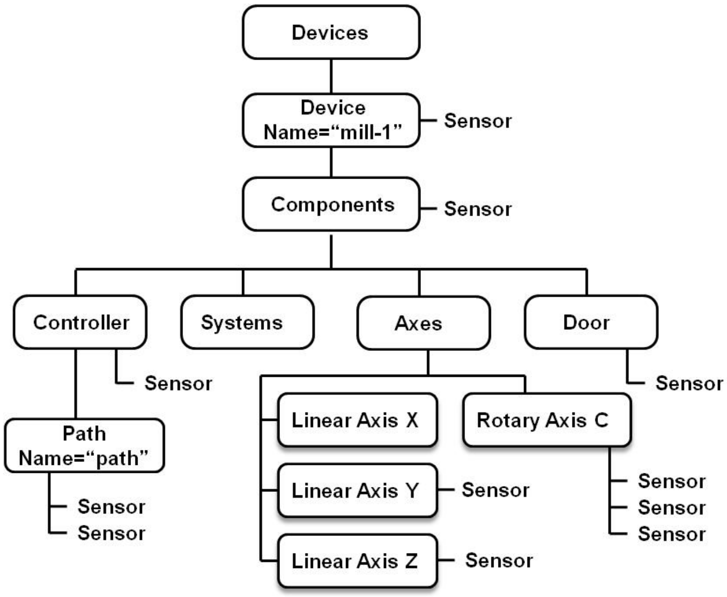

Figure 6: Sensor Associations

When a Sensor is modeled as a Component, it **MAY** have its own uuid so it can be tracked throughout its lifetime.

The following examples demonstrate how Sensor may be modeled in the XML schema differently based on how the sensor functions within the overall Device.

Example#1: If Sensor provides vibration measurement data for the spindle, it should be modeled as a Sensor for Rotary Axis C.

```
<Components>
  <Axes>
    <Components>
      <Rotary id="c" name="C">
        <Sensor id="spdlm" name="Spindlemonitor">
           <DataItems>
              <DataItem type="DISPLACEMENT" id="cvib" category="SAMPLE"
                 name="Svib" units="MILLIMETER"/>
           </DataItems>
        </Sensor>
      </Rotary>
    </Components>
  </Axes>
</Components>
```

Example#2: If Sensor provides measurement data for multiple Components within a Device and is not associated with any particular Component, it MAY be modeled in the XML schema as an independent Component of the Device.

```
 <Device id="d1" uuid="HM1" name="HMC_3Axis">
   <Description>3 Axis Mill</Description>
   <Components>
     <Sensor id="sensor" name="sensor"/>
        <DataItems>
           <DataItem type="TEMPERATURE" id="sentemp" category="SAMPLE"
                 name="Sensortemp" units="DEGREE"/>
        </DataItems>
  </Components>
</Device>
```

While Sensor MAY be modeled in different ways in the XML schema, the measured value of the *sensing element* **MUST** always be modeled as a DataItem associated with the Component to which the measured value is most closely associated.

Example#3: In this case, Sensor is modeled as a Component within a Device. Its measured values from the *sensing elements* are associated with other Components in the Device. The sensor also has internal diagnostics capabilities representing the CONDITION of the sensor itself.

The following represents a sensor with two *sensing elements*, one measures spindle vibration and the other measures the temperature for the X axis.   The sensor also has a *sensing element* measuring the internal temperature of the *sensor unit*.

```
<Device id="d1" uuid="HM1" name="HMC_3Axis">
  <Description>3 Axis Mill</Description>
  <Components>
    <Sensor id="sens1" name="Sensorunit">
      <DataItems>
        <DataItem type="TEMPERATURE" id="sentemp" category="SAMPLE"
              name="Sensortemp" units="DEGREE"/>
      </DataItems>
    </Sensor>
    <Axes>
      <Components>
        <Rotary id="c" name="C">
          <DataItems>
            <DataItem type="DISPLACEMENT" id="cvib" category="SAMPLE"
                name="Svib" units="MILLIMETER"/>
          </DataItems>
        </Rotary>
        <Linear id="x" name="X">
          <DataItems>
            <DataItem type="TEMPERATURE" id="xt"
             category="SAMPLE" name="Xtemp" units="DEGREE"/>
          </DataItems>
        </Linear>
      </Components>
    </Axes>
  </Components>
</Device>
```

8.3 Sensor as a `Device`

A `sensor` may function as an independent device. In this case, it is not associated with a parent `Device` or `Component`.

Examples of a sensor functioning as a `Device` would be a sensor used to monitor the ambient temperature of a building or an air quality monitoring system.   Another example would be a vibration monitoring system that is moved from one machine to another.   In these cases, the sensor functions as an intelligent device performing a specific function.

A sensor functioning as a `Device` would be modeled in the XML schema as follows:

```
<Device id="s1" uuid="HM1" name="AMBIENT_MONITOR">
  <Description>Ambient Temperature Monitor</Description>
  <DataItems>
    <DataItem type="TEMPERATURE" id="ambtemp" category="SAMPLE"
        name="Ambienttemp" units="DEGREE"/>
  </DataItems>
</Device>
```

A sensor that is modeled as a device **MUST** have an `uuid` so that it can be uniquely tracked.

8.4 Sensor Configuration

When a sensor is modeled in the XML schema as a Component or a Device, it may provide additional configuration information for the *sensor elements* and the *sensor unit* itself.

The Sensor configuration data provides information required for maintenance and support of the sensor.

Sensor configuration data is *only* available when the sensor is modeled as a Component or a Device. For details on the modeling of Configuration data in the XML schema, see *Part 2, Section 3.4.7.1 Component Configuration*. Details specific to SensorConfigurationType are provided below.

When Sensor represents the *sensor unit* for multiple *sensing element(s)*, each *sensing element* is represented by a Channel. Each Channel represents one *sensing element* and can have its own attributes and Configuration data.

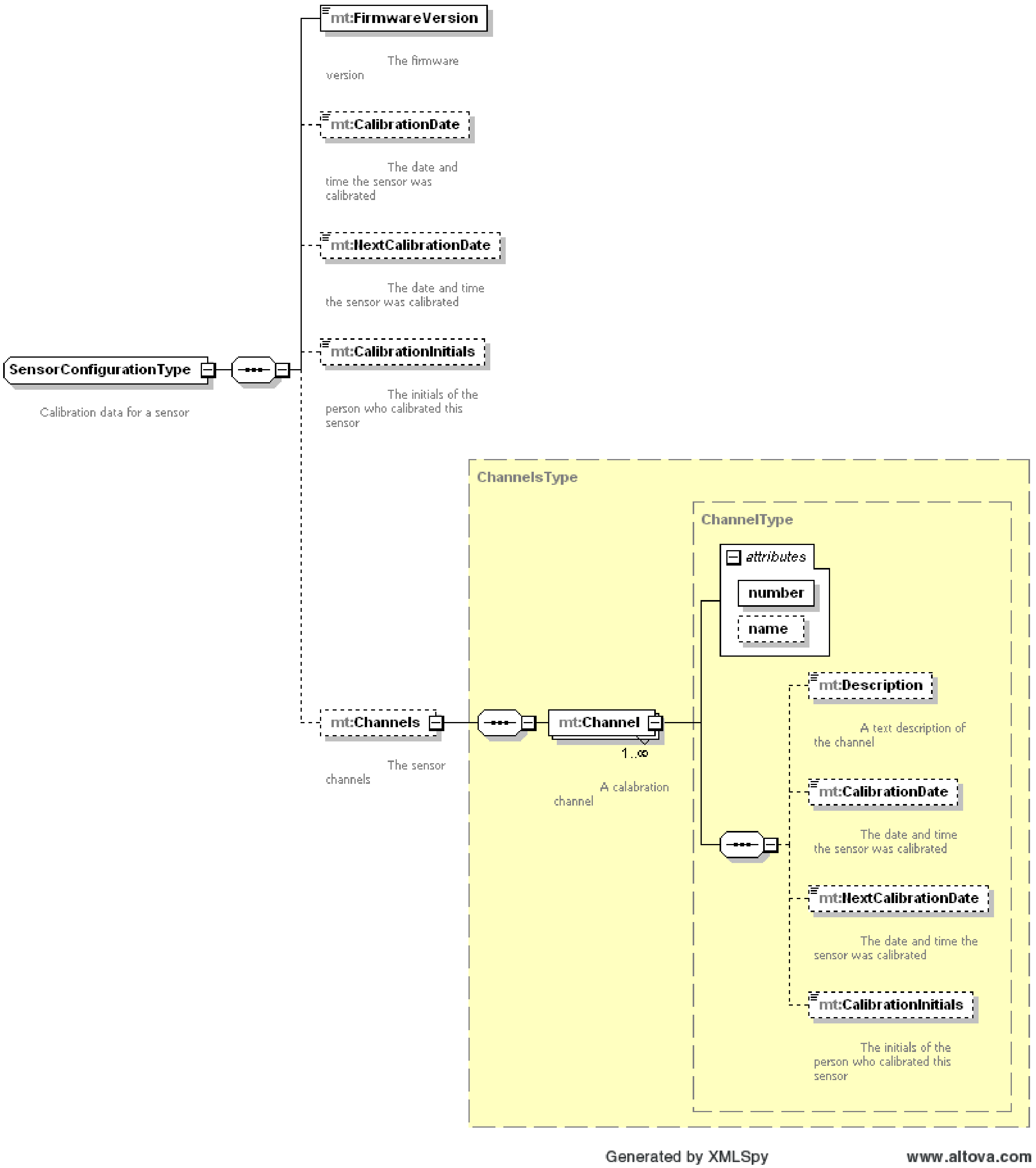
mt:FirmwareVersion
The firmware version
mt:CalibrationDate
The date and time the sensor was calibrated
mt:NextCalibrationDate
The date and time the sensor was calibrated
mt:CalibrationInitials
The initials of the person who calibrated this sensor
SensorConfigurationType
Calibration data for a sensor
ChannelsType
ChannelType
attributes
number
name
mt:Description
A text description of the channel
mt:CalibrationDate
The date and time the sensor was calibrated
mt:NextCalibrationDate
The date and time the sensor was calibrated
mt:CalibrationInitials
The initials of the person who calibrated this sensor
mt:Channels
The sensor channels
mt:Channel
1..∞
A calabration channel
Generated by XMLSpy
www.altova.com

Figure 14: `Configuration` Data for Sensors

Element	Description	Occurrence

`Configuration`
`(SensorConfigurationType)`
	An element that can contain descriptive content defining the configuration information for Sensor.
	For Sensor, the valid configuration is `SensorConfiguration`. `SensorConfiguration` provides data from a subset of items commonly found in a transducer electronic data sheet for sensors and actuators called TEDS.
	TEDS formats are defined in IEEE 1451.0 and 1451.4 transducer interface standards (ref 15 and 16, respectively).
	MTConnect does not support all of the data represented in the TEDS data, nor does it duplicate the function of the TEDS data sheets.
		0..1

8.4.1 `SensorConfiguration` Elements

The following table defines the configuration attributes available for `SensorConfiguration`:

Element	Description	Occurrence
`FirmwareVersion`	Version number for the sensor as specified by the manufacturer.	1
`CalibrationDate`	Date upon which the sensor was last calibrated. Dates **MUST** be represented in the W3C ISO 8601 format	0..1
`NextCalibrationDate`	Date upon which the sensor is next scheduled to be calibrated. Dates **MUST** be represented in the W3C ISO 8601 format	0..1
`CalibrationInitials`	The initials of the person verifying the validity of the calibration data	0..1
`Channels`	When `Sensor` represents multiple *sensing elements*, each *sensing element* is represented by a `Channel` for the `Sensor`.	0..1

8.4.1.1 Sensor Channel Attributes

`Channel` represents each *sensing element* connected to a *sensor unit*. Each `Sensor Channel` has the following composition:

Attribute	Description	Occurrence
`Number`	A unique identifier that will only refer to this *sensing element*. For example, this can be the manufacturer code and the serial number. The `Number` should be alphanumeric and not exceeding 255 characters. An NMTOKEN XML type.	1
`Name`	The `Name` of the *sensing element*. This name should be unique within the machine to allow for easier data integration. An NMTOKEN XML type.	0..1

8.4.1.2 Sensor Channel Elements

Element	Description	Occurrence
`Description`	An XML element that can contain any descriptive content. This can contain information about the *sensor element* and manufacturer specific details.	0..1
`CalibrationDate`	Date upon which the *sensor element* was last calibrated. Dates **MUST** be represented in the W3C ISO 8601 format	0..1
`NextCalibrationDate`	Date upon which the *sensor element* is next scheduled to be calibrated. Dates **MUST** be represented in the W3C ISO 8601 format	0..1
`CalibrationInitials`	The initials of the person verifying the validity of the calibration data	0..1

The following is an example of the configuration data for `Sensor` that is modeled as a `Component`. It has `Configuration` data for the *sensor unit*, one `Channel` named A/D:1, and two `DataItems` – `Voltage` (as a `SAMPLE`) and `Voltage` (as a `CONDITION` or alarm).

```
<Sensor id="sensor" name="sensor">
  <Configuration>
    <SensorConfiguration>
      <FirmwareVersion>2.02</FirmwareVersion>
      <CalibrationDate>2010-05-16</CalibrationDate>
      <NextCalibrationDate>2010-05-16</NextCalibrationDate>
      <CalibrationInitials>WS</CalibrationInitials>
      <Channels>
        <Channel number="1" name="A/D:1">
         <Description>A/D With Thermister</Description>
        </Channel>
      </Channels>
    </SensorConfiguration>
  </Configuration>
  <DataItems>
    <DataItem category="CONDITION" id="senvc" type="VOLTAGE" />
    <DataItem category="SAMPLE" id="senv" type="VOLTAGE" units="VOLT"
                 subType="DIRECT" />
  </DataItems>
</Sensor>
```

8.5 Sensor Data Types

When modeled as a `DataItem` element, `Sensor` will be represented in the XML document as one of the `DataItem` types defined in *Section 7* of this document. Most `Sensor` data types will be represented by `DataItem` types in the `SAMPLE` category since they typically represent the value of a continually varying measured variable (temperatures, pressures, positions, etc). However, some `Sensor` elements detect discrete events and are represented by `DataItem` types in the `EVENT` category; `Direction` would be an example of such a data type.

Appendices

A. 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 B5.57: *Methods for Performance Evaluation of Computer Numerically Controlled Lathes and Turning Centers*, 1998

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

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

15. IEEE STD 1451.0-2007*, Standard for a Smart Transducer Interface for Sensors and Actuators – Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH99684, October 5, 2007.*

16. IEEE STD 1451.4-1994, Standard for a Smart Transducer Interface for Sensors and Actuators – Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH95225, December 15, 2004.