MTConnect® Standard
Part 4.0 – Assets Information Model

Version 1.4.0

Prepared for: MTConnect Institute

Prepared on: March 31, 2018

MTConnect® Specification and Materials

Table of Content

1 Purpose of This Document 1

2 Terminology and Conventions 2

3 MTConnect Assets 3

3.1 Overview 3
3.2 `MTConnectAssets` 4
3.2.1 *`MTConnectAssets` Header 5*
3.2.1.1 `Header` Attributes 5
3.2.2 *`Assets` 7*
3.2.3 *`Asset` 7*
3.2.3.1 Common `Asset` Attributes 8
3.2.3.2 Common `Asset` Elements 10

4 MTConnect Assets Architecture 11

4.1 MTConnect Agent Asset Storage 11
4.2 *Asset* Protocol 12
4.2.1 *Asset by `assetId` 12*
4.2.2 *Asset for a Given Type 12*
4.2.3 *Assets Including `Removed` Assets 13*
4.2.4 *Assets for a Piece of Equipment 13*

5 Extensions to *Part 2.0 – Devices Information Model* 14

5.1 Data Item Types added for `EVENT` Category 14
5.1.1 *`ASSET_CHANGED` Data Item Type 14*
5.1.2 *`ASSET_REMOVED` Data Item Type 14*

6 Extensions to *Part 3.0 – Streams Information Model* 16

6.1 `AssetChanged` Extension to `Events` 16
6.1.1 *`AssetChanged` Attributes: 16*
6.2 `AssetRemoved` Extension to `Events` 17
6.2.1 *`AssetRemoved` Attributes: 17*

Appendices 18

A. Bibliography 18

Table of Figures

Figure 1: `MTConnectAssets` Schema 4

Figure 2: `Header` Schema Diagram for `MTConnectAssets` 5

Figure 3: `Asset` Schema 8

Figure 4: `Description` Schema 10

Figure 5: `AssetChanged` Schema 16

1 Purpose of This Document

This document, *Part 4.0 – Assets Information Model* of the MTConnect Standard, details information that is common to all types of *MTConnect Assets*. *Part 4.0* and its sub-parts of the MTConnect Standard provide semantic models for entities that are used in the manufacturing process, but are not considered to be a piece of equipment. These entities are defined as *MTConnect Assets*. These *Assets* may be removed from a piece of equipment without detriment to the function of the equipment and can be associated with other pieces of equipment during their lifecycle. The data associated with these *Assets* may be retrieved from multiple sources that are each responsible for providing their knowledge of the *Asset*.

2 Terminology and Conventions

Please refer to *Part 1.0 - Overview and Fundamentals, Section 2* for a dictionary of terms, reserved language, and document conventions used in the MTConnect Standard.

3 MTConnect Assets

3.1 Overview

The MTConnect Standard supports a simple distributed storage mechanism that allows applications and equipment to share and exchange complex information models in a similar way to a distributed data store. The *Asset Information Model* associates each electronic *MTConnectAssets* document with a unique identifier and allows for some predefined mechanisms to find, create, request, updated, and delete these electronic documents in a way that provides for consistency across multiple pieces of equipment.

The protocol provides a limited mechanism of accessing *MTConnect Assets* using the following properties: `assetId`, *Asset* type (element name of *Asset* root), and the piece of equipment associated with the *Asset*. These access strategies will provide the following services and answer the following questions: What *Assets* are from a particular piece of equipment? What are the *Assets* of a particular type? What *Assets* is stored for a given `assetId`?

Although these mechanisms are provided, an *MTConnect Agent* should not be considered a data store or a system of reference. The *Agent* is providing an ephemeral storage capability that will temporarily manage the data for applications wishing to communicate and manage data as needed by the various processes. An application cannot rely on an *Agent* for long term persistence or durability since the *Agent* is only required to temporarily store the *Asset* data and may require another system to provide the source data upon initialization. An *MTConnect Agent* is always providing the best-known equipment centric view of the data given the limitations of that piece of equipment.

Note: Currently only cutting tools have been addressed by the MTConnect Standard and other *MTConnect Assets* will be defined in later versions of the Standard.

3.2 MTConnectAssets

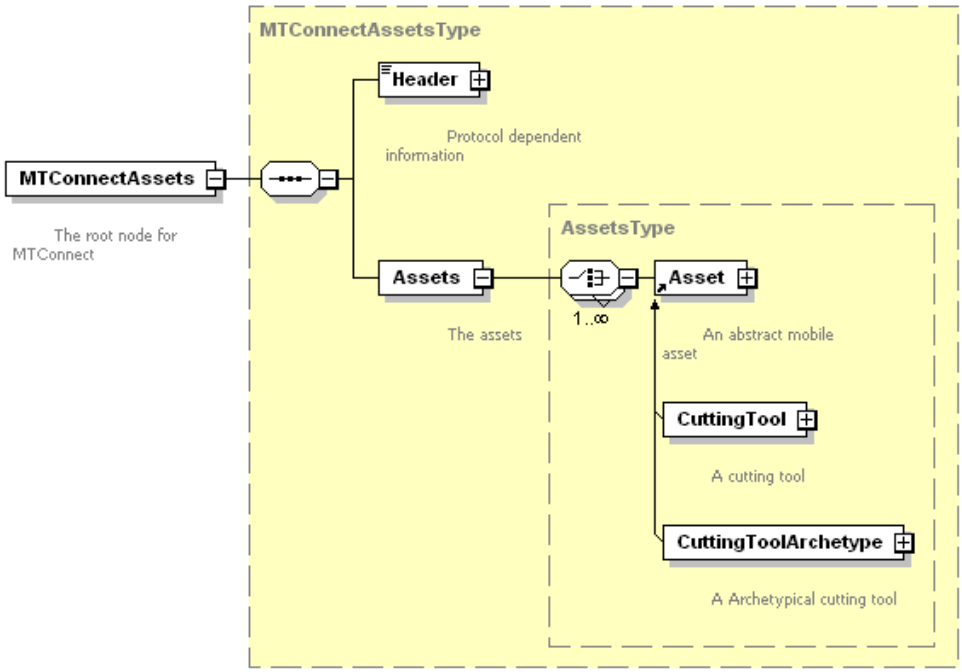


Figure 1: `MTConnectAssets` Schema

At the top level of the `MTConnectAssets` document is a standard header, as stated in *Part 1.0 - Overview and Fundamentals*, and one or more *MTConnect Assets*. Each *Asset* is required to have an `assetId` that serves as a unique identifier of that *Asset*. `assetId` allows an application to request the *Asset* data from an *MTConnect Agent*.

In the remaining *Part 4.x* sub-part documents of *MTConnect Assets*, various types of *Assets* will be introduced such as cutting tools and other *Asset* types. Currently only cutting tools have been defined in *Part 4.1 – Cutting Tools*.

3.2.1 `MTConnectAssets` Header

The `MTConnectAssets` header is where the protocol sequence information **MUST** be provided. The following XML schema represents the structure of the `MTConnectAssets` header showing the attributes defined for `MTConnectAssets`.

Refer to *Part 1.0 – Overview and Fundamentals* for more information on headers.

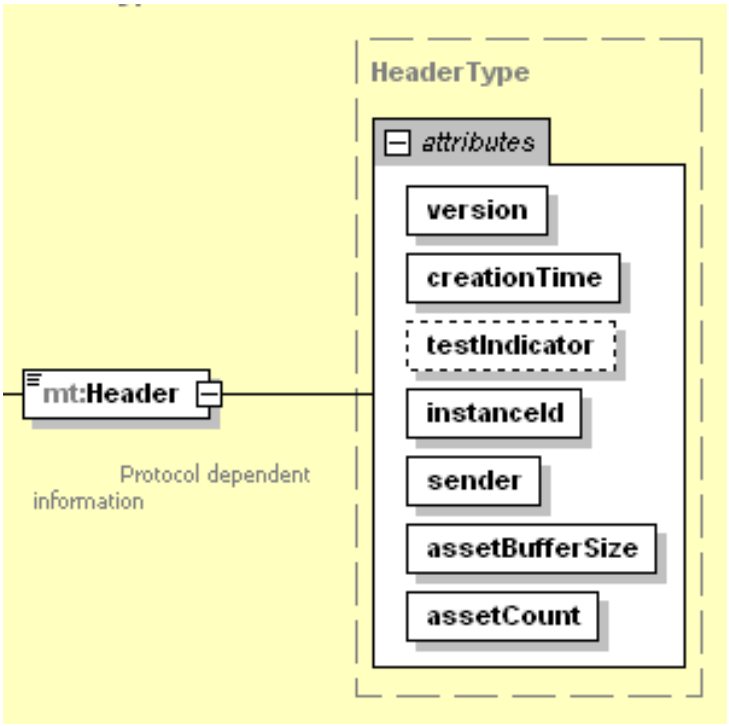


Figure 2: `Header` Schema Diagram for `MTConnectAssets`

3.2.1.1 `Header` Attributes

The following table defines the attributes used to provide information for an `MTConnectAssets` header.

Attribute	Description	Occurrence
`version`	The protocol version number. This is the *major* and *minor* version number of the MTConnect Standard being used. For example, if the version number of the Standard used is `10.21.33`, the `version` will be `10.21`. `version` is a required attribute.	1

Attribute	Description	Occurrence
`creationTime`	The time the response was created. `creationTime` is a required attribute.	1
`testIndicator`	Optional flag that indicates the system is operating in test mode. This data is only for testing and indicates that the data is simulated. `testIndicator` is an optional attribute.	0..1
`instanceId`	A number indicating which invocation of the *MTConnect Agent*. This is used to differentiate between separate instances of the *Agent*. This value **MUST** have a maximum value of 2^64-1 and **MUST** be stored in an unsigned 64-bit integer. `instanceId` is a required attribute.	1
`sender`	The *MTConnect Agent* identification information. `sender` is a required attribute.	1
`assetBufferSize`	The maximum number of *MTConnect Assets* that will be retained by the *MTConnect Agent*. The `assetBufferSize` **MUST** be an unsigned positive integer value with a maximum value of 2^32-1. `assetBufferSize` is a required attribute.	1
`assetCount`	The total number of *MTConnect Assets* in an *MTConnect Agent*. This **MUST** be an unsigned positive integer value with a maximum value of 2^32-1. This value **MUST NOT** be greater than `assetBufferSize` `assetCount` is a required attribute.	1

Example:

```
1.  <Header creationTime="2010-03-13T07:59:11+00:00" sender="localhost"
2.    instanceId="1268463594" assetBufferSize="1024" version="1.1"
3.    assetCount="12" />
```

3.2.2 Assets

Assets is an XML container used to group information about various *MTConnect Asset* types. Assets contains one or more Asset XML elements.

Element	Description	Occurrence
Assets	XML container that consists of one or more types of Asset XML elements.	0..1

3.2.3 Asset

An Asset XML element is a container type XML element used to organize information describing an entity that is not a piece of equipment. Asset is an abstract type XML element and will never appear directly in the MTConnect XML document. As an abstract type XML element, Asset will be replaced in the XML document by specific *MTConnect Asset* type.

Element	Description	Occurrence
Asset	An abstract XML element. Replaced in the XML document by types of Asset elements representing entities that are not pieces of equipment. There can be multiple types of Asset XML elements in the document.	1..INF

There are various types of entities or *Asset* types. Each type of *Asset* is described in sub-parts of *Part 4.0 – Assets Information Model*. These sub-parts are designated by a *Part 4.x* document number. Currently only the *MTConnect Asset* type of cutting tools has been defined in *Part 4.1 – Cutting Tools*.

For all *MTConnect Asset* types there are some common attributes and elements that apply to all of them. The following defines these common attributes and elements.

3.2.3.1 Common `Asset` Attributes

The following XML schema represents the structure of `Asset` showing the attributes defined for `Asset`.

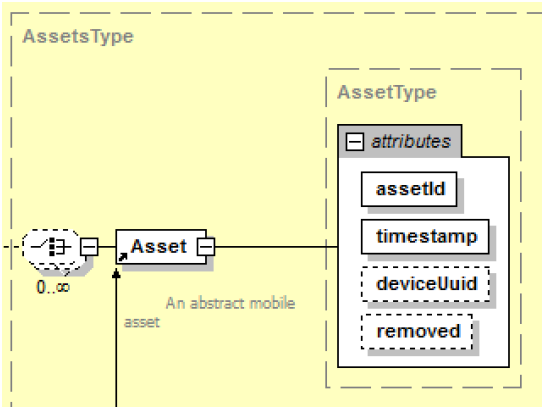


Figure 3: `Asset` Schema

The following table defines the attributes that are used to provide information for the `Asset` element.

Attribute	Description	Occurrence
`assetId`	The unique identifier for the *MTConnect Asset*. The identifier **MUST** be unique with respect to all other *Assets* in an MTConnect installation. The identifier **SHOULD** be globally unique with respect to all other *Assets*. `assetId` is a required attribute.	1
`timestamp`	The time this *MTConnect Asset* was last modified. Always given in UTC. The timestamp **MUST** be provided in UTC (Universal Time Coordinate, also known as GMT). This is the time the *Asset* data was last modified. `timestamp` is a required attribute.	1
`deviceUuid`	The piece of equipment's UUID that supplied this data. This is an optional element references to the `uuid` attribute given in the `Device` element. This can be any series of numbers and letters as defined by the XML type NMTOKEN.	0..1

Attribute	Description	Occurrence
removed	This is an optional attribute that is an indicator that the *MTConnect Asset* has been removed from the piece of equipment. If the *Asset* is marked as removed, it will not be visible to the client application unless the includeRemoved=true parameter is provided in the URL. If this attribute is not present it **MUST** be assumed to be false. The value is an xsi:boolean type and **MUST** be true or false.	0..1

All *MTConnect Assets* **MUST** have an assetId that differs from all the other *Assets* in a facility and preferably globally unique, such as a RFC 4122 UUID. There **MUST** never be more than one *Asset* provided by an *Agent* with the same assetId in the same shop.

The following attributes **MUST** be provided and are common to all *MTConnect Asset* types: the assetId attribute providing the unique identifier for the *Asset*, and the timestamp providing the time the *Asset* was inserted or updated. A removed flag that if true indicates the *Asset* has been removed (deleted) from the equipment is optional, however the *Asset* will still be available if requested directly or a request is made that includes removed *Assets*.

An MTConnectAssets document contains information pertaining to something that is not a direct component of the piece of equipment and can be relocated to another piece of equipment or location during its lifecycle. The Asset will contain data that will be changed as a unit, meaning that at any given point in time the latest version of the complete state for this *Asset* will be provided.

Each piece of equipment or location may have a different view of this *Asset* and it is the responsibility of an application to collect and determine the aggregate information and keep a historical record if required. An *MTConnect Agent* will allow any application or other equipment to request this information. The piece of equipment **MUST** supply the latest and most accurate information regarding a given *Asset*.

3.2.3.2 Common `Asset` Elements

The element `Description` is the only element common to all *Asset* types.

The following XML schema represents the structure of `Description`.

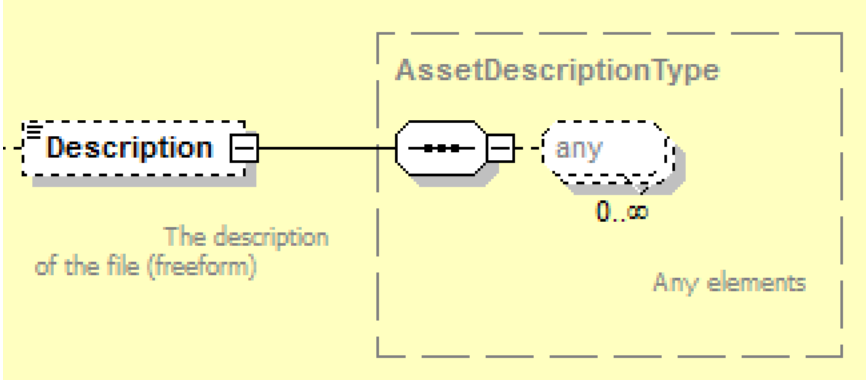


Figure 4: `Description` Schema

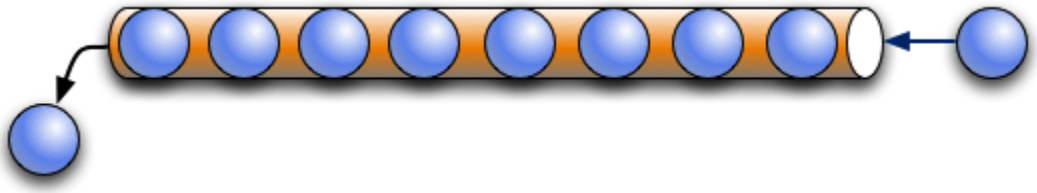
The following table defines the elements that are used to provide information for `Asset`.

Element	Description	Occurrence
`Description`	An optional element that can contain any descriptive content. This can contain configuration information and manufacturer specific details. This element is defined to contain mixed content and XML elements can be added to extend the descriptive semantics of MTConnect Standard.	0..1

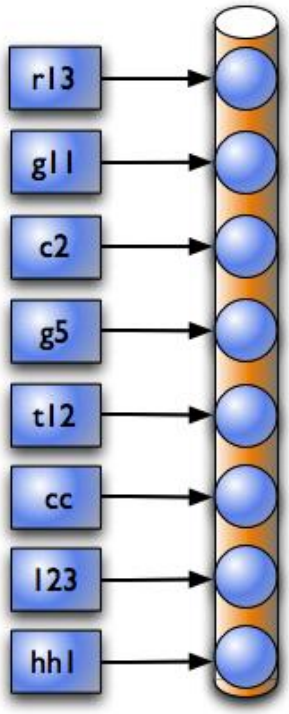
4 MTConnect Assets Architecture

4.1 MTConnect Agent Asset Storage

The *MTConnect Agent* stores *MTConnect Assets* in a similar fashion as the *Agent* data storage described in *Part 1.0 – Overview and Fundamentals*. The storage of information is contained in the *asset buffer*. The *MTConnect Agent* provides a limited number of *Assets* that can be stored at one time and uses the same method of pushing out the oldest *Asset* when the *asset buffer* is full. The *asset buffer* size for the *Asset* storage is maintained separately from the `Sample`, `Event`, and `Condition` storage.



MTConnect Assets also behave like a key/value in memory database. In the case of the *Asset,* the key is the `assetId` and the value is the XML document describing the *Asset*. The key can be any string of letters, punctuation or digits and represent the domain specific coding scheme for their assets. Each *Asset* type will have a recommended way to construct a unique `assetId`, for example, a cutting tool **SHOULD** be identified by the tool ID and serial number as a composed synthetic identifier.



As in this example above, each of the *Assets* is referred to by their key. The key is independent of the order in the *asset buffer* storage.

4.2 *Asset* Protocol

MTConnect Standard provides methods to retrieve an *MTConnect Asset* or a set of *Assets* given various criteria. These criteria are as follows: The `assetId`, the *Asset* `type` as defined by the name of the *Asset's* topmost element, and the originating piece of equipment.

The URL format is similar to the `Probe` and `Sample` structure. For example, to request an *MTConnect Asset* by `assetId`, reference each `assetId` directly as follows:

4.2.1 *Asset* by `assetId`

```
1.  url: http://example.com/asset/e39d23ba-ef2d-11e6-b12c-
2.    28cfe91a82ef
```

Returns the `MTConnectAssets` document for *Asset* e39d23ba-ef2d-11e6-b12c-28cfe91a82ef

Request multiple *Assets* by each `assetId`:

```
1.  url: http://example.com/asset/e39d23ba-ef2d-11e6-b12c-
2.    ;8cfe91a82ef;e46d5256-ef2d-11e6-96aa-28cfe91a82ef
```

Returns the `MTConnectAssets` document for *Assets* e39d23ba-ef2d-11e6-b12c-28cfe91a82ef and e46d5256-ef2d-11e6-96aa-28cfe91a82ef.

Request for all the *Assets* in the *MTConnect Agent*:

```
1.  url: http://example.com/assets
```

Returns all available *MTConnect Assets* in the *MTConnect Agent*. The *Agent* **MAY** return a limited set if there are too many *Asset* records. The *Assets* **MUST** be added to the beginning with the most recently modified *Asset*.

4.2.2 *Asset* for a Given Type

```
1.  url: http://example.com/assets?type="CuttingTool"
```

Returns all available `CuttingTool` *Assets* from the *MTConnect Agent* of the type `CuttingTool`. The *Agent* **MAY** return a limited set if there are too many *Asset* records. The *Assets* **MUST** be added to the beginning with the most recently modified assets.

Request for all *Assets* of a given type in the *MTConnect Agent* up to a maximum count:

```
1.  url: http://example.com/assets?type=CuttingTool&count=1000
```

Returns all available `CuttingTool` *Assets* from the *MTConnect Agent*. The *Agent* **MUST** return up to 1000 *Assets* beginning with the most recently modified *Assets* if they exist.

4.2.3 *Assets* Including Removed *Assets*

```
1.  url: http://example.com/assets?type=CuttingTool&removed=true
```

Returns all available `CuttingTool` *Assets* from the *MTConnect Agent*. With the removed flag, *Assets* that have been removed but are included in the result set.

4.2.4 *Assets* for a Piece of Equipment

If no `assetId` is provided with a general *Assets* request, it would be as follows:

```
1.  url: http://example.com/Mill123/assets
```

All *MTConnect Assets* will be provided for that piece of equipment (`Device`) up to the *MTConnect Agent's* maximum count or as specified with the `count` parameter. These *Assets* will be returned starting from the newest to oldest list.

Any of the previous constraints can also be applied to the request, for example, to get all the *CuttingTool* instances for a given piece of equipment:

```
1.  url: http://example.com/Mill123/asset/?type=CuttingTool&count=100
```

The previous request will get the newest 100 Cutting Tool Instance *Assets* from the *MTConnect Agent* for `Mill123`. Similarly:

```
1.  url: http://example.com/Mill123/asset/?type=CuttingToolArchetype
```

Will provide all Cutting Tool Archetype *Assets* with the `deviceUuid` of `Mill123`.

5 Extensions to *Part 2.0 – Devices Information Model*

This document will add the following data item types to support change notification when an *MTConnect Asset* is added or updated. The data item **MUST** be placed in the `DataItems` container associated with `Device`. The `Device` **MUST** be the piece of equipment that is supplying the asset data.

5.1 Data Item Types added for `EVENT` Category

Data Item type/subtype	Description
`ASSET_CHANGED`	The value of the **CDATA** for the event **MUST** be the `assetId` of the asset that has been added or changed. There will not be a separate message for new assets.
`ASSET_REMOVED`	The value of the **CDATA** for the event **MUST** be the `assetId` of the asset that has been removed. The asset will still be visible if requested with the `includeRemoved` parameter as described in the protocol section. When assets are removed they are not moved to the beginning of the most recently modified list.

5.1.1 `ASSET_CHANGED` Data Item Type

When an *MTConnect Asset* is added or modified, an `AssetChanged` event **MUST** be published to inform an application that new asset data is available. The application can request the new asset data from the piece of equipment at that time. Every time the asset data is modified an `AssetChanged` event will be published. Since the asset data is a complete electronic document, the system will publish a single `AssetChanged` event for the entire set of changes.

The asset data **MUST** remain constant until the `AssetChanged` event is published. Once it is published the data **MUST** change to reflect the new content at that instant. The timestamp of the asset will reflect the time the last change was made to the asset data.

5.1.2 `ASSET_REMOVED` Data Item Type

When an *MTConnect Asset* has been removed from an *MTConnect Agent*, or marked as removed, an `AssetRemoved` event **MUST** be generated in a similar way to the `AssetChanged` event. The CDATA of the `AssetRemoved` event **MUST** contain the `assetId` that was just removed.

Every time an *MTConnect Asset* is modified or added it will be moved to the beginning of the *asset buffer* and become the newest *Asset*. As the *asset buffer* fills up, the oldest *Asset* will be pushed out and its information will be removed. The MTConnect Standard does not specify the maximum size of the *asset buffer*, and if the implementation desires, permanent storage **MAY** be used to store the *Assets*. A value of `4,294,967,296` or 2^{32} can be given to indicate unlimited storage.

There is no requirement for persistent *Asset* storage. If the *MTConnect Agent* fails, all existing *MTConnect Assets* **MAY** be lost. It is the responsibility of the implementation to restore the lost *Asset* data and it is the responsibility of the application to persist the *Asset* data. The *MTConnect Agent* **MAY** make no guarantees about availability of *Asset* data after the *Agent* stops.

6 Extensions to *Part 3.0 – Streams Information Model*

The associated modifications **MUST** be added to *Part 3.0 – Streams Information Model* to add the following event to the `Events` in the streams.

6.1 `AssetChanged` Extension to `Events`

The `AssetChanged` element extends the base `Event` type XML data element defined in *Part 3.0 – Streams Information Model* and adds the `assetType` attribute to the base `Event`. This new `Event` will signal whenever a new *MTConnect Asset* is added or the existing definition of an *Asset* is updated. The `assetId` is provided as the CDATA value and can be used to request the *Asset* data from the *MTConnect Agent*.

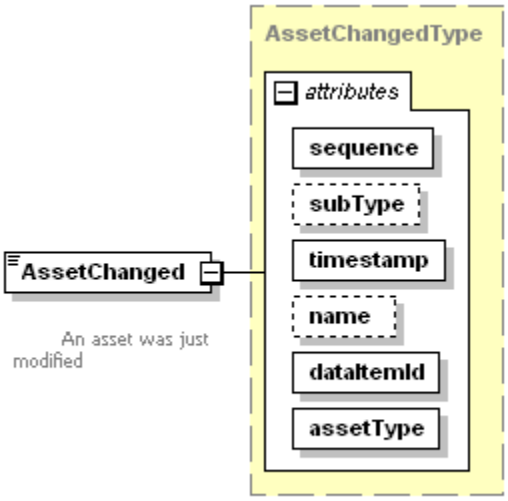


Figure 5: `AssetChanged` Schema

`AssetChanged` An *MTConnect Asset* has been added or modified. The **CDATA** for the `AssetChanged` element **MUST** be the `assetId` of the *Asset* that has been modified.

6.1.1 `AssetChanged` Attributes:

Attribute	Description	Occurrence
`assetType`	The type of asset that changed. `assetType` is a required attribute. Valid Data Values: -Cutting Tool	1

6.2 AssetRemoved Extension to Events

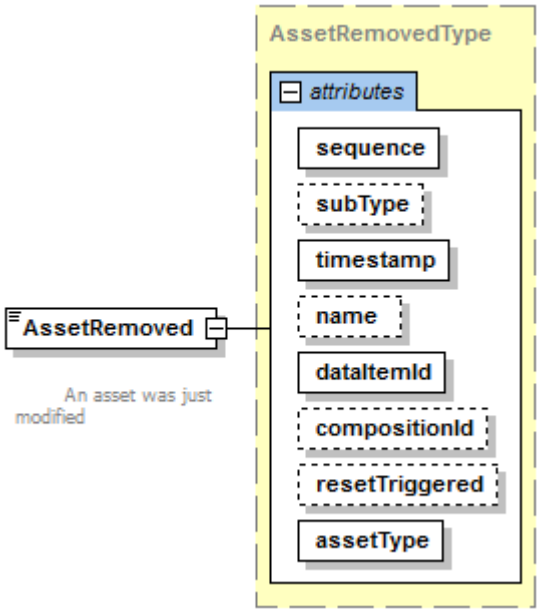


Figure 6: AssetRemoved Schema

AssetRemoved An *MTConnect Asset* has been removed. The **CDATA** for the AssetRemoved element **MUST** be the assetId of the *Asset* that has been removed.

6.2.1 AssetRemoved Attributes:

Attribute	Description	Occurrence
assetType	The type of asset that was removed. assetType is a required attribute. Valid Data Values: -Cutting Tool	1

The *MTConnect Asset* will still be available if requested if the removed=true argument is supplied. The assetId is provide as the CDATA value and can be used to request the *Asset* data from the *MTConnect Agent*.

Appendices

A. Bibliography

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.

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.

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.

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.

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.

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.

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

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.

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.

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

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

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

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

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

International Organization for Standardization. *ISO 13399*: *Cutting tool data representation and exchange*. Geneva, Switzerland, 2000.