MTConnect® Standard
Part 5 – Interfaces
Version 1.7.0

Prepared for: MTConnect Institute
Prepared on: February 25, 2021

MTConnect® is a registered trademark of AMT - The Association for Manufacturing Technology. Use of MTConnect is limited to use as specified on [http://www.mtconnect.org/](http://www.mtconnect.org/)
The Association for Manufacturing Technology (AMT) owns the copyright in this MTConnect Specification or Material. AMT grants to you a non-exclusive, non-transferable, revocable, non-sublicensable, fully-paid-up copyright license to reproduce, copy and redistribute this MTConnect Specification or Material, provided that you may only copy or redistribute the MTConnect Specification or Material in the form in which you received it, without modifications, and with all copyright notices and other notices and disclaimers contained in the MTConnect Specification or Material.

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 mailto:info@MTConnect.org.

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 noninfringement, 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

1 Purpose of This Document 2

2 Terminology and Conventions 3
   2.1 Glossary ......................................................... 3
   2.2 Acronyms ......................................................... 8
   2.3 MTConnect References ........................................... 8

3 Interfaces Overview 9
   3.1 Interfaces Architecture ........................................... 9
   3.2 Request and Response Information Exchange ..................... 11

4 Interfaces for Devices and Streams Information Models 14
   4.1 Interfaces ....................................................... 15
   4.2 Interface ......................................................... 15
      4.2.1 XML Schema Structure for Interface ..................... 15
      4.2.2 Interface Types ............................................ 17
      4.2.3 Data for Interface ........................................... 19
         4.2.3.1 References for Interface ............................ 19
      4.2.4 Data Items for Interface ................................... 20
         4.2.4.1 INTERFACE_STATE for Interface .................. 20
         4.2.4.2 Specific Data Items for the Interaction Model for Interface 21
         4.2.4.3 Event States for Interfaces .......................... 23

5 Operation and Error Recovery 28
   5.1 Request/Response Failure Handling and Recovery ............... 28

Appendices 36
   A Bibliography ..................................................... 36
Table of Figures

Figure 1: Data Flow Architecture for Interfaces .................................................. 10
Figure 2: Request and Response Overview ............................................................ 12
Figure 3: Interfaces as a Structural Element ......................................................... 14
Figure 4: Interface Schema ............................................................................. 16
Figure 5: Request State Diagram .................................................................... 24
Figure 6: Response State Diagram .................................................................... 27
Figure 7: Success Scenario ............................................................................. 28
Figure 8: Responder - Immediate Failure ......................................................... 29
Figure 9: Responder Fails While Providing a Service ......................................... 30
Figure 10: Requester Fails During a Service Request ......................................... 31
Figure 11: Requester Makes Unexpected State Change ....................................... 32
Figure 12: Responder Makes Unexpected State Change ....................................... 33
Figure 13: Requester/Responder Communication Failures ..................................... 34
List of Tables

Table 1: Sequence of interaction between pieces of equipment ............................. 12
Table 2: Interface types ....................................................................................... 17
Table 3: InterfaceState Event ............................................................................ 21
Table 4: Event Data Item types for Interface ...................................................... 22
Table 5: Request States ....................................................................................... 23
Table 6: Response States .................................................................................... 25
1 Purpose of This Document

This document, *MTConnect Standard: Part 5.0 - Interfaces* of the MTConnect® Standard, defines a structured data model used to organize information required to coordinate inter-operations between pieces of equipment.

This data model is based on an [Interaction Model](#) that defines the exchange of information between pieces of equipment and is organized in the MTConnect Standard as the XML element Interfaces.

[Interfaces](#) is modeled as an extension to the MTConnectDevices and MTConnect-Streams XML documents. Interfaces leverages similar rules and terminology as those used to describe a component in the MTConnectDevices XML document. Interfaces also uses similar methods for reporting data to those used in the MTConnectStreams XML document.

As defined in *MTConnect Standard: Part 2.0 - Devices Information Model*, Interfaces is modeled as a [Top Level](#) component in the MTConnectDevices document (see Figure 3). Each individual Interface XML element is modeled as a [Lower Level](#) component of Interfaces. The data associated with each Interface is modeled within each Lower Level component.

Note: See *MTConnect Standard: Part 2.0 - Devices Information Model* and *MTConnect Standard: Part 3.0 - Streams Information Model* of the MTConnect Standard for information on how Interfaces is structured in the XML documents which are returned from an [Agent](#) in response to a probe, sample, or current request.
2 Terminology and Conventions

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

2.1 Glossary

CDATA

General meaning:
An abbreviation for Character Data.
CDATA is used to describe a value (text or data) published as part of an XML element.
For example, "This is some text" is the CDATA in the XML element:
<Message ...>This is some text</Message>
Appears in the documents in the following form: CDATA

XML

Stands for eXtensible Markup Language.
XML defines a set of rules for encoding documents that both a human-readable and machine-readable.
XML is the language used for all code examples in the MTConnect Standard.
Refer to http://www.w3.org/XML for more information about XML.

Agent

Refers to an MTConnect Agent.
Software that collects data published from one or more piece(s) of equipment, organizes that data in a structured manner, and responds to requests for data from client software systems by providing a structured response in the form of a Response Document that is constructed using the semantic data models defined in the Standard.
Appears in the documents in the following form: Agent

Child Element

A portion of a data modeling structure that illustrates the relationship between an element and the higher-level Parent Element within which it is contained.
Appears in the documents in the following form: Child Element
Component

General meaning:
A Structural Element that represents a physical or logical part or subpart of a piece of equipment.

Appears in the documents in the following form: Component

Used in Information Models:
A data modeling element used to organize the data being retrieved from a piece of equipment.

- When used as an XML container to organize Lower Level Component elements.
  Appears in the documents in the following form: Components.
- When used as an abstract XML element. Component is replaced in a data model by a type of Component element. Component is also an XML container used to organize Lower Level Component elements, Data Entities or both.
  Appears in the documents in the following form: Component.

Controlled Vocabulary

A restricted set of values that may be published as the Valid Data Value for a Data Entity.

Appears in the documents in the following form: Controlled Vocabulary

Current Request

A Current Request is a Request to an Agent to produce an MTConnectStreams Response Document containing the Observations Information Model for a snapshot of the latest observations at the moment of the Request or at a given sequence number.

Data Entity

A primary data modeling element that represents all elements that either describe data items that may be reported by an Agent or the data items that contain the actual data published by an Agent.

Appears in the documents in the following form: Data Entity

Devices Information Model

A set of rules and terms that describes the physical and logical configuration for a piece of equipment and the data that may be reported by that equipment.

Appears in the documents in the following form: Devices Information Model
**Element Name**

A descriptive identifier contained in both the start-tag and end-tag of an XML element that provides the name of the element.

Appears in the documents in the following form: element name.

Used to describe the name for a specific XML element:

Reference to the name provided in the start-tag, end-tag, or empty-element tag for an XML element.

Appears in the documents in the following form: **Element Name**.

**Equipment Metadata**

See [Metadata](#).

**Information Model**

The rules, relationships, and terminology that are used to define how information is structured.

For example, an information model is used to define the structure for each MTConnect Response Document; the definition of each piece of information within those documents and the relationship between pieces of information.

Appears in the documents in the following form: **Information Model**.

**Interaction Model**

Defines how information is exchanged across an **Interface** between independent systems.

**Interface**

The means by which communication is achieved between independent systems.

**Lower Level**

A nested element that is below a higher level element.

**Metadata**

Data that provides information about other data.

For example, **Equipment Metadata** defines both the **Structural Elements** that represent the physical and logical parts and sub-parts of each piece of equipment, the relationships between those parts and sub-parts, and the definitions of the **Data Entities** associated with that piece of equipment.

Appears in the documents in the following form: **Metadata** or **Equipment Metadata**.
**MTConnect Agent**

See definition for **Agent**.

**MTConnectDevices Response Document**

A **Response Document** published by an **MTConnect Agent** in response to a **Probe Request**.

**MTConnectStreams Response Document**

A **Response Document** published by an **MTConnect Agent** in response to a **Current Request** or a **Sample Request**.

**observation**

The observed value of a property at a point in time.

**Observations Information Model**

An **Information Model** that describes the **Streaming Data** reported by a piece of equipment.

**Parent Element**

An XML element used to organize **Lower Level** child elements that share a common relationship to the **Parent Element**. Appears in the documents in the following form: **Parent Element**.

**Probe Request**

A **Probe Request** is a **Request** to an **Agent** to produce an **MTConnectDevices Response Document** containing the **Devices Information Model**.

**Publish/Subscribe**

In the MTConnect Standard, a communications messaging pattern that may be used to publish **Streaming Data** from an **Agent**. When a **Publish/Subscribe** communication method is established between a client software application and an **Agent**, the **Agent** will repeatedly publish a specific **MTConnectStreams document** at a defined period.

Appears in the documents in the following form: **Publish/Subscribe**.

**Request**

A communications method where a client software application transmits a message to an **Agent**. That message instructs the **Agent** to respond with specific information.

Appears in the documents in the following form: **Request**.
**Requester**
An entity that initiates a Request for information in a communications exchange.
Appears in the documents in the following form: \textit{Requester}.

**Responder**
An entity that responds to a Request for information in a communications exchange.
Appears in the documents in the following form: \textit{Responder}.

**Response Document**
An electronic document published by an MTConnect Agent in response to a Probe Request, Current Request, Sample Request or Asset Request.

**Sample Request**
A Sample Request is a Request to an Agent to produce a MTConnectStreams Response Document containing the Observations Information Model for a set of timestamped observations made by Components.

**semantic data model**
A methodology for defining the structure and meaning for data in a specific logical way.
It provides the rules for encoding electronic information such that it can be interpreted by a software system.
Appears in the documents in the following form: \textit{semantic data model}.

**sequence number**
The primary key identifier used to manage and locate a specific piece of Streaming Data in an Agent.
sequence number is a monotonically increasing number within an instance of an Agent.
Appears in the documents in the following form: \textit{sequence number}.

**Streaming Data**
The values published by a piece of equipment for the Data Entities defined by the Equipment Metadata.
Appears in the documents in the following form: \textit{Streaming Data}.

**Structural Element**
General meaning:
An XML element that organizes information that represents the physical and logical parts and sub-parts of a piece of equipment.

Appears in the documents in the following form: **Structural Element**.

Used to indicate hierarchy of Components:

When used to describe a primary physical or logical construct within a piece of equipment.

Appears in the documents in the following form: **Top Level Structural Element**.

When used to indicate a **Child Element** which provides additional detail describing the physical or logical structure of a **Top Level Structural Element**.

Appears in the documents in the following form: **Lower Level Structural Element**.

**Top Level**

**Structural Elements** that represent the most significant physical or logical functions of a piece of equipment.

**Valid Data Value**

One or more acceptable values or constrained values that can be reported for a **Data Entity**.

Appears in the documents in the following form: **Valid Data Value**(s).

### 2.2 Acronyms

**AMT**

The Association for Manufacturing Technology

### 2.3 MTConnect References

- [MTConnect Part 5.0](#) *MTConnect Standard: Part 5.0 - Interfaces*. Version 1.7.0.
3 Interfaces Overview

In many manufacturing processes, multiple pieces of equipment must work together to perform a task. The traditional method for coordinating the activities between individual pieces of equipment is to connect them using a series of wires to communicate equipment states and demands for action. These interactions use simple binary ON/OFF signals to accomplished their intention.

In the MTConnect Standard, Interfaces provides a means to replace this traditional method for interconnecting pieces of equipment with a structured Interaction Model that provides a rich set of information used to coordinate the actions between pieces of equipment. Implementers may utilize the information provided by this data model to (1) realize the interaction between pieces of equipment and (2) to extend the functionality of the equipment to improve the overall performance of the manufacturing process.

The Interaction Model used to implement Interfaces provides a lightweight and efficient protocol, simplifies failure recovery scenarios, and defines a structure for implementing a Plug-And-Play relationship between pieces of equipment. By standardizing the information exchange using this higher-level semantic information model, an implementer may more readily replace a piece of equipment in a manufacturing system with any other piece of equipment capable of providing similar Interaction Model functions.

Two primary functions are required to implement the Interaction Model for an Interfaces and manage the flow of information between pieces of equipment. Each piece of equipment needs to have the following:

- An Agent which provides:
  - The data required to implement the Interaction Model
  - Any other data from a piece of equipment needed to implement the Interface – operating states of the equipment, position information, execution modes, process information, etc.

- A client software application that enables the piece of equipment to acquire and interpret information from another piece of equipment.

3.1 Interfaces Architecture

MTConnect Standard is based on a communications method that provides no direct way for one piece of equipment to change the state of or cause an action to occur in another
piece of equipment. The Interaction Model used to implement Interfaces is based on a Publish/Subscribe type of communications as described in MTConnect Standard Part 1.0 - Overview and Fundamentals and utilizes a Request and Response information exchange mechanism. For Interfaces, pieces of equipment must perform both the publish (Agent) and subscribe (client) functions.

Note: The current definition of Interfaces addresses the interaction between two pieces of equipment. Future releases of the MTConnect Standard may address the interaction between multiple (more than two) pieces of equipment.

Figure 1 provides a high-level overview of a typical system architecture used to implement Interfaces.

Figure 1: Data Flow Architecture for Interfaces

Note: The data flow architecture illustrated in Figure 1 was historically referred to in the MTConnect Standard as a read-read concept.

In the implementation of the Interaction Model for Interfaces, two pieces of equipment can exchange information in the following manner. One piece of equipment indicates a Request for service by publishing a type of Request using a data item provided through an Agent as defined in Section 7 - Interfaces for Devices and Streams Information Models. The client associated with the second piece of equipment, which is subscribing to data from the first machine, detects and interprets that Request. If the second machine chooses to take any action to fulfill this Request, it can indicate its acceptance by publishing a Response using a data item provided through its Agent. The client on the first piece of equipment continues to monitor information from the second piece of equipment until it detects an indication that the Response to the Request has been completed or has failed.

An example of this type of interaction between pieces of equipment can be represented.
by a machine tool that wants the material to be loaded by a robot. In this example, the
machine tool is the Requester and the robot is the Responder. On the other hand, if the
robot wants the machine tool to open a door, the robot becomes the Requester and the
machine tool the Responder.

3.2 Request and Response Information Exchange

The concept of a Request and Response information exchange is not unique to MTConnect Interfaces. This style of communication is used in many different types of environments and technologies.

An early version of a Request and Response information exchange was used by early sailors. When it was necessary to communicate between two ships before radio communications were available, or when secrecy was required, a sailor on each ship could communicate with the other using flags as a signaling device to request information or actions. The responding ship could acknowledge those requests for action and identify when the requested actions were completed.

The same basic Request and Response concept is implemented by MTConnect Interfaces using the EVENT data items defined in Section 4 - Interfaces for Devices and Streams Information Models.

The DataItem elements defined by the Interaction Model each have a Request and Response subtype. These subtypes identify if the data item represents a Request or a Response. Using these data items, a piece of equipment changes the state of its Request or Response to indicate information that can be read by the other piece of equipment. To aid in understanding how the Interaction Model functions, one can view this Interaction Model as a simple state machine.

The interaction between two pieces of equipment can be described as follows. When the Requester wants an activity to be performed, it transitions its Request state from a READY state to an ACTIVE state. In turn, when the client on the Responder reads this information and interprets the Request, the Responder announces that it is performing the requested task by changing its response state to ACTIVE. When the action is finished, the Responder changes its response state to COMPLETE. This pattern of Request and Response provides the basis for the coordination of actions between pieces of equipment. These actions are implemented using EVENT category data items. (See Section 4 - Interfaces for Devices and Streams Information Models for details on the Event type data items defined for Interfaces.)

Note: The implementation details of how the Responder piece of equipment reacts to the Request and then completes the requested task are up to the implementer.
Figure 2 provides an example of the Request and Response state machine:

The initial condition of both the Request and Response states on both pieces of equipment is READY. The dotted lines indicate the on-going communications that occur to monitor the progress of the interactions between the pieces of equipment.

The interaction between the pieces of equipment as illustrated in Figure 2 progresses through the sequence in Table 1.

Table 1: Sequence of interaction between pieces of equipment

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The Request transitions from READY to ACTIVE signaling that a service is needed.</td>
</tr>
<tr>
<td>2</td>
<td>The Response detects the transition of the Request.</td>
</tr>
<tr>
<td>3</td>
<td>The Response transitions from READY to ACTIVE indicating that it is performing the action.</td>
</tr>
<tr>
<td>4</td>
<td>Once the action has been performed, the Response transitions to COMPLETE.</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>5</td>
<td>The Request detects the action is COMPLETE.</td>
</tr>
<tr>
<td>6</td>
<td>The Request transitions back to READY acknowledging that the service has been performed.</td>
</tr>
<tr>
<td>7</td>
<td>The Response detects the Request has returned to READY.</td>
</tr>
<tr>
<td>8</td>
<td>In recognition of this acknowledgement, the Response transitions back to READY.</td>
</tr>
</tbody>
</table>

After the final action has been completed, both pieces of equipment are back in the READY state indicating that they are able to perform another action.
4 Interfaces for Devices and Streams Information Models

The Interaction Model for implementing Interfaces is defined in the MTConnect Standard as an extension to the MTConnectDevices and MTConnectStreams XML documents.

A piece of equipment MAY support multiple different Interfaces. Each piece of equipment supporting Interfaces MUST organize the information associated with each Interface in a Top Level component called Interfaces. Each individual Interface is modeled as a Lower Level component called Interface. Interface is an abstract type XML element and will be replaced in the XML documents by specific Interface types defined below. The data associated with each Interface is modeled as data items within each of these Lower Level Interface components.

The XML tree in Figure 3 illustrates where Interfaces is modeled in the Devices Information Model for a piece of equipment.

Figure 3: Interfaces as a Structural Element
4.1 Interfaces

 Interfaces is an XML Structural Element in the MTConnectDevices XML document.
 Interfaces is a container type XML element. Interfaces is used to group information describing Lower Level Interface XML elements, which each provide information for an individual Interface.

 If the Interfaces container appears in the XML document, it MUST contain one or more Interface type XML elements.

4.2 Interface

 Interface is the next level of Structural Element in the MTConnectDevices XML document. As an abstract type XML element, Interface will be replaced in the XML documents by specific Interface types defined below.

 Each Interface is also a container type element. As a container, the Interface XML element is used to organize information required to implement the Interaction Model for an Interface. It also provides structure for describing the Lower Level Structural Elements associated with the Interface. Each Interface contains Data Entities available from the piece of equipment that may be needed to coordinate activities with associated pieces of equipment.

 The information provided by a piece of equipment for each Interface is returned in a ComponentStream container of an MTConnectStreams document in the same manner as all other types of components.

4.2.1 XML Schema Structure for Interface

 The XML schema in Figure 4 represents the structure of an Interface XML element.

 The schema for an Interface element is the same as defined for Component elements described in Section 4.4 in MTConnect Standard: Part 2.0 - Devices Information Model of the MTConnect Standard. The Figure 4 shows the attributes defined for Interface and the elements that may be associated with Interface.
Figure 4: Interface Schema
Refer to MTConnect Standard: Part 2.0 - Devices Information Model, Section 4.4 for complete descriptions of the attributes and elements that are illustrated in the Figure 4 for Interface.

### 4.2.2 Interface Types

As an abstract type XML element, Interface is replaced in the MTConnectDevices document with a XML element representing a specific type of Interface. An initial list of Interface types is defined in the Table 2.

**Table 2: Interface types**

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BarFeederInterface</td>
<td>BarFeederInterface provides the set of information used to coordinate the operations between a Bar Feeder and another piece of equipment.</td>
</tr>
<tr>
<td></td>
<td>Bar Feeder is a piece of equipment that pushes bar stock (i.e., long pieces of material of various shapes) into an associated piece of equipment – most typically a lathe or turning center.</td>
</tr>
<tr>
<td>Interface</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| MaterialHandlerInterface | MaterialHandlerInterface provides the set of information used to coordinate the operations between a piece of equipment and another associated piece of equipment used to automatically handle various types of materials or services associated with the original piece of equipment.  
A material handler is a piece of equipment capable of providing any one, or more, of a variety of support services for another piece of equipment or a process:  
- Loading/unloading material or tooling  
- Part inspection  
- Testing  
- Cleaning  
- Etc.  
A robot is a common example of a material handler. |
| DoorInterface      | DoorInterface provides the set of information used to coordinate the operations between two pieces of equipment, one of which controls the operation of a door.  
The piece of equipment that is controlling the door **MUST** provide the data item DOOR_STATE as part of the set of information provided. |
Continuation of Table 2

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChuckInterface</td>
<td>ChuckInterface provides the set of information used to coordinate the operations between two pieces of equipment, one of which controls the operation of a chuck.</td>
</tr>
<tr>
<td></td>
<td>The piece of equipment that is controlling the chuck <strong>MUST</strong> provide the data item CHUCK_STATE as part of the set of information provided.</td>
</tr>
</tbody>
</table>

Note: Additional Interface types may be defined in future releases of the MT-Connect Standard.

In order to implement the Interaction Model for Interfaces, each piece of equipment associated with an Interface **MUST** provide an Interface XML element for that type of Interface. A piece of equipment **MAY** support any number of unique Interfaces.

### 4.2.3 Data for Interface

Each Interface **MUST** provide (1) the data associated with the specific Interface to implement the Interaction Model and (2) any additional data that may be needed by another piece of equipment to understand the operating states and conditions of the first piece of equipment as it applies to the Interface.

Details on data items specific to the Interaction Model for each type of Interface are provided in Section 4.2.4 - Data Items for Interface.

An implementer may choose any other data available from a piece of equipment to describe the operating states and other information needed to support an Interface.

### 4.2.3.1 References for Interface

Some of the data items needed to support a specific Interface may already be defined elsewhere in the XML document for a piece of equipment. However, the implementer may not be able to directly associate this data with the Interface since the MTConnect Standard does not permit multiple occurrences of a piece of data to be configured in a XML document. References provides a mechanism for associating information defined elsewhere...
in the [Information Model] for a piece of equipment with a specific [Interface].

References is an [XML] container that organizes pointers to information defined elsewhere in the [XML] document for a piece of equipment. References MAY contain one or more Reference [XML] elements.

Reference is an [XML] element that provides an individual pointer to information that is associated with another Structural Element or Data Entity defined elsewhere in the [XML] document that is also required for an Interface.

References is an economical syntax for providing interface specific information without directly duplicating the occurrence of the data. It provides a mechanism to include all necessary information required for interaction and deterministic information flow between pieces of equipment.

For more information on the definition for References and Reference, see Section 4.7 and 4.8 of MTConnect Standard: Part 2.0 - Devices Information Model.

### 4.2.4 Data Items for Interface

Each Interface [XML] element contains data items which are used to communicate information required to execute the [Interface]. When these data items are read by another piece of equipment, that piece of equipment can then determine the actions that it may take based upon that data.

Some data items MAY be directly associated with the Interface element and others will be organized in a Lower Level References [XML] element.

It is up to an implementer to determine which additional data items are required for a particular Interface.

The data items that have been specifically defined to support the implementation of an Interface are provided below.

#### 4.2.4.1 INTERFACE_STATE for Interface

INTERFACE_STATE is a data item specifically defined for Interfaces. It defines the operational state of the Interface. This is an indicator identifying whether the Interface is functioning or not.

An INTERFACE_STATE data item MUST be defined for every Interface [XML] ele-
INTERFACE_STATE is reported in the MTConnectStreams XML document as InterfaceState. InterfaceState reports one of two states – ENABLED or DISABLED, which are provided in the CDATA for InterfaceState.

The Table 3 shows both the INTERFACE_STATE data item as defined in the MTConnectDevices document and the corresponding Element Name that MUST be reported in the MTConnectStreams document.

Table 3: InterfaceState Event

<table>
<thead>
<tr>
<th>DataItem Type</th>
<th>Element Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERFACE_STATE</td>
<td>InterfaceState</td>
<td>The current functional or operational state of an Interface type element indicating whether the Interface is active or not currently functioning.</td>
</tr>
</tbody>
</table>

**Valid Data Values:**

- **ENABLED:** The Interface is currently operational and performing as expected.
- **DISABLED:** The Interface is currently not operational.

When the INTERFACE_STATE is DISABLED, the state of all data items that are specific for the Interaction Model associated with that Interface MUST be set to NOT_READY.

### 4.2.4.2 Specific Data Items for the Interaction Model for Interface

A special set of data items have been defined to be used in conjunction with Interface type elements. When modeled in the MTConnectDevices document, these data items are all Data Entities in the EVENT category (See MTConnect Standard: Part 3.0 - Streams Information Model for details on how the corresponding data items are reported in the MTConnectStreams document). They provide information from a piece of equipment to Request a service to be performed by another associated piece of equipment; and for
the associated piece of equipment to indicate its progress in performing its Response to the Request for service.

Many of the data items describing the services associated with an Interface are paired to describe two distinct actions – one to Request an action to be performed and a second to reverse the action or to return to an original state. For example, a DoorInterface will have two actions OPEN_DOOR and CLOSE_DOOR. An example of an implementation of this would be a robot that indicates to a machine that it would like to have a door opened so that the robot could extract a part from the machine and then asks the machine to close that door once the part has been removed.

When these data items are used to describe a service associated with an Interface, they MUST have one of the following two subType elements: REQUEST or RESPONSE. These subType elements MUST be specified to define whether the piece of equipment is functioning as the Requester or Responder for the service to be performed. The Requester MUST specify the REQUEST subType for the data item and the Responder MUST specify a corresponding RESPONSE subType for the data item to enable the coordination between the two pieces of equipment.

These data items and their associated subType provide the basic structure for implementing the Interaction Model for an Interface.

Table 4 provides a list of the data items that have been defined to identify the services to be performed for or by a piece of equipment associated with an Interface.

The Table also provides the corresponding transformed Element Name for each data item that MAY be returned by an Agent as an Event type XML Data Entity in the MTConnectStreams XML document. The Controlled Vocabulary for each of these data items are defined in Section 4.2.4.3 - Event States for Interfaces.

<table>
<thead>
<tr>
<th>DataItem Type</th>
<th>Element Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATERIAL_FEED</td>
<td>MaterialFeed</td>
<td>Service to advance material or feed product to a piece of equipment from a continuous or bulk source.</td>
</tr>
<tr>
<td>MATERIAL_CHANGE</td>
<td>MaterialChange</td>
<td>Service to change the type of material or product being loaded or fed to a piece of equipment.</td>
</tr>
<tr>
<td>MATERIAL_RETRACT</td>
<td>MaterialRetract</td>
<td>Service to remove or retract material or product.</td>
</tr>
</tbody>
</table>
### Control Vocabulary

For each of the data items above, the `Valid Data Values` for the `CDATA` that is returned for these data items in the `MTConnectStreams` document is defined by a `Controlled Vocabulary`. This `Controlled Vocabulary` represents the state information to be communicated by a piece of equipment for the data items defined in the `Table 4`.

The `Request` portion of the `Interaction Model` for `Interfaces` has four states as defined in the `Table 5`.

#### Table 5: Request States

<table>
<thead>
<tr>
<th>Request State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOT_READY</td>
<td>The <code>Requester</code> is not ready to make a <code>Request</code></td>
</tr>
<tr>
<td>READY</td>
<td>The <code>Requester</code> is prepared to make a <code>Request</code> but no <code>Request</code> for service is required. The <code>Requester</code> will transition to ACTIVE when it needs a service to be performed.</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>The <code>Requester</code> has initiated a <code>Request</code> for a service and the service has not yet been completed by the <code>Responder</code></td>
</tr>
</tbody>
</table>
Continuation of Table 5

<table>
<thead>
<tr>
<th>Request State</th>
<th>Description</th>
</tr>
</thead>
</table>
| FAIL          | **CONDITION 1:**
|               | When the Requester has detected a failure condition, it indicates to the Responder to either not initiate an action or stop its action before it completes by changing its state to FAIL. |
|               | **CONDITION 2:**
|               | If the Responder changes its state to FAIL, the Requester MUST change its state to FAIL. |
|               | **ACTIONS:**
|               | After detecting a failure, the Requester SHOULD NOT change its state to any other value until the Responder has acknowledged the FAIL state by changing its state to FAIL. |
|               | Once the FAIL state has been acknowledged by the Responder, the Requester may attempt to clear its FAIL state. |
|               | As part of the attempt to clear the FAIL state, the Requester MUST reset any partial actions that were initiated and attempt to return to a condition where it is again ready to perform a service. If the recovery is successful, the Requester changes its Request state from FAIL to READY. If for some reason the Requester is not again prepared to perform a service, it transitions its state from FAIL to NOT_READY. |

Figure 5 shows a graphical representation of the possible state transitions for a Request.

**Figure 5:** Request State Diagram
The Response portion of the Interaction Model for Interfaces has five states as defined in the Table 6.

Table 6: Response States

<table>
<thead>
<tr>
<th>Response State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOT_READY</td>
<td>The <strong>Responder</strong> is not ready to perform a service.</td>
</tr>
<tr>
<td>READY</td>
<td>The <strong>Responder</strong> is prepared to react to a Request, but no Request for service has been detected. The <strong>Responder</strong> <strong>MUST</strong> transition to ACTIVE to inform the <strong>Requester</strong> that it has detected and accepted the Request and is in the process of performing the requested service. If the <strong>Responder</strong> is not ready to perform a Request, it <strong>MUST</strong> transition to a NOT_READY state.</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>The <strong>Responder</strong> has detected and accepted a Request for a service and is in the process of performing the service, but the service has not yet been completed. In normal operation, the <strong>Responder</strong> <strong>MUST NOT</strong> change its state to ACTIVE unless the <strong>Requester</strong> state is ACTIVE.</td>
</tr>
</tbody>
</table>
Continuation of Table 6

<table>
<thead>
<tr>
<th>Response State</th>
<th>Description</th>
</tr>
</thead>
</table>
| FAIL           | CONDITION 1:  
The Responder has failed while executing the actions required to perform a service and the service has not yet been completed or the Responder has detected that the Requester has unexpectedly changed state.  
CONDITION 2:  
If the Requester changes its state to FAIL, the Responder MUST change its state to FAIL.  
ACTIONS:  
After entering a FAIL state, the Responder SHOULD NOT change its state to any other value until the Requester has acknowledged the FAIL state by changing its state to FAIL.  
Once the FAIL state has been acknowledged by the Requester, the Responder may attempt to clear its FAIL state.  
As part of the attempt to clear the FAIL state, the Responder MUST reset any partial actions that were initiated and attempt to return to a condition where it is again ready to perform a service. If the recovery is successful, the Responder changes its Response state from FAIL to READY. If for some reason the Responder is not again prepared to perform a service, it transitions its state from FAIL to NOT_READY. |
| COMPLETE       | The Responder has completed the actions required to perform the service.  
The Responder MUST remain in the COMPLETE state until the Requester acknowledges that the service is complete by changing its state to READY.  
At that point, the Responder MUST change its state to either READY if it is again prepared to perform a service or NOT READY if it is not prepared to perform a service. |

The state values described in the Table 6 and Table 6 MUST be provided in the CDATA for each of the Interface specific data items provided in the MTConnectStreams document.

Figure 6 shows a graphical representation of the possible state transitions for a Response:
Figure 6: Response State Diagram
5 Operation and Error Recovery

The Request/Response state model implemented for Interfaces may also be represented by a graphical model. The scenario in Figure 7 demonstrates the state transitions that occur during a successful Request for service and the resulting Response to fulfill that service.

![Figure 7: Success Scenario]

5.1 Request/Response Failure Handling and Recovery

A significant feature of the Request/Response Interaction Model is the ability for either piece of equipment to detect a failure associated with either the Request or Response actions. When either a failure or unexpected action occurs, the Request and the Response portion of the Interaction Model can announce a FAIL state upon detecting a problem. The following are graphical models describing multiple scenarios where either the Requester or Responder detects and reacts to a failure. In these examples, either the Requester or Responder announces the detection of a failure by setting either the Request or the Response state to FAIL.

Once a failure is detected, the Interaction Model provides information from each piece of
equipment as they attempt to recover from a failure, reset all of their functions associated with the Interface to their original state, and return to normal operation.

The following are scenarios that describe how pieces of equipment may react to different types of failures and how they indicate when they are again ready to request a service or respond to a request for service after recovering from those failures:

Scenario #1 – Responder Fails Immediately

In this scenario, a failure is detected by the Responder immediately after a Request for service has been initiated by the Requester.

In this case, the Request transitions to ACTIVE and the Responder immediately detects a failure before it can transition the Response state to ACTIVE. When this occurs, the Responder transitions the Response state to FAIL.

After detecting that the Responder has transitioned its state to FAIL, the Requester MUST change its state to FAIL.

The Requester, as part of clearing a failure, resets any partial actions that were initiated and attempts to return to a condition where it is again ready to request a service. If the recovery is successful, the Requester changes its state from FAIL to READY. If for some reason the Requester cannot return to a condition where it is again ready to request a service, it transitions its state from FAIL to NOT_READY.
The **Responder** as part of clearing a failure, resets any partial actions that were initiated and attempts to return to a condition where it is again ready to perform a service. If the recovery is successful, the **Responder** changes its **Response** state from **FAIL** to **READY**. If for some reason the **Responder** is not again prepared to perform a service, it transitions its state from **FAIL** to **NOT READY**.

**Scenario #2 – **Responder** Fails While Providing a Service**

This is the most common failure scenario. In this case, the **Responder** will begin the actions required to provide a service. During these actions, the **Responder** detects a failure and transitions its **Response** state to **FAIL**.

![Figure 9: Responder Fails While Providing a Service](image)

When a **Requester** detects a failure of a **Responder**, it transitions its state from **ACTIVE** to **FAIL**.

The **Requester** resets any partial actions that were initiated and attempts to return to a condition where it is again ready to request a service. If the recovery is successful, the **Requester** changes its state from **FAIL** to **READY** if the failure has been cleared and it is again prepared to request another service. If for some reason the **Requester** cannot return to a condition where it is again ready to request a service, it transitions its state from **FAIL** to **NOT READY**.

The **Responder** as part of clearing a failure, resets any partial actions that were initiated and attempts to return to a condition where it is again ready to perform a service. If the recovery is successful, the **Responder** changes its **Response** state from **FAIL** to **READY** if
it is again prepared to perform a service. If for some reason the Responder is not again prepared to perform a service, it transitions its state from FAIL to NOT_READY.

Scenario #3 – Requester Failure During a Service Request

In this scenario, the Responder will begin the actions required to provide a service. During these actions, the Requester detects a failure and transitions its Request state to FAIL.

Figure 10: Requester Fails During a Service Request

When the Responder detects that the Requester has transitioned its Request state to FAIL, the Responder also transitions its Response state to FAIL.

The Requester, as part of clearing a failure, resets any partial actions that were initiated and attempts to return to a condition where it is again ready to request a service. If the recovery is successful, the Requester changes its state from FAIL to READY. If for some reason the Requester cannot return to a condition where it is again ready to request a service, it transitions its state from FAIL to NOT_READY.

The Responder, as part of clearing a failure, resets any partial actions that were initiated and attempts to return to a condition where it is again ready to perform a service. If the recovery is successful, the Responder changes its Response state from FAIL to READY. If for some reason the Responder is not again prepared to perform a service, it transitions its state from FAIL to NOT READY.

Scenario #4 – Requester Changes to an Unexpected State While Responder is Providing a Service

In some cases, a Requester may transition to an unexpected state after it has initiated a
February 25, 2021

As demonstrated in Figure 11, the Requester has initiated a Request for service and its Request state has been changed to ACTIVE. The Responder begins the actions required to provide the service. During these actions, the Requester transitions its Request state back to READY before the Responder can complete its actions. This SHOULD be regarded as a failure of the Requester.

Figure 11: Requester Makes Unexpected State Change

In this case, the Responder reacts to this change of state of the Requester in the same way as though the Requester had transitioned its Request state to FAIL (i.e., the same as in Scenario #3 above).

At this point, the Responder then transitions its Response state to FAIL.

The Responder resets any partial actions that were initiated and attempts to return to its original condition where it is again ready to perform a service. If the recovery is successful, the Responder changes its Response state from FAIL to READY. If for some reason the Responder is not again prepared to perform a service, it transitions its state from FAIL to NOT_READY.

Note: The same scenario exists if the Requester transitions its Request state to NOT_READY. However, in this case, the Requester then transitions its Request state to READY after it resets all of its functions back to a condition where it is again prepared to make a Request for service.
Scenario #5 – Responder Changes to an Unexpected State While Providing a Service

Similar to Scenario #5, a Responder may transition to an unexpected state while providing a service.

As demonstrated in Figure 12, the Responder is performing the actions to provide a service and the Response state is ACTIVE. During these actions, the Responder transitions its state to NOT_READY before completing its actions. This should be regarded as a failure of the Responder.

![Diagram: Requester and Responder state transitions](image)

**Figure 12: Responder Makes Unexpected State Change**

Upon detecting an unexpected state change of the Responder, the Requester transitions its state to FAIL.

The Requester resets any partial actions that were initiated and attempts to return to a condition where it is again ready to request a service. If the recovery is successful, the Requester changes its state from FAIL to READY. If for some reason the Requester cannot return to a condition where it is again ready to request a service, it transitions its state from FAIL to NOT_READY.

Since the Responder has failed to an invalid state, the condition of the Responder is unknown. Where possible, the Responder should try to reset to an initial state.

The Responder, as part of clearing the cause for the change to the unexpected state, should attempt to reset any partial actions that were initiated and then return to a condition where it is again ready to perform a service. If the recovery is successful, the Responder changes its Response state from the unexpected state to READY. If for some reason the Responder...
February 25, 2021

is not again prepared to perform a service, it maintains its state as NOT_READY.

Scenario #6 – Responder or Requester Become UNAVAILABLE or Experience a Loss of Communications

In this scenario, a failure occurs in the communications connection between the Responder and Requester. This failure may result from the InterfaceState from either piece of equipment returning a value of UNAVAILABLE or one of the pieces of equipment does not provide a heartbeat within the desired amount of time (See MTConnect Standard Part 1.0 - Overview and Fundamentals for details on heartbeat).

Figure 13: Requester/Responder Communication Failures

When one of these situations occurs, each piece of equipment assumes that there has been a failure of the other piece of equipment.

When normal communications are re-established, neither piece of equipment should assume that the Request/Response state of the other piece of equipment remains valid. Both pieces of equipment should set their state to FAIL.

The Requester as part of clearing its FAIL state, resets any partial actions that were initiated and attempts to return to a condition where it is again ready to request a service. If the recovery is successful, the Requester changes its state from FAIL to READY. If for some reason the Requester cannot return to a condition where it is again ready to request
a service, it transitions its state from FAIL to NOT_READY.

The Responder as part of clearing its FAIL state, resets any partial actions that were initiated and attempts to return to a condition where it is again ready to perform a service. If the recovery is successful, the Responder changes its Response state from FAIL to READY. If for some reason the Responder is not again prepared to perform a service, it transitions its state from FAIL to NOT_READY.
Appendices

A Bibliography


