



MTConnect[®] Standard

Version 1.5.0

Prepared for: MTConnect Institute

Prepared on: March 31, 2018

CONTENTS

Part 1 - Overview and Fundamentals v1.5.0

Part 2 - Devices v1.5.0

Part 3 - Streams v1.5.0

Part 4 - Assets v1.5.0

Part 4.1 - Cutting Tools v1.5.0

Part 5 - Interfaces v1.5.0



MTConnect® Standard

Part 1.0 – Overview and Fundamentals

Version 1.5.0

Prepared for: MTConnect Institute
Prepared on: December 2, 2019

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/>.

MTConnect Specification and Materials

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 Overview of MTConnect | 2 |
| 2 Purpose of This Document | 7 |
| 3 Terminology and Conventions | 8 |
| 3.1 Glossary | 8 |
| 3.2 MTConnect References | 32 |
| 4 MTConnect Standard | 33 |
| 4.1 MTConnect Documents Organization | 33 |
| 4.2 MTConnect Document Versioning | 34 |
| 4.2.1 Document Releases | 35 |
| 4.3 MTConnect Document Naming Conventions | 36 |
| 4.3.1 Document Title | 36 |
| 4.3.2 Electronic Document File Naming | 36 |
| 4.4 Document Conventions | 37 |
| 4.4.1 Use of MUST, SHOULD, and MAY | 37 |
| 4.4.2 Text Conventions | 37 |
| 4.4.3 Code Line Syntax and Conventions | 38 |
| 4.4.4 Semantic Data Model Content | 39 |
| 4.4.5 Referenced Standards and Specifications | 39 |
| 4.4.6 Deprecation and Deprecation Warnings | 40 |
| 4.4.6.1 Deprecation | 40 |
| 4.4.6.2 Deprecation Warning | 40 |
| 4.5 Backwards Compatibility | 41 |
| 5 MTConnect Fundamentals | 42 |
| 5.1 Agent | 42 |
| 5.1.1 Instance of an Agent | 44 |
| 5.1.2 Storage of Equipment Metadata for a Piece of Equipment | 44 |
| 5.1.3 Storage of Streaming Data | 45 |
| 5.1.3.1 Management of Streaming Data Storage | 45 |
| 5.1.3.2 Sequence Numbers | 46 |
| 5.1.3.3 Buffer Data Structure | 49 |
| 5.1.3.4 Time Stamp | 50 |
| 5.1.3.5 Recording Occurrences of Streaming Data | 51 |
| 5.1.3.6 Maintaining Last Value for Data Entities | 51 |
| 5.1.3.7 Unavailability of Data | 52 |
| 5.1.3.8 Persistence and Recovery | 52 |
| 5.1.3.9 Heartbeat | 53 |
| 5.1.3.10 Data Sets | 53 |

| | | |
|----------|--|-----------|
| 5.1.4 | Storage of Documents for MTConnect Assets | 54 |
| 5.2 | Response Documents | 56 |
| 5.2.1 | XML Documents | 57 |
| 5.3 | Semantic Data Models | 57 |
| 5.4 | Request/Response Information Exchange | 59 |
| 5.5 | Accessing Information from an Agent | 60 |
| 5.5.1 | Accessing Equipment Metadata from an Agent | 60 |
| 5.5.2 | Accessing Streaming Data from the Buffer of an Agent | 60 |
| 5.5.3 | Accessing MTConnect Assets Information from an Agent | 62 |
| 6 | XML Representation of Response Documents | 63 |
| 6.1 | Fundamentals of Using XML to Encode Response Documents | 64 |
| 6.2 | XML Declaration | 65 |
| 6.3 | Root Element | 65 |
| 6.3.1 | MTConnectDevices Root Element | 65 |
| 6.3.1.1 | MTConnectDevices Elements | 66 |
| 6.3.2 | MTConnectStreams Root Element | 67 |
| 6.3.2.1 | MTConnectStreams Elements | 68 |
| 6.3.3 | MTConnectAssets Root Element | 68 |
| 6.3.3.1 | MTConnectAssets Elements | 69 |
| 6.3.4 | MTConnectError Root Element | 69 |
| 6.3.4.1 | MTConnectError Elements | 70 |
| 6.4 | Schema and Namespace Declaration | 71 |
| 6.5 | Document Header | 71 |
| 6.5.1 | Header for MTConnectDevices | 72 |
| 6.5.1.1 | XML Schema Structure for Header for MTConnectDevices | 72 |
| 6.5.1.2 | Attributes for Header for MTConnectDevices | 72 |
| 6.5.2 | Header for MTConnectStreams | 76 |
| 6.5.2.1 | XML Schema Structure for Header for MTConnectStreams | 77 |
| 6.5.2.2 | Attributes for MTConnectStreams Header | 77 |
| 6.5.3 | Header for MTConnectAssets | 82 |
| 6.5.3.1 | XML Schema Structure for Header for MTConnectAssets | 82 |
| 6.5.3.2 | Attributes for Header for MTConnectAssets | 82 |
| 6.5.4 | Header for MTConnectError | 86 |
| 6.5.4.1 | XML Schema Structure for Header for MTConnectError | 86 |
| 6.5.4.2 | Attributes for Header for MTConnectError | 86 |
| 6.6 | Document Body | 90 |
| 6.7 | Extensibility | 91 |
| 7 | Protocol and Messaging | 93 |
| 8 | HTTP Messaging Supported by an Agent | 95 |

| | | |
|----------|--|------------|
| 8.1 | REST Interface | 95 |
| 8.2 | HTTP Request | 95 |
| 8.2.1 | authority Portion of an HTTP Request Line | 96 |
| 8.2.2 | path Portion of an HTTP Request Line | 97 |
| 8.2.3 | query Portion of an HTTP Request Line | 97 |
| 8.3 | MTConnect Request/Response Information Exchange Implemented with HTTP | 97 |
| 8.3.1 | Probe Request Implemented Using HTTP | 98 |
| 8.3.1.1 | Path Portion of the HTTP Request Line for a Probe Request | 98 |
| 8.3.1.2 | Query Portion of the HTTP Request Line for a Probe Request | 98 |
| 8.3.1.3 | Response to a Probe Request | 99 |
| 8.3.1.4 | HTTP Status Codes for a Probe Request | 99 |
| 8.3.2 | Current Request Implemented Using HTTP | 101 |
| 8.3.2.1 | Path Portion of the HTTP Request Line for a Current Request | 101 |
| 8.3.2.2 | Query Portion of the HTTP Request Line for a Current Request | 101 |
| 8.3.2.3 | Response to a Current Request | 104 |
| 8.3.2.4 | HTTP Status Codes for a Current Request | 104 |
| 8.3.3 | Sample Request Implemented Using HTTP | 106 |
| 8.3.3.1 | Path Portion of the HTTP Request Line for a Sample Request | 107 |
| 8.3.3.2 | Query Portion of the HTTP Request Line for a Sample Request | 107 |
| 8.3.3.3 | Response to a Sample Request | 111 |
| 8.3.3.4 | HTTP Status Codes for a Sample Request | 112 |
| 8.3.4 | Asset Request Implemented Using HTTP | 114 |
| 8.3.4.1 | Path Portion of the HTTP Request Line for an Asset Re- quest | 115 |
| 8.3.4.2 | Query Portion of the HTTP Request Line for an Asset Request | 115 |
| 8.3.4.3 | Response to an Asset Request | 116 |
| 8.3.4.4 | HTTP Status Codes for a Asset Request | 117 |
| 8.3.5 | HTTP Errors | 118 |
| 8.3.6 | Streaming Data | 119 |
| 8.3.6.1 | Heartbeat | 120 |
| 8.3.7 | References | 121 |
| 9 | Error Information Model | 122 |
| 9.1 | MTConnectError Response Document | 122 |
| 9.1.1 | Structural Element for MTConnectError | 122 |

| | | |
|----------|--|------------|
| 9.1.2 | Error Data Entity | 124 |
| 9.1.2.1 | XML Schema Structure for Error | 124 |
| 9.1.2.2 | Attributes for Error | 125 |
| 9.1.2.3 | Values for errorCode | 126 |
| 9.1.2.4 | CDATA for Error | 127 |
| 9.1.3 | Examples for MTConnectError | 127 |
| A | Appendices | 129 |
| A | Bibliography | 129 |
| B | Fundamentals of Using XML to Encode Response Documents | 131 |
| C | Schema and Namespace Declaration Information | 134 |

Table of Figures

| | |
|--|-----|
| Figure 1: Basic MTConnect Implementation Structure | 4 |
| Figure 2: MTConnect Architecture Model | 42 |
| Figure 3: Data Storage in Buffer | 45 |
| Figure 4: First In First Out Buffer Management | 45 |
| Figure 5: instanceId and sequence | 47 |
| Figure 6: Identifying the range of data with firstSequence and lastSequence | 47 |
| Figure 7: Identifying the range of data with from and count | 48 |
| Figure 8: Identifying the range of data with nextSequence and lastSequence | 49 |
| Figure 9: Data Storage Concept | 50 |
| Figure 10:First In First Out Asset Buffer Management | 54 |
| Figure 11:Relationship between assetId and stored Asset documents | 55 |
| Figure 12:Example Buffer | 61 |
| Figure 13:MTConnectDevices Structure | 66 |
| Figure 14:MTConnectStreams Structure | 67 |
| Figure 15:MTConnectAssets Structure | 68 |
| Figure 16:MTConnectError Structure | 70 |
| Figure 17:Header Schema Diagram for MTConnectDevices | 72 |
| Figure 18:Header Schema Diagram for MTConnectStreams | 77 |
| Figure 19:Header Schema Diagram for MTConnectAssets | 82 |
| Figure 20:Header Schema Diagram for MTConnectError | 86 |
| Figure 21:Errors Schema Diagram | 123 |
| Figure 22:Error Schema Diagram | 125 |

List of Tables

| | |
|--|-----|
| Table 1: Elements for MTConnectDevices | 66 |
| Table 2: Elements for MTConnectStreams | 68 |
| Table 3: Elements for MTConnectAssets | 69 |
| Table 4: Elements for MTConnectError | 71 |
| Table 5: MTConnectDevices Header | 73 |
| Table 6: MTConnectStreams Header | 78 |
| Table 7: MTConnectAssets Header | 83 |
| Table 8: MTConnectError Header | 87 |
| Table 9: Relationship between Response Document and Semantic Data Model | 90 |
| Table 10: Path of the HTTP Request Line for a Probe Request | 98 |
| Table 11: HTTP Status Codes for a Probe Request | 99 |
| Table 12: Path of the HTTP Request Line for a Current Request | 101 |
| Table 13: Query Parameters of the HTTP Request Line for a Current Request | 102 |
| Table 14: HTTP Status Codes for a Current Request | 105 |
| Table 15: Path of the HTTP Request Line for a Sample Request | 107 |
| Table 16: Query Parameters of the HTTP Request Line for a Sample Request | 108 |
| Table 17: HTTP Status Codes for a Sample Request | 112 |
| Table 18: Path of the HTTP Request Line for an Asset Request | 115 |
| Table 19: Query Parameters of the HTTP Request Line for an Asset Request | 115 |
| Table 20: HTTP Status Codes for an Asset Request | 117 |
| Table 21: MTConnect Errors Element | 123 |
| Table 22: Attributes for Error | 125 |
| Table 23: Values for errorCode | 126 |

1 Overview of MTConnect

2 MTConnect is a data and information exchange standard that is based on a *data dictionary*
3 of terms describing information associated with manufacturing operations. The standard
4 also defines a series of *semantic data models* that provide a clear and unambiguous repre-
5 sentation of how that information relates to a manufacturing operation. The MTConnect
6 Standard has been designed to enhance the data acquisition capabilities from equipment in
7 manufacturing facilities, to expand the use of data driven decision making in manufactur-
8 ing operations, and to enable software applications and manufacturing equipment to move
9 toward a plug-and-play environment to reduce the cost of integration of manufacturing
10 software systems.

11 The MTConnect standard supports two primary communications methods – *Request/Re-*
12 *sponse* and *Publish/Subscribe* type of communications. The *Request/Response* communi-
13 cations structure is used throughout this document to describe the functionality provided
14 by MTConnect. See *Section 8.3.6 - Streaming Data* for details describing the functionality
15 of the *Publish/Subscribe* communications structure available from an *Agent*.

16 Although the MTConnect Standard has been defined to specifically meet the requirements
17 of the manufacturing industry, it can also be readily applied to other application areas as
18 well.

19 The MTConnect Standard is an open, royalty free standard – meaning that it is available
20 for anyone to download, implement, and utilize in software systems at no cost to the
21 implementer.

22 The *semantic data models* defined in the MTConnect Standard provide the information re-
23 quired to fully characterize data with both a clear and unambiguous meaning and a mech-
24 anism to directly relate that data to the manufacturing operation where the data originated.
25 Without a *semantic data model*, client software applications must apply an additional layer
26 of logic to raw data to convey this same level of meaning and relationship to manufacturing
27 operations. The approach provided in the MTConnect Standard for modeling and organiz-
28 ing data allows software applications to easily interpret data from a wide variety of data
29 sources which reduces the complexity and effort to develop applications.

30 The data and information from a broad range of manufacturing equipment and systems
31 are addressed by the MTConnect Standard. Where the *data dictionary* and *semantic data*
32 *models* are insufficient to define some information within an implementation, an imple-
33 menter may extend the *data dictionary* and *semantic data models* to address their specific
34 requirements. See *Section 6.7 - Extensibility* for guidelines related to extensibility of the
35 MTConnect Standard.

36 To assist in implementation, the MTConnect Standard is built upon the most prevalent
37 standards in the manufacturing and software industries. This maximizes the number of
38 software tools available for implementation and provides the highest level of interoper-
39 ability with other standards, software applications, and equipment used throughout manu-
40 facturing operations.

41 Current MTConnect implementations are based on HTTP as a transport protocol and XML
42 as a language for encoding each of the *semantic data models* into electronic documents.
43 All software examples provided in the various MTConnect Standard documents are based
44 on these two core technologies.

45 The base functionality defined in the MTConnect Standard is the *data dictionary* describ-
46 ing manufacturing information and the *semantic data models*. The transport protocol and
47 the programming language used to represent or transfer the information provided by the
48 *semantic data models* are not restricted in the standard to HTTP and XML. Therefore,
49 other protocols and programming languages may be used to represent the semantic models
50 and/or transport the information provided by these data models between an *Agent* (server)
51 and a client software application as may be required by a specific implementation.

52 Note: The term "document" is used with different meanings in the MTConnect Stan-
53 dard:

- 54 • Meaning 1: The MTConnect Standard itself is comprised of multiple documents
55 each addressing different aspects of the Standard. Each document is referred to as a
56 *Part* of the Standard.
- 57 • Meaning 2: In an MTConnect implementation, the electronic documents that are
58 published from a data source and stored by an *Agent*.
- 59 • Meaning 3: In an MTConnect implementation, the electronic documents generated
60 by an *Agent* for transmission to a client software application.

61 The following will be used throughout the MTConnect Standard to distinguish be-
62 tween these different meanings for the term "document":

- 63 • MTConnect Document(s) or Document(s) shall be used to refer to printed or elec-
64 tronic document(s) that represent a *Part*(s) of the MTConnect Standard.
- 65 • All reference to electronic documents that are received from a data source and stored
66 in an *Agent* shall be referred to as "*Document*(s)" and are typically provided with a
67 prefix identifier; e.g. *Asset Document*.

68 • All references to electronic documents generated by an *Agent* and sent to a client
69 software application shall be referred to as a "*Response Document*".

70 When used with no additional descriptor, the form "document" shall be used to refer to
71 any printed or electronic document.

72 Manufacturing software systems implemented utilizing MTConnect can be represented by
73 a very simple structure as shown in *Figure 1* .

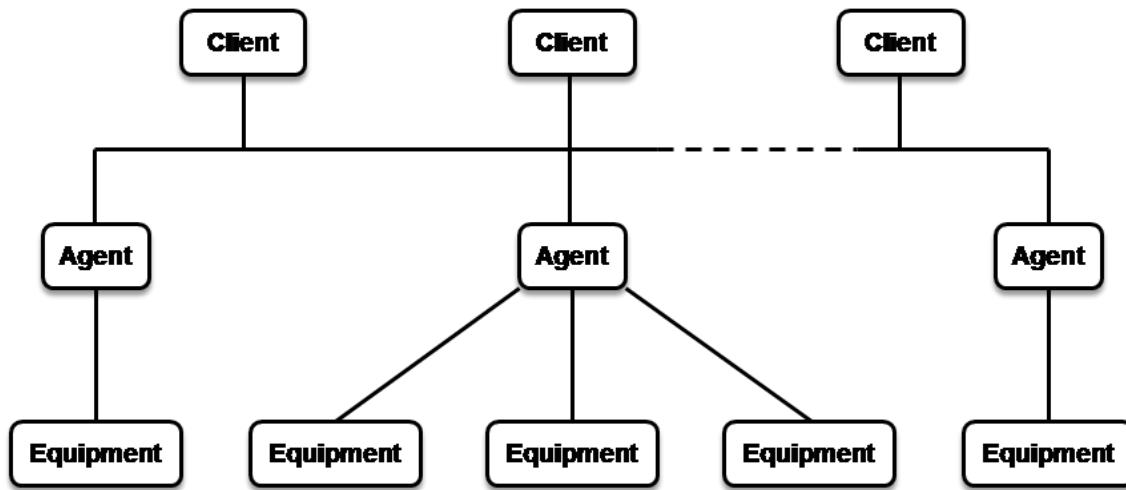


Figure 1: Basic MTConnect Implementation Structure

74 The three basic modules that comprise a software system implemented using MTConnect
75 are:

76 **Equipment:** Any data source. In the MTConnect Standard, equipment is defined as any
77 tangible property that is used to equip the operations of a manufacturing facility. Examples
78 of equipment are machine tools, ovens, sensor units, workstations, software applications,
79 and bar feeders.

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

84 Note: The *Agent* may be fully integrated into the piece of equipment or the *Agent* may be
85 independent of the piece of equipment. Implementation of an *Agent* is the responsibility
86 of the supplier of the piece of equipment and/or the implementer of the *Agent*.

87 **Client Software Application:** Software that requests data from *Agents* and processes
88 that data in support of manufacturing operations.

89 Based on *Figure 1*, it is important to understand that the MTConnect Standard only ad-
 90 dresses the following functionality and behavior of an *Agent*:

- 91 • the method used by a client software application to request information from an
 92 *Agent*.
- 93 • the response that an *Agent* provides to a client software application.
- 94 • a *data dictionary* used to provide consistency in understanding the meaning of data
 95 reported by a data source.
- 96 • the description of the *semantic data models* used to structure *Response Documents*
 97 provided by an *Agent* to a client software application.

98 These functions are the primary building blocks that define the *Base Functional Structure*
 99 of the MTConnect Standard.

100 There are a wide variety of data sources (equipment) and data consumption systems (client
 101 software systems) used in manufacturing operations. There are also many different uses
 102 for the data associated with a manufacturing operation. No single approach to implement-
 103 ing a data communication system can address all data exchange and data management
 104 functions typically required in the data driven manufacturing environment. MTConnect
 105 has been uniquely designed to address this diversity of data types and data usages by pro-
 106 viding different *semantic data models* for different data application requirements:

107 Data Collection: The most common use of data in manufacturing is the collection of
 108 data associated with the production of products and the operation of equipment that pro-
 109 duces those products. The MTConnect Standard provides comprehensive *semantic data*
 110 *models* that represent data collected from manufacturing operations. These *semantic data*
 111 *models* are detailed in *MTConnect Standard: Part 2.0 - Devices Information Model* and
 112 *MTConnect Standard: Part 3.0 - Streams Information Model* of the MTConnect Standard.

113 Inter-operations Between Pieces of Equipment: The MTConnect Standard provides
 114 an *Interaction Model* that structures the information required to allow multiple pieces of
 115 equipment to coordinate actions required to implement manufacturing activities. This
 116 *Interaction Model* is an implementation of a *Request/Response* messaging structure. This
 117 *Interaction Model* is called *Interfaces* which is detailed in *MTConnect Standard: Part*
 118 *5.0 - Interfaces* of the MTConnect Standard.

119 Shared Data: Certain information used in a manufacturing operation is commonly
 120 shared amongst multiple pieces of equipment and/or software applications. This infor-
 121 mation is not typically "owned" by any one manufacturing resource. The MTConnect

122 Standard represents this information through a series of *semantic data models* – each de-
123 scribing different types of information used in the manufacturing environment. Each type
124 of information is called an *MTConnect Asset*. *MTConnect Assets* are detailed in *MTCon-*
125 *nnect Standard: Part 4.0 - Assets Information Model*, and its sub-*Parts*, of the MTConnect
126 Standard.

¹²⁷ 2 Purpose of This Document

128 This document, *MTConnect Standard Part 1.0 - Overview and Fundamentals* of the *MT-*
129 *Connect* Standard, addresses two major topics relating to the MTConnect Standard. The
130 first sections of the document define the organization of the documents used to describe the
131 MTConnect Standard; including the terms and terminology used throughout the Standard.
132 The balance of the document defines the following:

- 133 • Operational concepts describing how an *Agent* should organize and structure data
134 that has been collected from a data source.
- 135 • Definition and structure of the *Response Documents* supplied by an *Agent*.
- 136 • The protocol used by a client software application to communicate with an *Agent*.

¹³⁷ 3 Terminology and Conventions

¹³⁸ 3.1 Glossary

¹³⁹ CDATA

¹⁴⁰ General meaning:

¹⁴¹ An abbreviation for Character Data.

¹⁴² CDATA is used to describe a value (text or data) published as part of an XML element.
¹⁴³

¹⁴⁴ For example, "This is some text" is the CDATA in the XML element:

¹⁴⁵ <Message ...>This is some text</Message>

¹⁴⁶ Appears in the documents in the following form: CDATA

¹⁴⁷ HTTP

¹⁴⁸ Hyper-Text Transport Protocol. The protocol used by all web browsers and web
¹⁴⁹ applications.

¹⁵⁰ Note: HTTP is an IETF standard and is defined in RFC 7230.

¹⁵¹ See <https://tools.ietf.org/html/rfc7230> for more information.

¹⁵² NMTOKEN

¹⁵³ The data type for XML identifiers.

¹⁵⁴ Note: The identifier must start with a letter, an underscore "_" or a colon. The next
¹⁵⁵ character must be a letter, a number, or one of the following ".", "-", "_", ":". The
¹⁵⁶ identifier must not have any spaces or special characters.

¹⁵⁷ Appears in the documents in the following form: NMTOKEN.

¹⁵⁸ REST

¹⁵⁹ Stands for REpresentational State Transfer: A software architecture where a client
¹⁶⁰ software application and server move through a series of state transitions based
¹⁶¹ solely on the request from the client and the response from the server.

¹⁶² Appears in the documents in the following form: REST.

¹⁶³ URI

¹⁶⁴ Stands for Universal Resource Identifier.

¹⁶⁵ See <http://www.w3.org/TR/uri-clarification/#RFC3986>

166 URL

167 Stands for Uniform Resource Locator.

168 See <http://www.w3.org/TR/uri-clarification/#RFC3986>

169 URN

170 Stands for Uniform Resource Name.

171 See <http://www.w3.org/TR/uri-clarification/#RFC3986>

172 UTC/GMT

173 Stands for Coordinated Universal Time/Greenwich Mean Time.

174 UTC/GMT is the primary time standard by which the world regulates clocks and
175 time.

176 The time stamp for all information reported in an *MTConnect Response Document*
177 is provided in UTC/GMT format.

178 UUID

179 General meaning:

180 Stands for Universally Unique Identifier. (Can also be referred to as a GUID in some
181 literature Globally Unique Identifier).

182 Note: Defined in RFC 4122 of the IETF. See <https://www.ietf.org/rfc/rfc4122.txt>
183 for more information.

184 Appears in the documents in the following form: UUID.

185 Used as an attribute for an XML element:

186 Used as an attribute that provides a unique identity for a piece of information re-
187 ported by an *Agent*.

188 Appears in the documents in the following form: `uuid`.

189 W3C

190 Stands for World Wide Web Consortium.

191 W3C is an international community of organizations and the public work together
192 to develop internet standards.

193 W3C Standards are used as a guide within the MTConnect Standard.

194 XML

195 Stands for eXtensible Markup Language.

196 XML defines a set of rules for encoding documents that both a human-readable and
197 machine-readable.

198 XML is the language used for all code examples in the MTConnect Standard.
199 Refer to <http://www.w3.org/XML> for more information about XML.

200 **XPath**

201 General meaning:
202 XPath is a command structure that describes a way for a software system to locate
203 information in an XML document.
204 XPath uses an addressing syntax based on a path through the document's logical
205 structure.
206 See <http://www.w3.org/TR/xpath> for more information on XPath.
207 Appears in the documents in the following form: XPath.

208 **Abstract Element**

209 An element that defines a set of common characteristics that are shared by a group
210 of elements.
211 An abstract element cannot appear in a document. In a specific implementation of
212 a schema, an abstract element is replaced by a derived element that is itself not an
213 abstract element. The characteristics for the derived element are inherited from the
214 abstract element.
215 Appears in the documents in the following form: abstract.

216 **Adapter**

217 An optional piece of hardware or software that transforms information provided by
218 a piece of equipment into a form that can be received by an *Agent*.
219 Appears in the documents in the following form: adapter.

220 **Agent**

221 Refers to an MTConnect Agent.
222 Software that collects data published from one or more piece(s) of equipment, orga-
223 nizes that data in a structured manner, and responds to requests for data from client
224 software systems by providing a structured response in the form of a *Response Doc-*
225 *ument* that is constructed using the *semantic data models* defined in the Standard.
226 Appears in the documents in the following form: Agent.

227 **Application Programming Interface**

228 A set of methods to provide communications between software applications.
229 The API defined in the MTConnect Standard describes the methods for providing
230 the *Request/Response* Information Exchange between an *Agent* and client software
231 applications.

232 Appears in the documents in the following forms: Application Programming Inter-
233 face or API.

234 ***Archetype***

235 General Description of an *MTConnect Asset*:

236 Archetype is a class of *MTConnect Assets* that provides the requirements, con-
237 straints, and common properties for a type of *MTConnect Asset*.

238 Appears in the documents in the following form: Archetype.

239 Used as an XML term describing an *MTConnect Asset*:

240 In an XML representation of the *Asset Information Models*, Archetype is an ab-
241 stract element that is replaced by a specific type of *Asset Archetype*.

242 Appears in the documents in the following form: Archetype

243 ***Asset***

244 General meaning:

245 Typically referred to as an *MTConnect Asset*.

246 An *MTConnect Asset* is something that is used in the manufacturing process, but is
247 not permanently associated with a single piece of equipment, can be removed from
248 the piece of equipment without compromising its function, and can be associated
249 with other pieces of equipment during its lifecycle.

250 Used to identify a storage area in an *Agent*:

251 See description of *buffer*.

252 Used as an *Information Model*:

253 Used to describe an *Information Model* that contains the rules and terminology that
254 describe information that may be included in electronic documents representing *MT-*
255 *Connect Assets*.

256 The *Asset Information Models* defines the structure for the *Assets Response Docu-*
257 *ment*.

258 Individual *Information Models* describe the structure of the *Asset Documents* rep-
259 resent each type of *MTConnect Asset*. Appears in the documents in the following
260 form: *Asset Information Models* or (asset type) *Information Model*.

261 Used when referring to an *MTConnect Asset*:

262 Refers to the information related to an *MTConnect Asset* or a group of *MTConnect*
263 *Assets*.

264 Appears in the documents in the following form: *Asset* or *Assets*.

265 Used as an XML container or element:

- 266 ● When used as an XML container that consists of one or more types of Asset
267 XML elements.

268 Appears in the documents in the following form: Assets.

- 269 ● When used as an abstract XML element. It is replaced in the XML document
270 by types of Asset elements representing individual Asset entities.

271 Appears in the documents in the following form: Asset.

272 Used to describe information stored in an Agent:

273 Identifies an electronic document published by a data source and stored in the *assets*
274 *buffer* of an Agent.

275 Appears in the documents in the following form: Asset Document.

276 Used as an XML representation of an MTConnect Response Document:

277 Identifies an electronic document encoded in XML and published by an Agent in
278 response to a Request for information from a client software application relating to
279 MTConnect Assets.

280 Appears in the documents in the following form: MTConnectAssets.

281 Used as an MTConnect Request:

282 Represents a specific type of communications request between a client software ap-
283 plication and an Agent regarding MTConnect Assets.

284 Appears in the documents in the following form: Asset Request.

285 Used as part of an HTTP Request:

286 Used in the path portion of an HTTP Request Line, by a client software applica-
287 tion, to initiate an Asset Request to an Agent to publish an MTConnectAssets
288 document.

289 Appears in the documents in the following form: asset.

290 **Asset Document**

291 An electronic document published by an Agent in response to a Request for infor-
292 mation from a client software application relating to Assets.

293 **Attribute**

294 A term that is used to provide additional information or properties for an element.

295 Appears in the documents in the following form: attribute.

296 **Base Functional Structure**

297 A consistent set of functionalities defined by the MTConnect Standard. This func-
298 tionality includes the protocol(s) used to communicate data to a client software ap-
299 plication, the *semantic data models* defining how that data is organized into Re-
300 sponse Documents, and the encoding of those Response Documents.

301 Appears in the documents in the following form: *Base Functional Structure*.

302 ***buffer***

303 General meaning:

304 A section of an *Agent* that provides storage for information published from pieces
305 of equipment.

306 Used relative to *Streaming Data*:

307 A section of an *Agent* that provides storage for information relating to individual
308 pieces of *Streaming Data*.

309 Appears in the documents in the following form: *buffer*.

310 Used relative to *MTConnect Assets*:

311 A section of an *Agent* that provides storage for *Asset Documents*.

312 Appears in the documents in the following form: *assets buffer*.

313 ***Child Element***

314 A portion of a data modeling structure that illustrates the relationship between an
315 element and the higher-level *Parent Element* within which it is contained.

316 Appears in the documents in the following form: *Child Element*.

317 ***Client***

318 A process or set of processes that send *Requests* for information to an *Agent*; e.g.
319 software applications or a function that implements the *Request* portion of an *Inter-*
320 *face Interaction Model*.

321 Appears in the documents in the following form: *client*.

322 ***Component***

323 General meaning:

324 A *Structural Element* that represents a physical or logical part or subpart of a piece
325 of equipment.

326 Appears in the documents in the following form: *Component*.

327 Used in *Information Models*:

328 A data modeling element used to organize the data being retrieved from a piece of
329 equipment.

- 330 • When used as an XML container to organize *Lower Level Component* ele-
331 ments.

332 Appears in the documents in the following form: Components.

- 333 ● When used as an abstract XML element. Component is replaced in a data
334 model by a type of *Component* element. Component is also an XML con-
335 tainer used to organize *Lower Level* Component elements, *Data Entities*, or
336 both.

337 Appears in the documents in the following form: Component.

338 ***Composition***

339 General meaning:

340 Data modeling elements that describe the lowest level basic structural or functional
341 building blocks contained within a Component element.

342 Appears in the documents in the following form: *Composition*

343 Used in Information Models:

344 A data modeling element used to organize the data being retrieved from a piece of
345 equipment.

- 346 ● When used as an XML container to organize Composition elements.

347 Appears in the documents in the following form: Compositions

- 348 ● When used as an abstract XML element. Composition is replaced in a data
349 model by a type of *Composition* element.

350 Appears in the documents in the following form: Composition.

351 ***Condition***

352 General meaning:

353 An indicator of the health of a piece of equipment or a Component and its ability to
354 function.

355 Used as a modeling element:

356 A data modeling element used to organize and communicate information relative to
357 the health of a piece of equipment or Component.

358 Appears in the documents in the following form: *Condition*.

359 Used in Information Models:

360 An XML element used to represent Condition elements.

- 361 ● When used as an XML container to organize *Lower Level* Condition ele-
362 ments.

363 Appears in the documents in the following form: Condition.

- 364 ● When used as a *Lower Level* element, the form Condition is an abstract
365 type XML element. This *Lower Level* element is a *Data Entity*. Condition
366 is replaced in a data model by type of *Condition* element.
367 Appears in the documents in the following form: Condition.

368 Note: The form Condition is used to represent both above uses.

369 **Controlled Vocabulary**

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

372 Appears in the documents in the following form: *Controlled Vocabulary*.

373 **Current**

374 General meaning:

375 Meaning 1: A term describing the most recent occurrence of something.

376 Meaning 2: A term used to describe movement; e.g. electric current or air current.

377 Appears in the documents in the following form: current

378 Used in reference to an Agent:

379 A reference to the most recent information available to an *Agent*.

380 Appears in the documents in the following form: current.

381 Used as an MTConnect Request:

382 A specific type of communications request between a client software application and
383 an *Agent* regarding *Streaming Data*.

384 Appears in the documents in the following form: *Current Request*.

385 Used as part of an HTTP Request:

386 Used in the path portion of an *HTTP Request Line*, by a client software applica-
387 tion, to initiate a *Current Request* to an *Agent* to publish an MTConnectStreams
388 document.

389 Appears in the documents in the following form: current.

390 **Current Request**

391 An HTTP request to the *Agent* for returning latest known values for the *DataItem*
392 as an MTConnectStreams XML document

393 **data dictionary**

394 Listing of standardized terms and definitions used in *MTConnect Information Mod-
395 els*.

396 Appears in the documents in the following form: *data dictionary*.

397 ***Data Entity***

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

401 Appears in the documents in the following form: *Data Entity*.

402 ***Data Item***

403 General meaning:

404 Descriptive information or properties and characteristics associated with a *Data En-*
 405 *tity*.

406 Appears in the documents in the following form: data item.

407 Used in an XML representation of a *Data Entity*:

- 408 • When used as an XML container to organize DataItem elements.

409 Appears in the documents in the following form: DataItems.

- 410 • When used to represent a specific *Data Entity*, the form DataItem is an XML
 411 element.

412 Appears in the documents in the following form: DataItem.

413 ***Data Set***

414 A set of *key-value pairs* where each entry is uniquely identified by the *key*.

415 ***Data Source***

416 Any piece of equipment that can produce data that is published to an *Agent*.

417 Appears in the documents in the following form: data source.

418 ***Data Streaming***

419 A method for an *Agent* to provide a continuous stream of information in response to
 420 a single *Request* from a client software application.

421 Appears in the documents in the following form: *Data Streaming*.

422 ***Deprecated***

423 An indication that specific content in an *MTConnect Document* is currently usable
 424 but is regarded as being obsolete or superseded. It is recommended that deprecated
 425 content should be avoided.

426 Appears in the documents in the following form: **DEPRECATED**.

427 ***Deprecation Warning***

428 An indicator that specific content in an *MTConnect Document* may be changed to
429 **DEPRECATED** in a future release of the standard.

430 Appears in the documents in the following form: **DEPRECATION WARNING**.

431 ***Device***

432 A part of an information model representing a piece of equipment.

433 Used in an XML representation of a *Response Document*:

- 434 • When used as an XML container to organize *Device* elements.

435 Appears in the documents in the following form: *Devices*.

- 436 • When used as an XML container to represent a specific piece of equipment and
437 is composed of a set of *Structural Elements* that organize and provide relevance
438 to data published from that piece of equipment.

439 Appears in the documents in the following form: *Device*.

440 ***Devices Information Model***

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

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

444 ***Document***

445 General meaning:

446 A piece of written, printed, or electronic matter that provides information.

447 Used to represent an *MTConnect Document*:

448 Refers to printed or electronic document(s) that represent a *Part(s)* of the MTCon-
449 nect Standard.

450 Appears in the documents in the following form: *MTConnect Document*.

451 Used to represent a specific representation of an *MTConnect Document*:

452 Refers to electronic document(s) associated with an *Agent* that are encoded using
453 *XML; Response Documents or Asset Documents*.

454 Appears in the documents in the following form: *MTConnect XML Document*.

455 Used to describe types of information stored in an *Agent*:

456 In an implementation, the electronic documents that are published from a data source
457 and stored by an *Agent*.

458 Appears in the documents in the following form: *Asset Document*.

459 Used to describe information published by an *Agent*:

460 A document published by an *Agent* based upon one of the *semantic data models*
 461 defined in the MTConnect Standard in response to a request from a client.

462 Appears in the documents in the following form: *Response Document*.

463 **Document Body**

464 The portion of the content of an *MTConnect Response Document* that is defined
 465 by the relative *MTConnect Information Model*. The *Document Body* contains the
 466 *Structural Elements* and *Data Entities* reported in a *Response Document*.

467 Appears in the documents in the following form: *Document Body*.

468 **Document Header**

469 The portion of the content of an *MTConnect Response Document* that provides infor-
 470 mation from an *Agent* defining version information, storage capacity, protocol, and
 471 other information associated with the management of the data stored in or retrieved
 472 from the *Agent*.

473 Appears in the documents in the following form: *Document Header*.

474 **Element**

475 Refers to an XML element.

476 An XML element is a logical portion of an XML document or schema that begins
 477 with a `start-tag` and ends with a corresponding `end-tag`.

478 The information provided between the `start-tag` and `end-tag` may contain
 479 attributes, other elements (sub-elements), and/or CDATA.

480 Note: Also, an XML element may consist of an `empty-element tag`. Refer
 481 to *Appendix B* for more information on element tags.

482 Appears in the documents in the following form: `element`.

483 **Element Name**

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

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

487 Used to describe the name for a specific XML element:

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

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

491 ***Equipment***

492 Represents anything that can publish information and is used in the operations of a
493 manufacturing facility shop floor. Examples of equipment are machine tools, ovens,
494 sensor units, workstations, software applications, and bar feeders.

495 Appears in the documents in the following form: equipment or piece of equipment.

496 ***Equipment Metadata***

497 See *Metadata*

498 ***Error Information Model***

499 The rules and terminology that describes the *Response Document* returned by an
500 *Agent* when it encounters an error while interpreting a *Request* for information from
501 a client software application or when an *Agent* experiences an error while publishing
502 the *Response* to a *Request* for information.

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

504 ***Event***

505 General meaning:

506 The occurrence of something that happens or takes place.

507 Appears in the documents in the following form: event.

508 Used as a type of *Data Entity*:

509 An identification that represents a change in state of information associated with a
510 piece of equipment or an occurrence of an action. Event also provides a means to
511 publish a message from a piece of equipment.

512 Appears in the documents in the following form: *Event*.

513 Used as a category attribute for a *Data Entity*:

514 Used as a value for the `category` attribute for an XML `DataItem` element.

515 Appears in the documents in the following form: EVENT.

516 Used as an XML container or element:

- 517 • When used as an XML container that consists of one or more types of *Event*
518 XML elements.

519 Appears in the documents in the following form: Events.

- 520 • When used as an abstract XML element. It is replaced in the XML document
521 by types of *Event* elements.

522 Appears in the documents in the following form: Event.

523 ***Extensible***

524 The ability for an implementer to extend *MTConnect Information Models* by adding
525 content not currently addressed in the MTConnect Standard.

526 ***Fault State***

527 In the MTConnect Standard, a term that indicates the reported status of a *Condition*
528 category *Data Entity*.

529 Appears in the documents in the following form: *Fault State*.

530 ***heartbeat***

531 General meaning:

532 A function that indicates to a client application that the communications connection
533 to an *Agent* is still viable during times when there is no new data available to report
534 often referred to as a "keep alive" message.

535 Appears in the documents in the following form: *heartbeat*.

536 When used as part of an *HTTP Request*:

537 The form *heartbeat* is used as a parameter in the query portion of an *HTTP*
538 *Request Line*.

539 Appears in the documents in the following form: *heartbeat*.

540 ***Higher Level***

541 A nested element that is above a lower level element.

542 ***HTTP Error Message***

543 In the MTConnect Standard, a response provided by an *Agent* indicating that an
544 *HTTP Request* is incorrectly formatted or identifies that the requested data is not
545 available from the *Agent*.

546 Appears in the documents in the following form: *HTTP Error Message*.

547 ***HTTP Header***

548 In the MTConnect Standard, the content of the *Header* portion of either an *HTTP*
549 *Request* from a client software application or an *HTTP Response* from an *Agent*.

550 Appears in the documents in the following form: *HTTP Header*.

551 ***HTTP Method***

552 In the MTConnect Standard, a portion of a command in an *HTTP Request* that indicates
553 the desired action to be performed on the identified resource; often referred to
554 as verbs.

555 ***HTTP Request***

556 In the MTConnect Standard, a communications command issued by a client soft-
557 ware application to an *Agent* requesting information defined in the *HTTP Request*
558 *Line*.

559 Appears in the documents in the following form: *HTTP Request*.

560 ***HTTP Request Line***

561 In the MTConnect Standard, the first line of an *HTTP Request* describing a specific
562 *Response Document* to be published by an *Agent*.

563 Appears in the documents in the following form: *HTTP Request Line*.

564 ***HTTP Response***

565 In the MTConnect Standard, the information published from an *Agent* in reply to
566 an *HTTP Request*. An *HTTP Response* may be either a *Response Document* or an
567 *HTTP Error Message*.

568 Appears in the documents in the following form: *HTTP Response*.

569 ***HTTP Server***

570 In the MTConnect Standard, a software program that accepts *HTTP Requests* from
571 client software applications and publishes *HTTP Responses* as a reply to those *Re-*
572 *quests*.

573 Appears in the documents in the following form: *HTTP Server*.

574 ***HTTP Status Code***

575 In the MTConnect Standard, a numeric code contained in an *HTTP Response* that
576 defines a status category associated with the *Response* either a success status or a
577 category of an HTTP error.

578 Appears in the documents in the following form: *HTTP Status Code*.

579 ***id***

580 **General meaning:**

581 An identifier used to distinguish a piece of information.

582 Appears in the documents in the following form: *id*.

583 **Used as an XML attribute:**

584 When used as an attribute for an XML element - *Structural Element*, *Data Entity*, or
585 *Asset*. *id* provides a unique identity for the element within an XML document.

586 Appears in the documents in the following form: *id*.

587 ***Implementation***

588 A specific instantiation of the MTConnect Standard.

589 ***Information Model***

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

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

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

596 ***instance***

597 Describes a set of *Streaming Data* in an *Agent*. Each time an *Agent* is restarted with
598 an empty *buffer*, data placed in the *buffer* represents a new *instance* of the *Agent*.

599 Appears in the documents in the following form: *instance*.

600 ***Interaction Model***

601 The definition of information exchanged to support the interactions between pieces
602 of equipment collaborating to complete a task.

603 Appears in the documents in the following form: *Interaction Model*.

604 ***Interface***

605 General meaning:

606 The exchange of information between pieces of equipment and/or software systems.

607 Appears in the documents in the following form: *interface*.

608 Used as an *Interaction Model*:

609 An *Interaction Model* that describes a method for inter-operations between pieces
610 of equipment.

611 Appears in the documents in the following form: *Interface*.

612 Used as an XML container or element:

613 - When used as an XML container that consists of one or more types of *Interface*
614 XML elements.

615 Appears in the documents in the following form: *Interfaces*.

616 - When used as an abstract XML element. It is replaced in the XML document
617 by types of *Interface* elements.

618 Appears in the documents in the following form: *Interface*

619 **key**

620 A unique identifier in a *key-value pair* association.

621 **key-value pair**

622 An association between an identifier referred to as the *key* and a value which taken
623 together create a *key-value pair*. When used in a set of *key-value pairs* each *key* is
624 unique and will only have one value associated with it at any point in time.

625 **Lower Level**

626 A nested element that is below a higher level element.

627 **Message**

628 General meaning:

629 The content of a communication process.

630 Appears in the documents in the following form: *message*.

631 Used relative to an Agent:

632 Describes the information that is exchanged between an *Agent* and a client soft-
633 ware application. A *Message* may contain either a *Request* from a client software
634 application or a *Response* from an *Agent*.

635 Appears in the documents in the following form: *Message*.

636 Used as a type of Data Entity:

637 Describes a type of *Data Entity* in the *Devices Information Model* that can contain
638 any text string of information or native code to be transferred from a piece of equip-
639 ment.

640 Appears in the documents in the following form: MESSAGE.

641 Used as an Element Name:

642 An *Element Name* for a *Data Entity* in the *Streams Information Model* that can
643 contain any text string of information or native code to be transferred from a piece
644 of equipment.

645 Appears in the documents in the following form: Message.

646 **Metadata**

647 Data that provides information about other data.

648 For example, *Equipment Metadata* defines both the *Structural Elements* that rep-
649 resent the physical and logical parts and sub-parts of each piece of equipment, the
650 relationships between those parts and sub-parts, and the definitions of the *Data En-*
651 *tities* associated with that piece of equipment.

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

- 653 ***MTConnect Agent***
654 See definition for *Agent*.
- 655 ***MTConnect Document***
656 See *Document*.
- 657 ***MTConnect Request***
658 A communication request for information issued from a client software application
659 to an *Agent*.
660 Appears in the documents in the following form: *MTConnect Request*.
- 661 ***MTConnect XML Document***
662 See *Document*.
- 663 ***MTConnectAssets Response Document***
664 An electronic document published by an *Agent* in response to a *Request* for infor-
665 mation from a client software application relating to *MTConnect Assets*.
666 Appears in the documents in the following form: *MTConnectAssets Response Doc-
667 ument*.
- 668 ***MTConnectDevices Response Document***
669 An electronic document published by an *Agent* in response to a *Request* for infor-
670 mation from a client software application that includes *Metadata* for one or more
671 pieces of equipment.
672 Appears in the documents in the following form: *MTConnectDevices Response
673 Document*.
- 674 ***MTConnectErrors Response Document***
675 An electronic document published by an *Agent* whenever it encounters an error
676 while interpreting a *Request* for information from a client software application or
677 when an *Agent* experiences an error while publishing the *Response* to a *Request* for
678 information.
679 Appears in the documents in the following form: *MTConnectErrors Response Doc-
680 ument*.
- 681 ***MTConnectStreams Response Document***
682 An electronic document published by an *Agent* in response to a *Request* for infor-
683 mation from a client software application that includes *Streaming Data* from the
684 *Agent*.
685 Appears in the documents in the following form: *MTConnectStreams Response
686 Document*.

687 **parameter**

688 General Meaning:

689 A variable that must be given a value during the execution of a program or a com-
690 munications command.

691 When used as part of an *HTTP Request*:

692 Represents the content (keys and associated values) provided in the *Query* portion
693 of an *HTTP Request Line* that identifies specific information to be returned in a
694 *Response Document*.

695 Appears in the documents in the following form: parameter.

696 **Parent Element**

697 An XML element used to organize *Lower Level* child elements that share a common
698 relationship to the *Parent Element*.

699 Appears in the documents in the following form: Parent Element.

700 **Persistence**

701 A method for retaining or restoring information.

702 **Probe**

703 General meaning of a physical entity:

704 An instrument commonly used for measuring the physical geometrical characteris-
705 tics of an object.

- 706 • Used to describe a measurement device:

707 The form probe is used to define a measurement device that provides position
708 information.

709 Appears in the documents in the following form: probe.

- 710 • Used within a *Data Entity*:

711 The form PROBE is used to designate a subtype for the *Data Entity* PATH_-
712 POSITION indicating a measurement position relating to a probe unit.

713 Appears in the documents in the following form: PROBE.

- 714 General meaning for communications with an *Agent*:

715 Probe is used to define a type of communication request.

- 716 • Used as a type of communication request:

717 The form *Probe Request* represents a specific type of communications request
718 between a client software application and an *Agent* regarding *Metadata* for one
719 or more pieces of equipment.

720 Appears in the documents in the following form: Probe Request.

721 ● Used in an *HTTP Request Line*:

722 The form *probe* is used to designate a *Probe Request* in the <Path> portion
723 of an *HTTP Request Line*.

724 Appears in the documents in the following form: *probe*.

725 **Protocol**

726 A set of rules that allow two or more entities to transmit information from one to the
727 other.

728 **Publish/Subscribe**

729 In the MTConnect Standard, a communications messaging pattern that may be used
730 to publish *Streaming Data* from an *Agent*. When a *Publish/Subscribe* communi-
731 cation method is established between a client software application and an *Agent*,
732 the *Agent* will repeatedly publish a specific MTConnect Streams document at a
733 defined period.

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

735 **Query**

736 General Meaning:

737 A portion of a request for information that more precisely defines the specific infor-
738 mation to be published in response to the request.

739 Appears in the documents in the following form: *Query*.

740 Used in an *HTTP Request Line*:

741 The form *query* includes a string of parameters that define filters used to refine the
742 content of a *Response Document* published in response to an *HTTP Request*.

743 Appears in the documents in the following form: *query*.

744 **Request**

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

747 Appears in the documents in the following form: *Request*.

748 **Request/Response**

749 A communications pattern that supports the transfer of information between an
750 *Agent* and a client software application. In a *Request/Response* information ex-
751 change, a client software application requests specific information from an *Agent*.
752 An *Agent* responds to the *Request* by publishing a *Response Document*.

753 Appears in the documents in the following form: *Request/Response*.

754 ***Requester***

755 An entity that initiates a *Request* for information in a communications exchange.

756 Appears in the documents in the following form: *Requester*.

757 ***reset***

758 A reset is associated with an occurrence of a *Data Entity* indicated by the
759 `resetTriggered` attribute. When a reset occurs, the accumulated value or statistic
760 are reverted back to their initial value. A *Data Entity* with a *Data Set* representa-
761 tion removes all *key-value pairs*, setting the *Data Set* to an empty set.

762 ***Responder***

763 An entity that responds to a *Request* for information in a communications exchange.

764 Appears in the documents in the following form: *Responder*.

765 ***Response Document***

766 See *Document*.

767 ***Root Element***

768 The first *Structural Element* provided in a *Response Document* encoded using XML.
769 The *Root Element* is an XML container and is the *Parent Element* for all other XML
770 elements in the document. The *Root Element* appears immediately following the
771 XML Declaration.

772 Appears in the documents in the following form: *Root Element*.

773 ***Sample***

774 General meaning:

775 The collection of one or more pieces of information.

776 Used when referring to the collection of information:

777 When referring to the collection of a piece of information from a data source.

778 Appears in the documents in the following form: sample.

779 Used as an *MTConnect Request*:

780 When representing a specific type of communications request between a client soft-
781 ware application and an *Agent* regarding *Streaming Data*.

782 Appears in the documents in the following form: *Sample Request*.

783 Used as part of an *HTTP Request*:

784 Used in the path portion of an *HTTP Request Line*, by a client software applica-
785 tion, to initiate a *Sample Request* to an *Agent* to publish an *MTConnect Streams*
786 document.

787 Appears in the documents in the following form: `sample`.
788 Used to describe a *Data Entity*:
789 Used to define a specific type of *Data Entity*. A *Sample* type *Data Entity* reports the
790 value for a continuously variable or analog piece of information.
791 Appears in the documents in the following form: *Sample* or *Samples*.
792 Used as an XML container or element:
793 ● When used as an XML container that consists of one or more types of *Sample*
794 XML elements.
795 Appears in the documents in the following form: *Samples*.
796 ● When used as an abstract XML element. It is replaced in the XML document
797 by types of *Sample* elements representing individual *Sample* type of *Data*
798 *Entity*.
799 Appears in the documents in the following form: *Sample*.

800 ***Sample Request***

801 A request from the *Agent* for a stream of time series data.

802 ***schema***

803 General meaning:

804 The definition of the structure, rules, and vocabularies used to define the information
805 published in an electronic document.

806 Appears in the documents in the following form: *schema*.

807 Used in association with an *MTConnect Response Document*:

808 Identifies a specific schema defined for an *MTConnect Response Document*.

809 Appears in the documents in the following form: *schema*.

810 ***semantic data model***

811 A methodology for defining the structure and meaning for data in a specific logical
812 way.

813 It provides the rules for encoding electronic information such that it can be inter-
814 preted by a software system.

815 Appears in the documents in the following form: *semantic data model*.

816 ***sequence number***

817 The primary key identifier used to manage and locate a specific piece of *Streaming*
818 *Data* in an *Agent*.

819 *sequence number* is a monotonically increasing number within an instance of an
820 *Agent*.

821 Appears in the documents in the following form: *sequence number*.

822 **Standard**

823 General meaning:

824 A document established by consensus that provides rules, guidelines, or character-
825 istics for activities or their results (as defined in ISO/IEC Guide 2:2004).

826 Used when referring to the MTConnect Standard:

827 The MTConnect Standard is a standard that provides the definition and semantic
828 data structure for information published by pieces of equipment.

829 Appears in the documents in the following form: Standard or MTConnect Standard.

830 **Streaming Data**

831 The values published by a piece of equipment for the *Data Entities* defined by the
832 *Equipment Metadata*.

833 Appears in the documents in the following form: *Streaming Data*.

834 **Streams Information Model**

835 The rules and terminology (*semantic data model*) that describes the *Streaming Data*
836 returned by an *Agent* from a piece of equipment in response to a *Sample Request* or
837 a *Current Request*.

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

839 **Structural Element**

840 General meaning:

841 An XML element that organizes information that represents the physical and logical
842 parts and sub-parts of a piece of equipment.

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

844 Used to indicate hierarchy of Components:

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

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

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

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

851 ***subtype***

852 General meaning:

853 A secondary or subordinate type of categorization or classification of information.

854 In software and data modeling, a subtype is a type of data that is related to another
855 higher-level type of data.

856 Appears in the documents in the following form: subtype.

857 Used as an attribute for a *Data Entity*:

858 Used as an attribute that provides a sub-categorization for the type attribute for a
859 piece of information.

860 Appears in the documents in the following form: subType.

861 ***time stamp***

862 General meaning:

863 The best available estimate of the time that the value(s) for published or recorded
864 information was measured or determined.

865 Appears in the documents as "time stamp".

866 Used as an attribute for recorded or published data:

867 An attribute that identifies the time associated with a *Data Entity* as stored in an
868 Agent.

869 Appears in the documents in the following form: timestamp.

870 ***Top Level***

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

873 ***type***

874 General meaning:

875 A classification or categorization of information.

876 In software and data modeling, a type is a grouping function to identify pieces of
877 information that share common characteristics.

878 Appears in the documents in the following form: type.

879 Used as an attribute for a *Data Entity*:

880 Used as an attribute that provides a categorization for piece of information that share
881 common characteristics.

882 Appears in the documents in the following form: type.

883 **Valid Data Value**

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

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

887 **WARNING**

888 General Meaning:

889 A statement or action that indicates a possible danger, problem, or other unexpected
890 situation.

891 Used relative to changes in an *MTConnect Document*:

892 Used to indicate that specific content in an *MTConnect Document* may be changed
893 in a future release of the standard.

894 Appears in the documents in the following form: **WARNING**.

895 Used as a *Valid Data Value* for a *Condition*:

896 Used as a *Valid Data Value* for a *Condition* type *Data Entity*.

897 Appears in the documents in the following form: **WARNING**.

898 Used as an *Element Name* for a *Data Entity*:

899 Used as the *Element Name* for a *Condition* type *Data Entity* in an *MTConnect-*
900 *Streams Response Document*.

901 Appears in the documents in the following form: **Warning**.

902 **XML Container**

903 In the MTConnect Standard, a type of XML element.

904 An XML container is used to organize other XML elements that are logically related
905 to each other. A container may have either *Data Entities* or other *Structural Elements*
906 as *Child Elements*.

907 **XML Document**

908 An XML document is a structured text file encoded using XML.

909 An XML document is an instantiation of an XML schema. It has a single root XML
910 element, conforms to the XML specification, and is structured based upon a specific
911 schema.

912 *MTConnect Response Documents* may be encoded as an XML document.

913 **XML Schema**

914 In the MTConnect Standard, an instantiation of a schema defining a specific docu-
915 ment encoded in XML.

916 3.2 MTConnect References

- 917 [MTConnect Part 1.0] *MTConnect Standard Part 1.0 - Overview and Fundamentals*. Version 1.5.0.
- 918
- 919 [MTConnect Part 2.0] *MTConnect Standard: Part 2.0 - Devices Information Model*. Version 1.5.0.
- 920
- 921 [MTConnect Part 3.0] *MTConnect Standard: Part 3.0 - Streams Information Model*. Version 1.5.0.
- 922
- 923 [MTConnect Part 4.0] *MTConnect Standard: Part 4.0 - Assets Information Model*. Version 1.5.0.
- 924
- 925 [MTConnect Part 5.0] *MTConnect Standard: Part 5.0 - Interfaces*. Version 1.5.0.

926 4 MTConnect Standard

927 The MTConnect Standard is organized in a series of documents (also referred to as MT-
 928 Connect Documents) that each address a specific set of requirements defined by the Stan-
 929 dard. Each MTConnect Document will be referred to as a *Part* of the Standard; e.g.,
 930 *MTConnect Standard Part 1.0 - Overview and Fundamentals*. Together, these documents
 931 describe the *Base Functional Structure* specified in the MTConnect Standard.

932 Implementation of any manufacturing data management system may utilize information
 933 from any number of these documents. However, it is not necessary to realize all informa-
 934 tion contained in these documents for any one specific implementation.

935 4.1 MTConnect Documents Organization

936 The MTConnect specification is organized into the following documents:

937 *MTConnect Standard Part 1.0 - Overview and Fundamentals*: Provides an overview of
 938 the MTConnect Standard and defines the terminology and structure used throughout all
 939 documents associated with the Standard. Additionally, [MTConnect Part 1.0] describes
 940 the functions provided by an *Agent* and the protocol used to communicate with an *Agent*.

941 *MTConnect Standard: Part 2.0 - Devices Information Model*: Defines the *semantic data*
 942 *model* that describes the data that can be supplied by a piece of equipment. This model
 943 details the XML elements used to describe the structural and logical configuration for a
 944 piece of equipment. It also describes each type of data that may be supplied by a piece of
 945 equipment in a manufacturing operation.

946 *MTConnect Standard: Part 3.0 - Streams Information Model*: Defines the *semantic data*
 947 *model* that organizes the data that is collected from a piece of equipment and transferred
 948 to a client software application from an *Agent*.

949 *MTConnect Standard: Part 4.0 - Assets Information Model*: Provides an overview of *MT-*
 950 *Connect Assets* and the functions provided by an *Agent* to communicate information relat-
 951 ing to *Assets*. The various *semantic data models* describing each type of *MTConnect Asset*
 952 are defined in sub-*Part* documents (*Part 4.x*) of the MTConnect Standard.

953 *MTConnect Standard: Part 5.0 - Interfaces*: Defines the MTConnect implementation of
 954 the *Interaction Model* used to coordinate actions between pieces of equipment used in
 955 manufacturing systems.

956 4.2 MTConnect Document Versioning

957 The MTConnect Standard will be periodically updated with new and expanded function-
958 ality. Each new release of the Standard will include additional content adding new func-
959 tionality and/or extensions to the *semantic data models* defined in the Standard.

960 The MTConnect Standard uses a three-digit version numbering system to identify each
961 release of the Standard that indicates the progression of enhancements to the Standard. The
962 format used to identify the documents in a specific version of the MTConnect Standard is:

963 *major.minor.revision*

964 *major* – Identifier representing a consistent set of functionalities defined by the MTCon-
965 nect Standard. This functionality includes the protocol(s) used to communicate data to a
966 client software application, the *semantic data models* defining how that data is organized
967 into *Response Documents*, and the encoding of those *Response Documents*. This set of
968 functionalities is referred to as the *Base Functional Structure*.

969 When a release of the MTConnect Standard removes or modifies any of the protocol(s),
970 *semantic data models*, or encoding of the *Response Documents* included in the *Base Func-*
971 *tional Structure* in such a way that it breaks backward compatibility and a client software
972 application can no longer communicate with an *Agent* or cannot interpret the information
973 provided by an *Agent*, the *major* version identifier for the Documents in the release is
974 revised to a successively higher number.

975 See *Section 4.5 - Backwards Compatibility* for details regarding the interaction between a
976 client software application and versions of the MTConnect Standard.

977 *minor* – Identifier representing a specific set of functionalities defined by the MTConnect
978 Standard. Each release of the Standard (with a common *major* version identifier) includes
979 new and/or expanded functionality – protocol extensions, new or extended *semantic data*
980 *models*, and/or new programming languages. Each of these releases of the Standard is
981 indicated by a successively higher *minor* version identifier.

982 If a new *major* version of the MTConnect Standard is released, the *minor* version identifier
983 will be reset to 0.

984 *revision* – A supplemental identifier representing only organizational or editorial changes
985 to a *minor* version document with no changes in the functionality described in that docu-
986 ment.

987 New releases of a specific document are indicated by a successively higher revision version
988 identifier.

989 If a new *minor* version of a document is released, the *revision* identifier will be reset to 0.

990 An example of the version identifier for a specific document would be:

Version M.N.R

991 4.2.1 Document Releases

992 A *major* revision change represents a substantial change to the MTConnect Standard. At
993 the time of a *major* revision change, all documents representing the MTConnect Standard
994 will be updated and released together.

995 A *minor* revision change represents some level of extended functionality supported by the
996 MTConnect Standard. At the time of a *minor* version release, MTConnect Documents
997 representing the changes or enhancements to the Standard will be updated as required.
998 However, all documents, whether updated or not, will be released together with a new
999 *minor* version number. Providing all documents at a common *major* and *minor* version
1000 makes it easier for implementers to manage the compatibility and upgrade of the different
1001 software tools incorporated into a manufacturing software system.

1002 Since a *revision* represents no functional changes to the MTConnect Standard and includes
1003 only editorial or descriptive changes that enhance the understanding of the functionality
1004 supported by the Standard, individual documents within the Standard may be released
1005 at any time with a new *revision* and that release does not impact any other documents
1006 associated with the MTConnect Standard.

1007 The latest released version of each document provided for the MTConnect Standard, and
1008 historical releases of those documents, are provided at <http://www.mtconnect.org>.

1009 4.3 MTConnect Document Naming Conventions

1010 MTConnect Documents are identified as follows:

1011 4.3.1 Document Title

1012 Each MTConnect Document **MUST** be identified as follows:

MTConnect® Standard

Part #.# - *Title*

Version M.N.R.

1013 The following keys are used to distinguish different *Parts* of the MTConnect Standard and
1014 the version of the MTConnect Document:

1015 #.# – Identifier of the specific Part and sub-*Part* of the MTConnect Standard

1016 Title – Description of the type of information contained in the MTConnect Document

1017 M – Indicator of the *major* version of the MTConnect Document

1018 N – Indicator of the *minor* version of the MTConnect Document

1019 R – Indicator of the revision of the MTConnect Document

1020 For example, a release of *MTConnect Standard: Part 2.0 - Devices Information Model*
1021 would be:

MTConnect® Standard

Part 2.0 - *Devices Information Model*

Version 1.2.0

1022 4.3.2 Electronic Document File Naming

1023 Electronic versions of the MTConnect Documents will be provided in PDF format and
1024 follow this naming convention:

1025 MTC_Part#-#_Title_M-N-R.pdf

1026 The electronic version of the same release of *MTConnect Standard: Part 2.0 - Devices*
1027 *Information Model* would be:

1028 MTC_Part_2-0_Devices_Information_Model_1-2-0.pdf

1029 4.4 Document Conventions

1030 Additional information regarding specific content in the MTConnect Standard is provided
1031 in the sections below.

1032 4.4.1 Use of MUST, SHOULD, and MAY

1033 These words convey specific meaning in the MTConnect Standard when presented in cap-
1034 ital letters, Times New Roman font, and a Bold font style.

- 1035 • The word **MUST** indicates content that is mandatory to be provided in an imple-
1036 mentation where indicated.
- 1037 • The word **SHOULD** indicates content that is recommended, but the exclusion of
1038 which will not invalidate an implementation.
- 1039 • The word **MAY** indicates content that is optional. It is up to the implementer to
1040 decide if the content is relevant to an implementation.
- 1041 • The word **NOT** may be added to the words **MUST** or **SHOULD** to negate the re-
1042 quirement.

1043 4.4.2 Text Conventions

1044 The following conventions will be used throughout the MTConnect Documents to provide
1045 a clear and consistent understanding of the use of each type of information used to define
1046 the MTConnect Standard.

1047 These conventions are:

- 1048 • Standard text is provided in Times New Roman font.

- 1049 • References to documents, sections or sub-sections of a document, or figures within a
 1050 document are *italicized*; e.g., *MTConnect Standard: Part 2.0 - Devices Information*
 1051 *Model*.
- 1052 • Terms with a specific meaning in the MTConnect Standard will be *italicized*; e.g.,
 1053 *major* indicating a version of the Standard.
- 1054 • When these same terms are used within the text without specific reference to their
 1055 function within the MTConnect Standard, they will be provided as non-italicized
 1056 font; e.g., *major* indicating a descriptor of another term.
- 1057 • Terms representing content of an MTConnect *semantic data model* or the protocol
 1058 used in MTConnect will be provided in fixed size, Courier New font; e.g., *compo-*
 1059 *nent*, *probe*, *current*.
- 1060 When these same terms are used within the text without specific reference to
 1061 their function within the MTConnect Standard, they will be provided as Times New
 1062 Roman font.
- 1063 • All *Valid Data Values* that are restricted to a limited or controlled vocabulary will be
 1064 provided in upper case Courier New font with an *_*(underscore) separating words.
 1065 For example: ON, OFF, ACTUAL, COUNTER_CLOCKWISE, etc.
- 1066 • All descriptive attributes associated with each piece of data defined in a *Response*
 1067 *Document* will be provided in Courier New font and camel case font style. For
 1068 example: nativeUnits.

1069 4.4.3 Code Line Syntax and Conventions

1070 The following conventions will be used throughout the MTConnect Documents to describe
 1071 examples of software code produced by an *Agent* or commands provided to an *Agent* from
 1072 a client software application.

1073 All examples are provided in fixed size Courier New font with line numbers.

1074 These conventions are:

- 1075 • XML Code examples:

Example 1: XML Code Examples

```
1076   1 <MTConnectStreams xmlns:m="urn:mtconnect.com:
1077   2   MTConnectStreams:1.1" xmlns:xsi=
1078   3   "http://www.w3.org/2001/XMLSchema-instance"
1079   4   xmlns="urn:mtconnect.com:MTConnectStreams:1.1"
```

1080 • HTTP URL examples:

1081 – `http://<authority>/<path>[?<query>]` When a portion of a URL is enclosed in
 1082 angle brackets ("<" and ">"), that section of the URL is a place holder for
 1083 specific information that will replace the term between the angle brackets.

1084 Note: The angle brackets in a URL do not relate to the angle brackets
 1085 used as the tag elements in an XML example.

1086 – A portion of a URL that is enclosed in square brackets "[" and "]" indicates
 1087 that the enclosed content is optional.

1088 – All other characters in the URL are literal.

1089 4.4.4 Semantic Data Model Content

1090 For each of the *semantic data models* defined in the MTConnect Standard, there are tables
 1091 describing pieces of information provided in the data models. Each table has a column
 1092 labeled *Occurrence*. *Occurrence* defines the number of times the content defined in the
 1093 tables **MAY** be provided in the usage case specified.

1094 • If the *Occurrence* is 1, the content **MUST** be provided.

1095 • If the *Occurrence* is 0..1, the content **MAY** be provided and if provided, at most,
 1096 only one occurrence of the content **MUST** be provided.

1097 • If the *Occurrence* is 0..*, the content **MAY** be provided and any number of occur-
 1098 rences of the content **MAY** be provided.

1099 • If the *Occurrence* is 1..*, one or more occurrences of the content **MUST** be pro-
 1100 vided.

1101 • If the *Occurrence* is a number, e.g., 2, exactly that number of occurrences of the
 1102 content **MUST** be provided.

1103 Note: "*" indicates multiple number of occurrences and is represented by ∞ in the
 1104 figures.

1105 4.4.5 Referenced Standards and Specifications

1106 Other standards and specifications may be used to describe aspects of the protocol, *data*
 1107 *dictionary*, or *semantic data models* defined in the MTConnect Standard. When a spe-

1108 cific standard or specification is referenced in the MTConnect Standard, the name of the
1109 standard or specification will be provided in *italicized* font.

1110 See *Section 3 - Terminology and Conventions*: Bibliography for a complete listing of
1111 standards and specifications used or referenced in the MTConnect Standard.

1112 4.4.6 Deprecation and Deprecation Warnings

1113 When the MTConnect Institute adds new functionality to the MTConnect Standard, the
1114 new content may supersede some of the functionality of existing content or significantly
1115 enhance one of the *semantic data models*. When this occurs, existing content may no
1116 longer be valid for use in the new version of the Standard.

1117 4.4.6.1 Deprecation

1118 In cases when new content supersedes the functionality of the existing content, the original
1119 content **MUST** no longer be included in future implementations – only the new content
1120 should be used.

1121 The superseded content is identified by striking through the original content (*original*
1122 *content*) and marking the content with the words "**DEPRECATED** in Version *M.N*".

1123 The deprecated content must remain in all future *minor* versions of the document. The
1124 content may be removed when a *major* version update is released. This provides imple-
1125 menters guidance on how to interpret data that may be provided from equipment utilizing
1126 an older version of the Standard. This content provides the information required for imple-
1127 menters to develop software applications that support backwards compatibility with older
1128 versions of the standard.

1129 A software application may be designed to be compliant with any specific *minor* version
1130 of the standard. That software application may be collecting data from many different
1131 pieces of equipment. Each of these pieces of equipment may be providing data defined
1132 by the current version or any of the previous *minor* versions of the standard. To maintain
1133 compatibility with existing pieces of equipment, software applications should be imple-
1134 mented to interpret data defined in the current release of the MTConnect Standard, as well
1135 as all deprecated content associated with earlier versions of the Standard.

1136 4.4.6.2 Deprecation Warning

1137 When new content provides improved alternatives for defining the *semantic data mod-*

1138 *els*, the MTConnect Institute may determine that the original content could possibly be
1139 deprecated in the future. When this occurs, a content will be marked with the words
1140 **"DEPRECATION WARNING "** to identify the content that may be deprecated in the
1141 future. This provides advanced notice to implementers that they should choose to utilize
1142 the improved alternatives when developing new products or software systems to avoid the
1143 possibility that the original content may be deprecated in a future version of the Standard.

1144 4.5 Backwards Compatibility

1145 MTConnect Documents with a different *major* version identifier represent a significant
1146 change in the *Base Functional Structure* of the MTConnect Standard. This means that
1147 the schema or protocol defined by the Standard may have changed in ways that will re-
1148 quire software applications to change how they request and/or interpret data received from
1149 an *Agent*. Software applications should be fully version aware since no assumption of
1150 backwards compatibility should be assumed at the time of a *major* revision change to the
1151 MTConnect Standard.

1152 The MTConnect Institute strives to maintain version compatibility through all *minor* re-
1153 visions of the MTConnect Standard. New *minor* versions may introduce extensions to
1154 existing *semantic data models*, extend the protocol used to communicate to the *Agent*,
1155 and/or add new *semantic data models* to extend the functionality of the Standard. Client
1156 software applications may be designed to be compliant with any specific *minor* version
1157 of the MTConnect Standard. Additionally, software applications should be capable of in-
1158 terpreting information from an *Agent* providing data based upon a lower *minor* version
1159 identifier. It should also be capable of interpreting information from an *Agent* providing
1160 data based upon a higher *minor* version identifier of the MTConnect Standard than the
1161 version supported by the client, even though the client may ignore or not be capable of
1162 interpreting the extended content provided by the *Agent*.

1163 A *revision* version of any MTConnect Document provides only editorial changes requiring
1164 no changes to an *Agent* or a client application.

1165 5 MTConnect Fundamentals

1166 The MTConnect Standard defines the functionality of an *Agent*. In an MTConnect instal-
 1167 lation, pieces of equipment publish information to an *Agent*. Client software applications
 1168 request information from the *Agent* using a communications protocol. Based on the spe-
 1169 cific information that the client software application has requested from the *Agent*, the
 1170 *Agent* forms a *Response Document* based upon one of the *semantic data models* defined
 1171 in the MTConnect Standard and then transmits that document to the client software appli-
 1172 cation.

1173 *Figure 2* illustrates the architecture of a typical MTConnect installation.

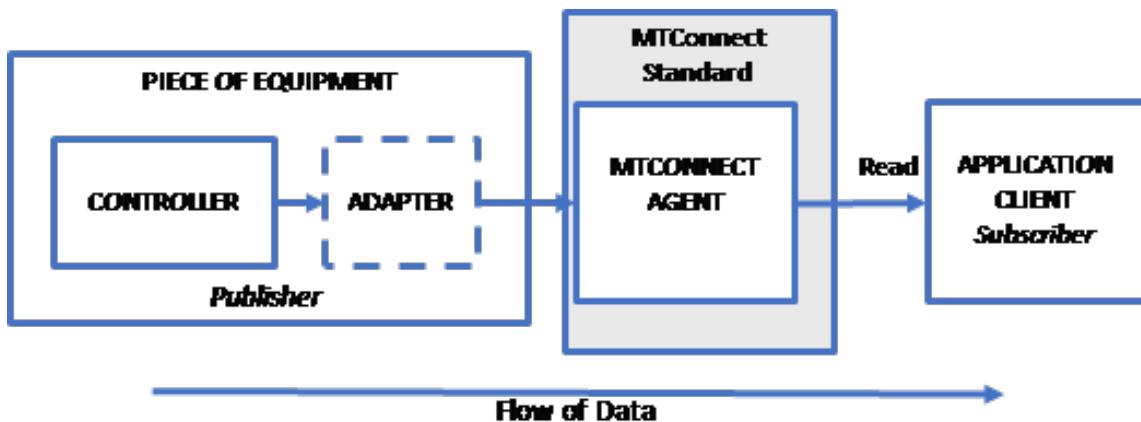


Figure 2: MTConnect Architecture Model

1174 Note: In each implementation of a communication system based on the MTConnect
 1175 Standard, there **MUST** be a schema defined that encodes the rules and termi-
 1176 nology defined for each of the *semantic data models*. These schemas **MAY** be
 1177 used by client software applications to validate the content and structure of the
 1178 *Response Documents* published by an *Agent*.

1179 5.1 Agent

1180 An *Agent* is the centerpiece of an MTConnect implementation. It provides two primary
 1181 functions:

- 1182 • Organizes and manages individual pieces of information published by one or more
 1183 pieces of equipment.

- 1184 • Publishes that information in the form of a *Response Document* to client software
1185 applications.

1186 The MTConnect Standard addresses the behavior of an *Agent* and the structure and mean-
1187 ing of the data published by an *Agent*. It is the responsibility of the implementer of an
1188 *Agent* to determine the means by which the behavior is achieved for a specific *Agent*.

1189 An *Agent* is software that may be installed as part of a piece of equipment or it may be
1190 installed separately. When installed separately, an *Agent* may receive information from
1191 one or more pieces of equipment.

1192 Some pieces of equipment may be able to communicate directly to an *Agent*. Other pieces
1193 of equipment may require an *Adapter* to transform the information provided by the equip-
1194 ment into a form that can be sent to an *Agent*. In either case, the method of transmitting
1195 information from the piece of equipment to an *Agent* is implementation dependent and is
1196 not addressed as part of the MTConnect Standard.

1197 One function of an *Agent* is to store information that it receives from a piece of equipment
1198 in an organized manner. A second function of an *Agent* is to receive *Requests* for informa-
1199 tion from one or many client software applications and then respond to those *Requests* by
1200 publishing a *Response Document* that contains the requested information.

1201 There are three types of information stored by an *Agent* that **MAY** be published in a *Re-*
1202 *sponse Document*. These are:

- 1203 • *Equipment Metadata* defines the *Structural Elements* that represent the physical and
1204 logical parts and sub-parts of each piece of equipment that can publish data to the
1205 *Agent*, the relationships between those parts and sub-parts, and the *Data Entities*
1206 associated with each of those *Structural Elements*. This *Equipment Metadata* is
1207 provided in an *MTConnectDevices Response Document*. See *MTConnect Standard:*
1208 *Part 2.0 - Devices Information Model* for more information on *Equipment Metadata*.
- 1209 • *Streaming Data* provides the values published by pieces of equipment for the *Data*
1210 *Entities* defined by the *Equipment Metadata*. *Streaming Data* is provided in an *MT-*
1211 *ConnectStreams Response Document*. See *MTConnect Standard: Part 2.0 - Devices*
1212 *Information Model* for more information on *Streaming Data*.
- 1213 • *MTConnect Assets* represent information used in a manufacturing operation that is
1214 commonly shared amongst multiple pieces of equipment and/or software applica-
1215 tions. *MTConnect Assets* are provided in an *MTConnectAssets Response Document*.
1216 See *MTConnect Standard: Part 4.0 - Assets Information Model* for more informa-
1217 tion on *MTConnect Assets*.

1218 The exchange between an *Agent* and a client software application is a *Request* and *Re-*
1219 *sponse* information exchange mechanism. See *Section 5.4 - Request/Response Information*
1220 *Exchange* for details on this *Request/Response* information exchange mechanism.

1221 **5.1.1 Instance of an Agent**

1222 As described above, an *Agent* collects and organizes values published by pieces of equip-
1223 ment. As with any piece of software, an *Agent* may be periodically restarted. When an
1224 *Agent* restarts, it **MUST** indicate to client software applications whether the information
1225 available in the *buffer* represents a completely new set of data or if the *buffer* includes data
1226 that had been collected prior to the restart of the *Agent*.

1227 Any time an *Agent* is restarted and begins to collect a completely new set of *Streaming*
1228 *Data*, that set of data is referred to as an *instance* of the *Agent*. The *Agent* **MUST** maintain
1229 a piece of information called `instanceId` that represents the specific *instance* of the
1230 *Agent*.

1231 `instanceId` is represented by a 64-bit integer. The `instanceId` **MAY** be imple-
1232 mented using any mechanism that will guarantee that the value for `instanceId` will be
1233 unique each time the *Agent* begins collecting a new set of data.

1234 When an *Agent* is restarted and it provides a method to recover all, or some portion, of
1235 the data that was stored in the *buffer* before it stopped operating, the *Agent* **MUST** use the
1236 same `instanceId` that was defined prior to the restart.

1237 **5.1.2 Storage of Equipment Metadata for a Piece of Equipment**

1238 An *Agent* **MUST** be capable of publishing *Equipment Metadata* for each piece of equip-
1239 ment that publishes information through the *Agent*. *Equipment Metadata* is typically a
1240 static file defining the *Structural Elements* associated with each piece of equipment re-
1241 porting information through the *Agent* and the *Data Entities* that can be associated with
1242 each of these *Structural Elements*. See details on *Structural Elements* and *Data Entities* in
1243 *MTConnect Standard: Part 2.0 - Devices Information Model*.

1244 The MTConnect Standard does not define the mechanism to be used by an *Agent* to ac-
1245 quire, maintain, or store the *Equipment Metadata*. This mechanism **MUST** be defined as
1246 part of the implementation of a specific *Agent*.

1247 5.1.3 Storage of Streaming Data

1248 *Streaming Data* that is published from a piece(s) of equipment to an *Agent* is stored by the
 1249 *Agent* based upon the sequence upon which each piece of data is received. As described
 1250 below, the order in which data is stored by the *Agent* is one of the factors that determines
 1251 the data that may be included in a specific *MTConnectStreams Response Document*.

1252 5.1.3.1 Management of Streaming Data Storage

1253 An *Agent* stores a fixed amount of data. The amount of data stored by an *Agent* is dependent
 1254 upon the implementation of a specific *Agent*. The examples below demonstrate how
 1255 discrete pieces of data received from pieces of equipment are stored.

1256 The method for storing *Streaming Data* in an *Agent* can be thought of as a tube that can
 1257 hold a finite set of balls. Each ball represents the occurrence of a *Data Entity* published
 1258 by a piece of equipment. This data is pushed in one end of the tube until there is no more
 1259 room for additional balls. At that point, any new data inserted will push the oldest data out
 1260 the back of the tube. The data in the tube will continue to shift in this manner as new data
 1261 is received.

1262 This tube is referred to as a *buffer* in an *Agent*.

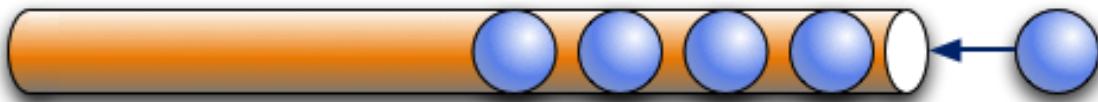


Figure 3: Data Storage in Buffer

1263 In *Figure 4*, the maximum number of *Data Entities* that can be stored in the *buffer* of
 1264 the *Agent* is 8. The maximum number of *Data Entities* that can be stored in the *buffer* is
 1265 represented by a value called `bufferSize`. This example illustrates that when the *buffer*
 1266 fills up, the oldest piece of data falls out the other end.

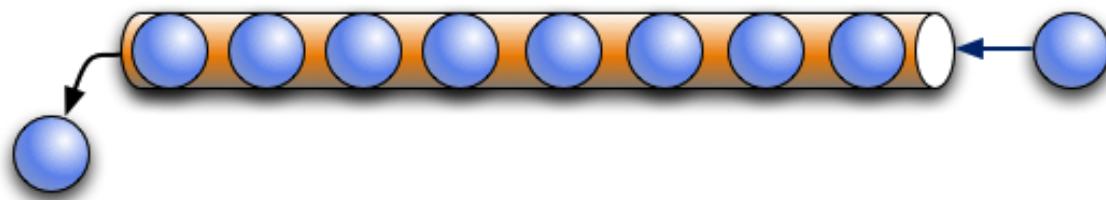


Figure 4: First In First Out Buffer Management

1267 This process constrains the memory storage requirements for an *Agent* to a fixed maximum
1268 size since the MTConnect Standard only requires an *Agent* to store a finite number of
1269 pieces of data.

1270 As an implementation guideline, the *buffer* **SHOULD** be sized large enough to provide
1271 storage for a reasonable amount of information received from all pieces of equipment
1272 that are publishing information to that *Agent*. The implementer should also consider the
1273 impact of a temporary loss of communications between a client software application and
1274 an *Agent* when determining the size for the *buffer*. A larger *buffer* will allow a client
1275 software application more time to reconnect to an *Agent* without losing data.

1276 **5.1.3.2 Sequence Numbers**

1277 In an *Agent*, each occurrence of a *Data Entity* in the *buffer* will be assigned a monotonically
1278 increasing *sequence number* as it is inserted into the *buffer*. The *sequence number*
1279 is a 64-bit integer and the values assigned as *sequence numbers* will never wrap around or
1280 be exhausted; at least within the next 100,000 years based on the size of a 64-bit number.

1281 *sequence number* is the primary key identifier used to manage and locate a specific piece
1282 of data in an *Agent*. The *sequence number* associated with each *Data Entity* reported by
1283 an *Agent* is identified with an attribute called `sequence`.

1284 The *sequence number* for each piece of data **MUST** be unique for an instance of an *Agent*
1285 (see *Section 5.1.1 - Instance of an Agent* for information on *instances* of an *Agent*). If data
1286 is received from more than one piece of equipment, the *sequence numbers* are based on
1287 the order in which the data is received regardless of which piece of equipment produced
1288 that data. The *sequence number* **MUST** be a monotonically increasing number that spans
1289 all pieces of equipment publishing data to an *Agent*. This allows for multiple pieces of
1290 equipment to publish data through a single *Agent* with no *sequence number* collisions and
1291 unnecessary protocol complexity.

1292 The *sequence number* **MUST** be reset to one (1) each time an *Agent* is restarted and begins
1293 to collect a fresh set of data; i.e., each time `instanceId` is changed.

1294 *Figure 5* demonstrates the relationship between `instanceId` and `sequence` when an
1295 *Agent* stops and restarts and begins collecting a new set of data. In this case, the `in-`
1296 `stanceId` is changed to a new value and `value` for `sequence` resets to one (1):

| instanceId | sequence |
|-------------------|-----------------|
| 234556 | 234 |
| | 235 |
| | 236 |
| | 237 |
| | 238 |

| Agent Stops and Restarts | |
|---------------------------------|----------|
| 234557 | 1 |
| | 2 |
| | 3 |
| | 4 |
| | 5 |

Figure 5: instanceId and sequence

1297 *Figure 6* also shows two additional pieces of information defined for an *Agent*:

- 1298 • *firstSequence* – the oldest piece of data contained in the *buffer*; i.e., the next
1299 piece of data to be moved out of the *buffer*
- 1300 • *lastSequence* – the newest data added to the *buffer*

1301 *firstSequence* and *lastSequence* provide guidance to a software application iden-
1302 tifying the range of data available that may be requested from an *Agent*.

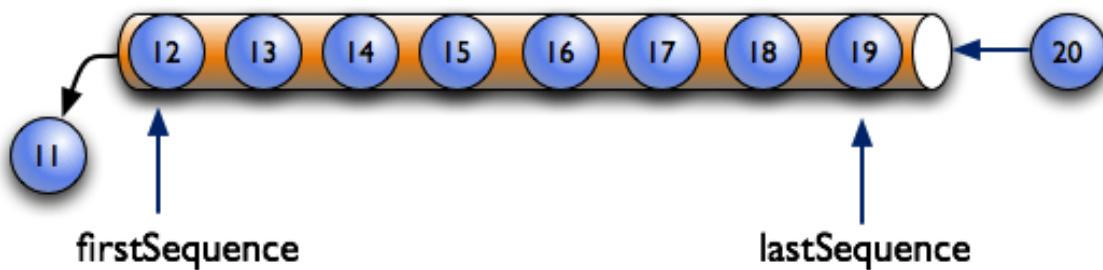


Figure 6: Identifying the range of data with *firstSequence* and *lastSequence*

1303 When a client software application requests data from an *Agent*, it can specify both the
1304 *sequence number* of the first piece of data (*from*) that **MUST** be included in the *Response*

1305 Document and the total number (count) of pieces of data that **SHOULD** be included in
 1306 that document.

1307 In *Figure 7*, the request specifies that the data to be returned starts at *sequence number* 15
 1308 (*from*) and includes a total of three items (count).

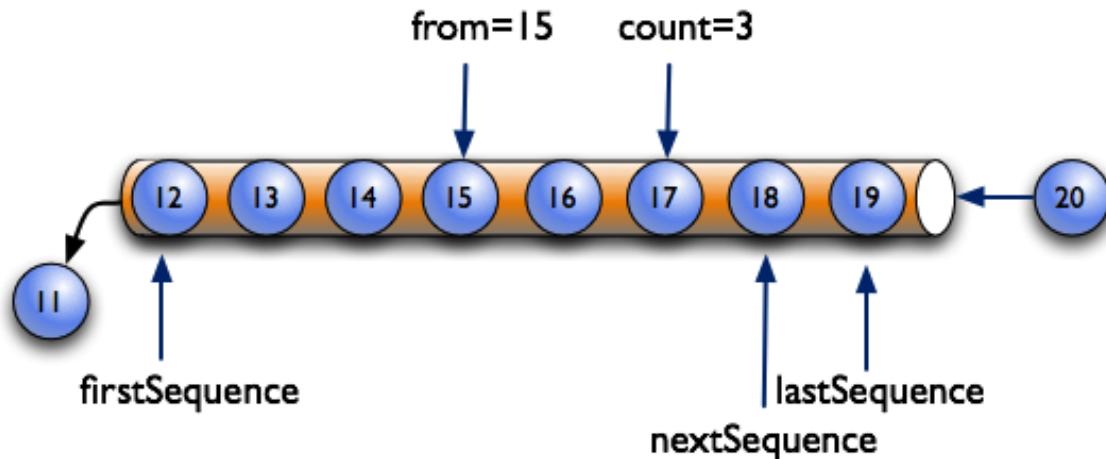


Figure 7: Identifying the range of data with from and count

1309 Once a *Response* to a *Request* has been completed, the value of *nextSequence* will be
 1310 established. *nextSequence* is the *sequence number* of the next piece of data available
 1311 in the *buffer*. In the example in *Figure 7*, the next *sequence number* (*nextSequence*)
 1312 will be 18.

1313 As shown in *Figure 8*, the combination of *from* and *count* defined by the *Request*
 1314 indicates a *sequence number* for data that is beyond that which is currently in the *buffer*.
 1315 In this case, *nextSequence* is set to a value of *lastSequence* + 1.

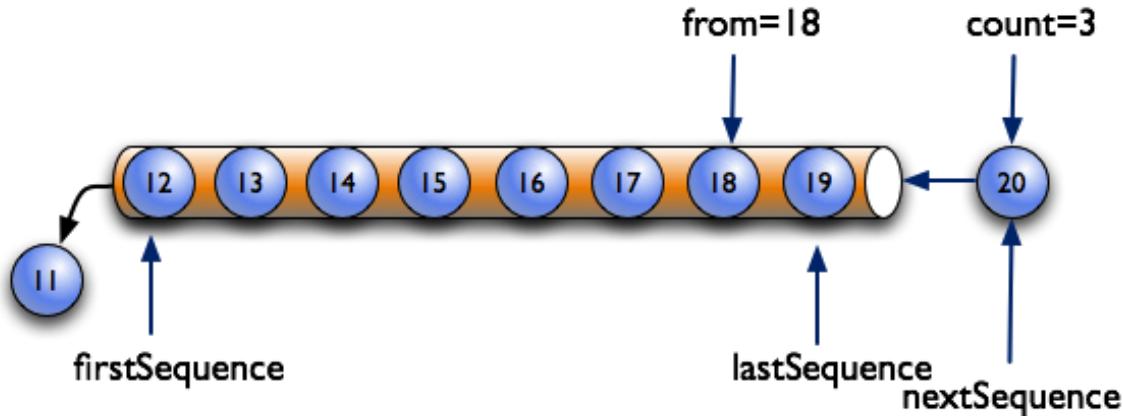


Figure 8: Identifying the range of data with nextSequence and lastSequence

5.1.3.3 Buffer Data Structure

The information in the *buffer* of an *Agent* can be thought of as a four-column table of data.
Each column in the table represents:

- The first column is the *sequence number* associated with each *Data Entity* - sequence.
- The second column is the time that the data was published by a piece of equipment. This time is defined as the *timestamp* associated with that *Data Entity*. See *Section 5.1.3.4 - Time Stamp* for details on *timestamp*.
- The third column, *dataItemId*, refers to the identity of *Data Entities* as they will appear in the *MTConnectStreams Response Document*. See *Section 5 of MTConnect Standard: Part 3.0 - Streams Information Model* for details on *dataItemId* for a *Data Entity* and how that identify relates to the *id* attribute of the corresponding *Data Entity* in the *Devices Information Model*.
- The fourth column is the value associated with each *Data Entity*.

Figure 9 is an example demonstrating the concept of how data may be stored in an *Agent*:

| AGENT | | | |
|--------------|---------------------------------|--------------------|--------------------|
| Seq | Time | dataItemId | Value |
| 101 | 2016-12-13T09:44:00.2221 | AVAIL-28277 | UNAVAILABLE |
| 102 | 2016-12-13T09:54:00.3839 | AVAIL-28277 | AVAILABLE |
| 103 | 2016-12-13T10:00:00.0594 | POS-Y-28277 | 25.348 |
| 104 | 2016-12-13T10:00:00.0594 | POS-Z-28277 | 13.23 |
| 105 | 2016-12-13T10:00:03.2839 | SS-28277 | 0 |
| 106 | 2016-12-13T10:00:03.2839 | POS-X-73746 | 11.195 |
| 107 | 2016-12-13T10:00:03.2839 | POS-Y-73746 | 24.938 |
| 108 | 2016-12-13T10:01:37.8594 | POS-Z-73746 | 1.143 |
| 109 | 2016-12-13T10:02:03.2617 | SS-28277 | 1002 |

Figure 9: Data Storage Concept

1331 The storage mechanism for the data, the internal representation of the data, and the imple-
 1332 mentation of the *Agent* itself is not part of the MTConnect Standard. The implementer can
 1333 choose both the amount of data to be stored in the *Agent* and the mechanism for how the
 1334 data is stored. The only requirement is that an *Agent* publish the *Response Documents* in
 1335 the required format.

1336 5.1.3.4 Time Stamp

1337 Each piece of equipment that publishes information to an *Agent* **SHOULD** provide a time
 1338 stamp indicating when each piece of information was measured or determined. If no time
 1339 stamp is provided, the *Agent* **MUST** provide a time stamp for the information based upon
 1340 when that information was received at the *Agent*.

1341 The timestamp associated with each piece of information is reported by an *Agent* as
 1342 timestamp. timestamp **MUST** be reported in UTC (Coordinated Universal Time)
 1343 format; e.g., "2010-04-01T21:22:43Z".

1344 Note: Z refers to UTC/GMT time, not local time.

1345 Client software applications should use the value of timestamp reported for each piece
 1346 of information as the means for ordering when pieces of information were generated as
 1347 opposed to using sequence for this purpose.

1348 Note: It is assumed that `timestamp` provides the best available estimate of the time
 1349 that the value(s) for the published information was measured or determined.

1350 If two pieces of information are measured or determined at the exact same time, they
 1351 **MUST** be reported with the same value for `timestamp`. Likewise, all information that
 1352 is recorded in the *buffer* with the same value for `timestamp` should be interpreted as
 1353 having been recorded at the same point in time; even if that data was published by more
 1354 than one piece of equipment.

1355 5.1.3.5 Recording Occurrences of Streaming Data

1356 An *Agent* **MUST** record data in the *buffer* each time the value for that specific piece of data
 1357 changes. If a piece of equipment publishes multiple occurrences of a piece of data with
 1358 the same value, the *Agent* **MUST NOT** record multiple occurrence for that *Data Entity*.

1359 Note: There is one exception to this rule. Some *Data Entities* may be defined with a
 1360 representation attribute value of DISCRETE (**DEPRECATED** in *Version 1.5*) (See *Section 7.2.2.12 of MTConnect Standard: Part 2.0 - Devices*
 1361 *Information Model* for details on representation.) In this case, each occur-
 1362 rence of the data represents a new and unique piece of information. The
 1363 *Agent* **MUST** then record each occurrence of the *Data Entity* that is published
 1364 by a piece of equipment.

1366 The value for each piece of information reported by an *Agent* must be considered by a
 1367 client software application to be valid until such a time that another occurrence of that
 1368 piece of information is published by the *Agent*.

1369 5.1.3.6 Maintaining Last Value for Data Entities

1370 An *Agent* **MUST** retain a copy of the last available value associated with each *Data Entity*
 1371 known to the *Agent*; even if an occurrence of that *Data Entity* is no longer in the *buffer*.
 1372 This function allows an *Agent* to provide a software application a view of the last known
 1373 value for each *Data Entity* associated with a piece of equipment.

1374 The *Agent* **MUST** also retain a copy of the last value associated with each *Data Entity* that
 1375 has flowed out of the *buffer*. This function allows an *Agent* to provide a software applica-
 1376 tion a view of the last known value for each *Data Entity* associated with a *Current Request*
 1377 with an `at` parameter in the `query` portion of its *HTTP Request Line* (See *Section 8.3.2 -*
 1378 *Current Request Implemented Using HTTP* for details on *Current Request*).

1379 **5.1.3.7 Unavailability of Data**

1380 An *Agent* **MUST** maintain a list of *Data Entities* that **MAY** be published by each piece of
 1381 equipment providing information to the *Agent*. This list of *Data Entities* is derived from
 1382 the *Equipment Metadata* stored in the *Agent* for each piece of equipment.

1383 Each time an *Agent* is restarted, the *Agent* **MUST** place an occurrence of every *Data*
 1384 *Entity* in the *buffer*. The value reported for each of these *Data Entities* **MUST** be set to
 1385 UNAVAILABLE and the timestamp for each **MUST** be set to the time that the last piece
 1386 of data was collected by the *Agent* prior to the restart.

1387 If at any time an *Agent* loses communications with a piece of equipment, or the *Agent* is
 1388 unable to determine a valid value for all, or any portion, of the *Data Entities* published by
 1389 a piece of equipment, the *Agent* **MUST** place an occurrence of each of these *Data Entities*
 1390 in the *buffer* with its value set to UNAVAILABLE. This signifies that the value is currently
 1391 indeterminate and no assumptions of a valid value for the data is possible.

1392 Since an *Agent* may receive information from multiple pieces of equipment, it **MUST**
 1393 consider the validity of the data from each of these pieces of equipment independently.

1394 There is one exception to the rules above. Any *Data Entity* that is constrained to a constant
 1395 data value **MUST** be reported with the constant value and the *Agent* **MUST NOT** set the
 1396 value of that *Data Entity* to UNAVAILABLE.

1397 Note: The schema for the *Devices Information Model* (defined in *MTConnect Standard: Part 2.0 - Devices Information Model*) defines how the value reported for
 1398 an individual piece of data may be constrained to one or more specific values.
 1399

1400 **5.1.3.8 Persistence and Recovery**

1401 The implementer of an *Agent* must decide on a strategy regarding the storage of *Streaming*
 1402 *Data* in the *buffer* of the *Agent*.

1403 In the simplest form, an *Agent* can hold the *buffer* information in volatile memory where
 1404 no data is persisted when the *Agent* is stopped. In this case, the *Agent* **MUST** update the
 1405 value for `instanceId` when the *Agent* restarts to indicate that the *Agent* has begun to
 1406 collect a new set of data.

1407 If the implementation of an *Agent* provides a method of persisting and restoring all or
 1408 a portion of the information in the *buffer* of the *Agent* (*sequence numbers, time stamps,*
 1409 *identify, and values*), the *Agent* **MUST NOT** change the value of the `instanceId` when
 1410 the *Agent* restarts. This will indicate to a client software application that it does not need to
 1411 reset the value for `nextSequence` when it requests the next set of data from the *Agent*.

1412 When an implementer chooses to provide a method to persist the information in an *Agent*,
 1413 they may choose to store as much data as is practical in a recoverable storage system. Such
 1414 a method may also include the ability to store historical information that has previously
 1415 been pushed out of the *buffer*.

1416 **5.1.3.9 Heartbeat**

1417 An *Agent* **MUST** provide a function that indicates to a client application that the HTTP
 1418 connection is still viable during times when there is no new data available to report in a
 1419 *Response Document*. This function is defined as *heartbeat*.

1420 *heartbeat* represents the amount of time after a *Response Document* has been published
 1421 until a new *Response Document* **MUST** be published, even when no new data is available.

1422 See *Section 8.3.3.2 - Query Portion of the HTTP Request Line for a Sample Request* for
 1423 more details on configuring the *heartbeat* function.

1424 **5.1.3.10 Data Sets**

1425 An *Agent* **MUST** maintain the current state of the *Data Set* for every *Data Entity* with a
 1426 representation of *Data Set* for all data associated with a *sequence number* as described in
 1427 *Section 5.1.3.1 - Management of Streaming Data Storage*.

1428 *Data Entities* represented as *Data Sets* provides a facility for providing multiple values
 1429 for a single *Data Entity* where each entry in the *Data Set* is a *key-value pair* uniquely
 1430 identified by the *key*. For more details on *Data Entities* defined as *Data Sets*, see *MTConnect*
 1431 *Standard: Part 2.0 - Devices Information Model Section 7.2.2.12* and *MTConnect*
 1432 *Standard: Part 3.0 - Streams Information Model Section 5.3.4*.

1433 Any number of *key-value pairs* may be added, removed or changed in a single update to
 1434 the *Data Set*. An *Agent* **MUST** publish the changes to one or more *key-value pairs* as a
 1435 single *Data Entity* associated with a single *sequence number*. An *Agent* **MUST** indicate
 1436 the removal of a *key-value pair* from a *Data Set*.

1437 When the *Data Entity* definition has the *discrete* attribute set to *false* or is not
 1438 present, an *Agent*, when streaming data, **MUST** suppress identical successive *key-value*
 1439 *pairs* and only publish the *key-value pairs* that have changed since the previous state of
 1440 the *Data Set*.

1441 When the *Data Entity* definition has the *discrete* attribute set to *true*, an *Agent*, when
 1442 streaming data, **MUST** report all *key-value pairs* regardless of the previous state of the
 1443 *Data Set*, and **MUST NOT** suppressed any identical *key-value pairs*.

1444 When a *reset* occurs, the current state of the *Data Set* **MUST** be cleared and contain no
 1445 *key-value pairs*. The *Data Set* **MAY** be simultaneously populated with a new set of *key-*
 1446 *value pairs*. The previous entries **MUST NOT** be included and **MUST NOT** indicate
 1447 removal. An *Agent* **MUST NOT** suppress reporting any *key-value pairs* regardless of the
 1448 prior state of the *Data Set*.

1449 When the *Data Entity* is UNAVAILABLE the *Data Set* **MUST** be cleared and contain no
 1450 *key-value pairs*. The prior state of the *Data Set* **MUST** not be retained and the *Data Set*
 1451 **MUST** be repopulated when the data is available.

1452 5.1.4 Storage of Documents for MTConnect Assets

1453 An *Agent* also stores information associated with *MTConnect Assets*.

1454 When a piece of equipment publishes a document that represents information associated
 1455 with an *MTConnect Asset*, an *Agent* stores that document in a *buffer*. This *buffer* is called
 1456 the *assets buffer*. The document is called an *Asset Document*.

1457 The *assets buffer* **MUST** be a separate *buffer* from the one where the *Streaming Data* is
 1458 stored.

1459 The *Asset Document* that is published by the piece of equipment **MUST** be organized
 1460 based upon one of the applicable *Asset Information Models* defined in one of the *Parts 4.x*
 1461 of the *MTConnect Standard*.

1462 An *Agent* will only retain a limited number of *Asset Documents* in the *assets buffer*. The
 1463 *assets buffer* functions similar to the *buffer* for *Streaming Data*; i.e., when the *assets buffer*
 1464 is full, the oldest *Asset Document* is pushed from the *buffer*.

1465 *Figure 10* demonstrates the oldest *Asset Document* being pushed from the *assets buffer*
 1466 when a new *Asset Document* is added and the *assets buffer* is full:

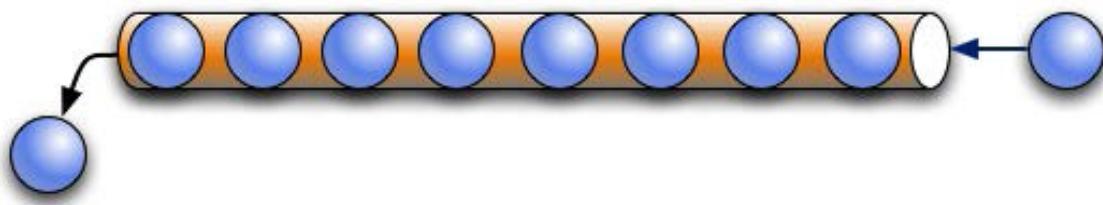


Figure 10: First In First Out Asset Buffer Management

1467 Within an *Agent*, the management of *Asset Documents* behave like a key/value storage in a
 1468 database. In the case of *MTConnect Assets*, the key is an identifier for an Asset (see details

1469 on `assetId` in *MTConnect Standard: Part 4.0 - Assets Information Model*) and the value
 1470 is the *Asset Document* that was published by the piece of equipment.

1471 *Figure 11* demonstrates the relationship between the key (`assetId`) and the stored *Asset*
 1472 *Documents*:

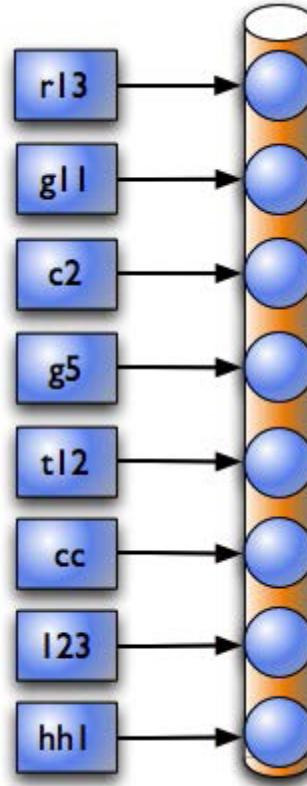


Figure 11: Relationship between `assetId` and stored *Asset documents*

1473 Note: The key (`assetId`) is independent of the order of the *Asset Documents* stored
 1474 in the *assets buffer*.

1475 When an *Agent* receives a new *Asset Document* representing an *MTConnect Asset*, it must
 1476 determine whether this document represents an *MTConnect Asset* that is not currently
 1477 represented in the *assets buffer* or if the document represents new information for an *MT-*
 1478 *Connect Asset* that is already represented in the *assets buffer*. When a new *Asset Document*
 1479 is received, one of the following **MUST** occur:

- 1480 • If the *Asset Document* represents an *MTConnect Asset* that is not currently repre-
 1481 sented in the *assets buffer*, the *Agent* **MUST** add the new document to the front
 1482 of the *assets buffer*. If the *assets buffer* is full, the oldest *Asset Document* will be
 1483 removed from the *assets buffer*.

- 1484 • If the *Asset Document* represents an *MTConnect Asset* that is already represented in
 1485 the *assets buffer*, the *Agent* **MUST** remove the existing *Asset Document* representing
 1486 that *MTConnect Asset* from the *assets buffer* and add the new *Asset Document* to the
 1487 front of the *assets buffer*.

1488 The MTConnect Standard does not specify the maximum number of *Asset Documents*
 1489 that may be stored in the *assets buffer*; that limit is determined by the implementation
 1490 of a specific *Agent*. The number of *Asset Documents* that may be stored in an *Agent* is
 1491 defined by the value for `assetBufferSize` (See *Section 6.5 - Document Header* for
 1492 more information on `assetBufferSize`). A value of 4,294,967,296 or 2^{32} can be
 1493 provided for `assetBufferSize` to indicate unlimited storage.

1494 There is no requirement for an *Agent* to provide persistence for the *Asset Documents* stored
 1495 in the *assets buffer*. If an *Agent* should fail, all *Asset Documents* stored in the *assets buffer*
 1496 **MAY** be lost. It is the responsibility of the implementer to determine if *Asset Documents*
 1497 stored in an *Agent* may be restored or if those *Asset Documents* are retained by some other
 1498 software application.

1499 Additional details on how an *Agent* organizes and manages information associated with
 1500 *MTConnect Assets* are provided in *MTConnect Standard: Part 4.0 - Assets Information*
 1501 *Model*.

1502 5.2 Response Documents

1503 *Response Documents* are electronic documents generated and published by an *Agent* in
 1504 response to a *Request* for data.

1505 The *Response Documents* defined in the MTConnect Standard are:

- 1506 • *MTConnectDevices Response Document*: An electronic document that contains the
 1507 information published by an *Agent* describing the data that can be published by one
 1508 or more piece(s) of equipment. The structure of the *MTConnectDevices Response*
 1509 *Document* document is based upon the requirements defined by the *Devices Infor-*
 1510 *mation Model*. See *MTConnect Standard: Part 2.0 - Devices Information Model* for
 1511 details on this information model.
- 1512 • *MTConnectStreams Response Document*: An electronic document that contains the
 1513 information published by an *Agent* that contains the data that is published by one
 1514 or more piece(s) of equipment. The structure of the *MTConnectStreams Response*

1515 *Document* document is based upon the requirements defined by the *Streams Infor-*
 1516 *mation Model*. See *MTConnect Standard: Part 3.0 - Streams Information Model* for
 1517 details on this information model.

- 1518 • *MTConnectAssets Response Document*: An electronic document that contains the
 1519 information published by an *Agent* that **MAY** include one or more *Asset Documents*.
 1520 The structure of the *MTConnectAssets Response Document* document is based upon
 1521 the requirements defined by the *Asset Information Models*. See *MTConnect Stan-*
 1522 **dard: Part 4.0 - Assets Information Model** for details on this information model.
- 1523 • *MTConnectErrors Response Document*: An electronic document that contains the
 1524 information provided by an *Agent* when an error has occurred when trying to re-
 1525 spond to a *Request* for data. The structure of the *MTConnectErrors Response Doc-*
 1526 ument is based upon the requirements defined by the *Error Information Model*. See
 1527 **Section 9 - Error Information Model** of this document for details on this information
 1528 model.

1529 *Response Documents* may be represented by any document format supported by an *Agent*.
 1530 No matter what document format is used to structure these documents, the requirements
 1531 for representing the data and other information contained in those documents **MUST** ad-
 1532 here to the requirements defined in the *Information Models* associated with each document.

1533 5.2.1 XML Documents

1534 XML is currently the only document format supported by the MTConnect Standard for
 1535 encoding *Response Documents*. Other document formats may be supported in the future.

1536 Since XML is the document format supported by the MTConnect Standard for encoding
 1537 documents, all examples demonstrating the structure of the *Response Documents* provided
 1538 throughout the MTConnect Standard are based on XML. These documents will be referred
 1539 to as *MTConnect XML Documents* or *XML Documents*.

1540 *Section 6 - XML Representation of Response Documents* defines how each document is
 1541 structured as an *XML Document*.

1542 5.3 Semantic Data Models

1543 A *semantic data model* is a software engineering method for representing data where the
 1544 context and the meaning of the data is constrained and fully defined.

1545 Each of the *semantic data models* defined by the MTConnect Standard include:

- 1546 • The types of information that may be published by a piece of equipment,
- 1547 • The meaning of that information and units of measure, if applicable,
- 1548 • Structural information that defines how different pieces of information relate to each
1549 other, and
- 1550 • Structural information that defines how the information relates to where the infor-
1551 mation was measured or generated by the piece of equipment.

1552 As described previously, the content of the *Response Documents* provided by an *Agent* are
1553 each defined by a specific *semantic data model*. The details for the *semantic data model*
1554 used to define each of the *Response Documents* are detail as follows:

- 1555 • *MTConnectDevices Response Document: MTConnect Standard: Part 2.0 - Devices*
1556 *Information Model.*
- 1557 • *MTConnectStreams Response Document: MTConnect Standard: Part 3.0 - Streams*
1558 *Information Model.*
- 1559 • *MTConnectAssets Response Document: MTConnect Standard: Part 4.0 - Assets*
1560 *Information Model* and its sub-Parts.
- 1561 • *MTConnectErrors Response Document: MTConnect Standard Part 1.0 - Overview*
1562 *and Fundamentals, Section 9 - Error Information Model.*

1563 Without semantics, a single piece of data does not convey any relevant meaning to a person
1564 or a client software application. However, when that piece of data is paired with some
1565 semantic context, the data inherits significantly more meaning. The data can then be more
1566 completely interpreted by a client software application without human intervention.

1567 The MTConnect *semantic data models* allows the information published by a piece of
1568 equipment to be transmitted to client software application with a full definition of the
1569 meaning of that information and in full context defining how that information relates to
1570 the piece of equipment that measured or generated the information.

1571 5.4 Request/Response Information Exchange

1572 The transfer of information between an *Agent* and a client software application is based
 1573 on a *Request/Response* information exchange approach. A client software application
 1574 requests specific information from an *Agent*. An *Agent* responds to the *Request* by pub-
 1575 lishing a *Response Document*.

1576 In normal operation, there are four types of *MTConnect Requests* that can be issued by
 1577 a client software application that will result in different *Responses* by an *Agent*. These
 1578 *Requests* are:

1579 • *Probe Request*– A client software application requests the *Equipment Metadata* for
 1580 each piece of equipment that **MAY** publish information through an *Agent*. The *Agent*
 1581 publishes a *MTConnectDevices Response Document* that contains the requested in-
 1582 formation. A *Probe Request* is represented by the term `probe` in a *Request* from a
 1583 client software application.

1584 • *Current Request* – A client software application requests the current value for each
 1585 of the data types that have been published from a piece(s) of equipment to an *Agent*.
 1586 The *Agent* publishes a *MTConnectStreams Response Document* that contains the
 1587 requested information. A *Current Request* is represented by the term `current` in
 1588 a *Request* from a client software application.

1589 • *Sample Request* – A client software application requests a series of data values from
 1590 the *buffer* in an *Agent* by specifying a range of *sequence numbers* representing that
 1591 data. The *Agent* publishes a *MTConnectStreams Response Document* that contains
 1592 the requested information. A *Sample Request* is represented by the term `sample` in
 1593 a *Request* from a client software application.

1594 • *Asset Request* – A client software application requests information related to *MT-*
 1595 *Connect Assets* that has been published to an *Agent*. The *Agent* publishes an *MT-*
 1596 *ConnectAssets Response Document* that contains the requested information. An *As-*
 1597 *set Request* is represented by the term `asset` in a *Request* from a client software
 1598 application.

1599 Note: If an *Agent* is unable to respond to the request for information or the re-
 1600 quest includes invalid information, the *Agent* will publish an *MTConnectErrors*
 1601 *Response Document*. See *Section 9 - Error Information Model* for information
 1602 regarding *Error Information Model*

1603 The specific format for the *Request* for information from an *Agent* will depend on the
 1604 *Protocol* implemented as part of the *Request/Response* information exchange mechanism

1605 deployed in a specific implementation. See *Section 7 - Protocol and Messaging*, *Protocol*
1606 for details on implementing the *Request/Response* information exchange.

1607 Also, the specific format for the *Response Documents* may also be implementation de-
1608 pending. See *Section 6 - XML Representation of Response Documents* for details on the
1609 format for the *Response Documents* encoded with XML.

1610 **5.5 Accessing Information from an Agent**

1611 Each of the *Requests* defined for the *Request/Response* information exchange requires
1612 an *Agent* to respond with a specific view of the information stored by the *Agent*. The
1613 following describes the relationships between the information stored by an *Agent* and the
1614 contents of the *Response Documents*.

1615 **5.5.1 Accessing Equipment Metadata from an Agent**

1616 The *Equipment Metadata* associated with each piece of equipment that publishes infor-
1617 mation to an *Agent* is typically static information that is maintained by the *Agent*. The
1618 MTConnect Standard does not define how the *Agent* captures or maintains that informa-
1619 tion. The only requirement that the MTConnect Standard places on an *Agent* regarding this
1620 *Equipment Metadata* is that the *Agent* properly store this information and then configure
1621 and publish a *MTConnectDevices Response Document* in response to a *Probe Request*.

1622 All issues associated with the capture and maintenance of the *Equipment Metadata* is the
1623 responsibility of the implementer of a specific *Agent*.

1624 **5.5.2 Accessing Streaming Data from the Buffer of an Agent**

1625 There are two *Requests* defined for the *Request/Response* information exchange that re-
1626 quire an *Agent* to provide different views of the information stored in the *buffer* of the
1627 *Agent*. These *Requests* are *current* and *sample*.

1628 The example in *Figure 12* demonstrates how an *Agent* interprets the information stored
1629 in the *buffer* to provide the content that is published in different versions of the *MTCon-*
1630 *nnectStreams Response Document* based on the specific *Request* that is issued by a client
1631 software application.

1632 In this example, an *Agent* with a *buffer* that can hold up to eight (8) *Data Entities*; i.e., the

1633 value for bufferSize is 8. This *Agent* is collecting information for two pieces of data
 1634 – Pos representing a position and Line representing a line of logic or commands in a
 1635 control program.

1636 In this *buffer*, the value for firstSequence is 12 and the value for lastSequence
 1637 is 19. There are five (5) different values for Pos and three (3) different values for Line.

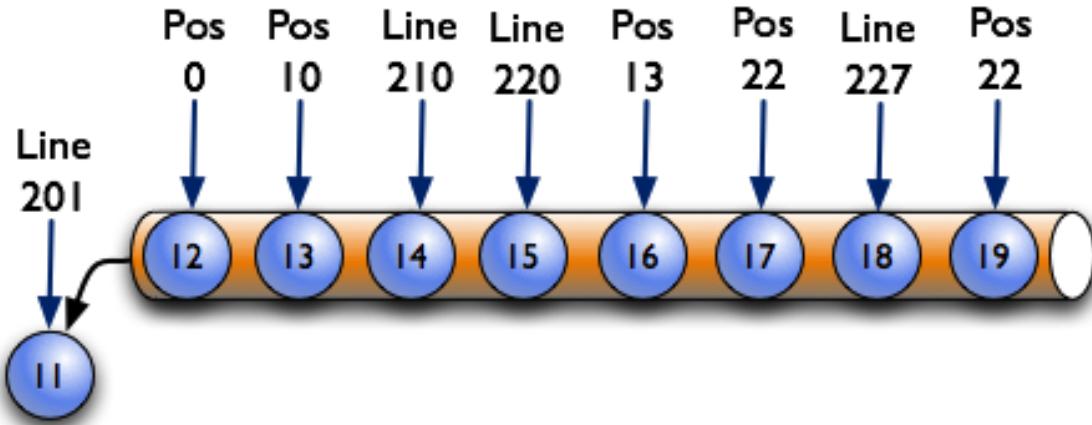


Figure 12: Example Buffer

1638 If an *Agent* receives a *Sample Request* from a client software application, the *Agent* **MUST**
 1639 publish an *MTConnectStreams Response Document* that contains a range of data values.
 1640 The range of values are defined by the `from` and `count` parameters that must be included
 1641 as part of the *Sample Request*. If the value of `from` is 14 and the value of `count` is 5,
 1642 the *Agent* **MUST** publish an *MTConnectStreams Response Document* that includes five
 1643 (5) pieces of data represented by *sequence numbers* 14, 15, 16, 17, and 18 – three (3)
 1644 occurrences of *Line* and two (2) occurrences of *Pos*. In this case, *nextSequence* will
 1645 also be returned with a value of 19.

1646 Likewise, if the same *Agent* receives a *Current Request* from a client software application,
 1647 the *Agent* **MUST** publish an *MTConnectStreams Response Document* that contains the
 1648 most current information available for each of the types of data that is being published to
 1649 the *Agent*. In this case, the specific data that **MUST** be represented in the *MTConnect-
 1650 Streams Response Document* is *Pos* with a value of 22 and a *sequence number* of 19 and
 1651 *Line* with a value of 227 and a *sequence number* of 18.

1652 There is also a derivation of the *Current Request* that will cause an *Agent* to publish an
 1653 *MTConnectStreams Response Document* that contains a set of data relative to a specific
 1654 *sequence number*. The *Current Request* **MAY** include an additional parameter called `at`.
 1655 When the `at` parameter, along with an `instanceId`, is included as part of a *Current Re-
 1656 quest*, an *Agent* **MUST** publish an *MTConnectStreams Response Document* that contains

1657 the most current information available for each of the types of *Data Entities* that are being
1658 published to the *Agent* that occur immediately at or before the *sequence number* specified
1659 with the `at` parameter.

1660 For example, if the *Request* is `current?at=15`, an *Agent* **MUST** publish a *MTConnectStreams Response Document* that contains the most current information available for
1661 each of the *Data Entities* that are stored in the *buffer* of the *Agent* with a *sequence number*
1662 of 15 or lower. In this case, the specific data that **MUST** be represented in the *MTConnectStreams Response Document* is `Pos` with a value of 10 and a *sequence number* of 13
1664 and `Line` with a value of 220 and a *sequence number* of 15.

1666 If a current *Request* is received for a *sequence number* of 11 or lower, an *Agent* **MUST**
1667 return an `OUT_OF_RANGE` *MTConnectErrors Response Document*. The same *HTTP Error*
1668 *Message* **MUST** be given if a *sequence number* is requested that is greater than the
1669 end of the *buffer*. See *Section 9 - Error Information Model* for more information on *MT-*
1670 *ConnectErrors Response Document*.

1671 5.5.3 Accessing MTConnect Assets Information from an Agent

1672 When an *Agent* receives an *Asset Request*, the *Agent* **MUST** publish an *MTConnectAssets*
1673 document that contains information regarding the *Asset Documents* that are stored
1674 in the *Agent*.

1675 See *MTConnect Standard: Part 4.0 - Assets Information Model* for details on *MTConnect*
1676 *Assets*, *Asset Requests*, and the *MTConnectAssets Response Document*.

1677 6 XML Representation of Response Documents

1678 As defined in *Section 5.2.1 - XML Documents*, XML is currently the only language sup-
 1679 ported by the MTConnect Standard for encoding *Response Documents*.

1680 *Response Documents* must be valid and conform to the *schema* defined in the *semantic*
 1681 *data model* defined for that document. The *schema* for each *Response Document* **MUST**
 1682 be updated to correlate to a specific version of the MTConnect Standard. Versions, within
 1683 a *major* version, of the MTConnect Standard will be defined in such a way to best maintain
 1684 backwards compatibility of the *semantic data models* through all *minor* revisions of the
 1685 Standard. However, new *minor* versions may introduce extensions or enhancements to
 1686 existing *semantic data models*.

1687 To be valid, a *Response Document* must be well-formed; meaning that, amongst other
 1688 things, each element has the required XML *start-tag* and *end-tag* and that the document may also include
 1689 a determination that required elements and attributes are present, they only occur in the
 1690 appropriate location in the document, and they appear only the correct number of times.
 1691 If the document is not well-formed, it may be rejected by a client software application.
 1692 The *semantic data model* defined for each *Response Document* also specifies the elements
 1693 and *Child Elements* that may appear in a document. XML elements may contain *Child*
 1694 *Elements*, CDATA, or both. The *semantic data model* also defines the number of times
 1695 each element and *Child Element* may appear in the document.

1697 Each *Response Document* encoded using XML consists of the following primary sections:

- 1698 ● XML Declaration
- 1699 ● Root Element
- 1700 ● Schema and Namespace Declaration
- 1701 ● Document Header
- 1702 ● Document Body

1703 The following will provide details defining how each of the *Response Documents* are en-
 1704 coded using XML.

1705 Note: See *Section 3 - Terminology and Conventions* for the definition of XML related
 1706 terms used in the MTConnect Standard.

1707 6.1 Fundamentals of Using XML to Encode Response Documents

1708 The MTConnect Standard follows industry conventions for formatting the elements and
 1709 attributes included in an XML document. The general guidelines are as follows:

1710 • All element names **MUST** be specified in Pascal case (first letter of each word is
 1711 capitalized). For example: <PowerSupply/>.

1712 • The name for an attribute **MUST** be Camel case; similar to Pascal case, but the first
 1713 letter will be lower case. For example: <MyElement nativeName="bob" />
 1714 where MyElement is the *Element Name* and nativeName is an attribute.

1715 • All CDATA values that are defined with a limited or controlled vocabulary **MUST**
 1716 be in upper case with an _ (underscore) separating words. For example: ON, OFF,
 1717 ACTUAL, and COUNTER_CLOCKWISE.

1718 • The values provided for a date and/or a time **MUST** follow the W3C ISO 8601
 1719 format with an arbitrary number of decimals representing fractions of a second.
 1720 Refer to the following specification for details on the format for dates and times:
 1721 <http://www.w3.org/TR/NOTE-datetime>.

1722 The format for the value describing a date and a time will be
 1723 YYYY-MM-DDThh:mm:ss.ffff. An example would be: 2017-01-13T13:01.213415Z.

1724 Note: Z refers to UTC/GMT time, not local time.

1725 The accuracy and number of decimals representing fractions of a second for a timestamp
 1726 **MUST** be determined by the capabilities of the piece of equipment publishing
 1727 information to an *Agent*. All time values **MUST** be provided in UTC (GMT).

1728 • XML element names **MUST** be spelled out and abbreviations are not permitted. See
 1729 the exclusion below regarding the use of the suffix Ref.

1730 • XML attribute names **SHOULD** be spelled out and abbreviations **SHOULD** be
 1731 avoided. The exception to this rule is the use of id when associated with an identifier.
 1732 See the exclusion below regarding the use of the suffix Ref.

1733 • The abbreviation Ref for Reference is permitted as a suffix to element names of
 1734 either a *Structural Element* or a *Data Entity* to provide an efficient method to associate
 1735 information defined in another location in a *Data Model* without duplicating
 1736 that original data or structure. See *Section 4.8 in MTConnect Standard: Part 2.0 -*
 1737 *Devices Information Model* for more information on Reference.

1738 **6.2 XML Declaration**

1739 The first section of a *Response Document* encoded with XML **SHOULD** be the *XML*
1740 *Declaration*. The declaration is a single element.

1741 An example of an *XML Declaration* would be:

Example 2: Example of xml declaration

1742 1 <?xml version="1.0" encoding="UTF-8"?>

1743 This element provides information regarding how the XML document is encoded and the
1744 character type used for that encoding. See the W3C website for more details on the XML
1745 declaration.

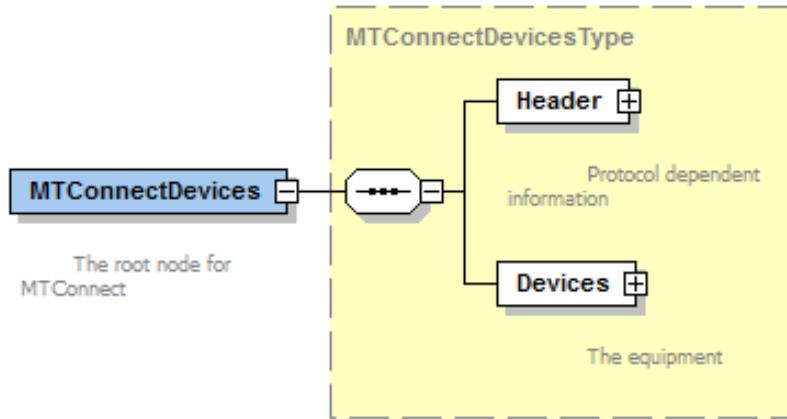
1746 **6.3 Root Element**

1747 Every *Response Document* **MUST** contain only one root element. The MTConnect Stan-
1748 dard defines MTConnectDevices, MTConnectStreams, MTConnectAssets, and
1749 MTConnectError as *Root Elements*.

1750 The *Root Element* specifies a specific *Response Document* and appears at the top of the
1751 document immediately following the *XML Declaration*.

1752 **6.3.1 MTConnectDevices Root Element**

1753 MTConnectDevices is the *Root Element* for the *MTConnectDevices Response Docu-*
1754 *ment*.

**Figure 13:** MTConnectDevices Structure

1755 MTConnectDevices **MUST** contain two *Child Elements* - Header and Devices.

1756 Details for Header are defined in *Section 6.5 - Document Header*.

1757 Devices is an XML container that represents the *Document Body* for an *MTConnectDevices Response Document* – see *Section 6.6 - Document Body*. Details for the *semantic data model* describing the contents for Devices are defined in *MTConnect Standard: Part 2.0 - Devices Information Model*.

1758

1759

1760

1761 MTConnectDevices also has a number of attributes. These attributes are defined in
1762 *Section 6.4 - Schema and Namespace Declaration*.

1763 6.3.1.1 MTConnectDevices Elements

1764 An MTConnectDevices element **MUST** contain a Header and a Devices element.

Table 1: Elements for MTConnectDevices

| Element | Description | Occurrence |
|---------|---|------------|
| Header | An XML container in an <i>MTConnect Response Document</i> that provides information from an <i>Agent</i> defining version information, storage capacity, and parameters associated with the data management within the <i>Agent</i> . | 1 |

| Continuation of Table 1 | | |
|-------------------------|--|------------|
| Element | Description | Occurrence |
| Devices | The XML container in an <i>MTConnect Response Document</i> that provides the <i>Equipment Metadata</i> for each of the pieces of equipment associated with an <i>Agent</i> . | 1 |

1765 6.3.2 MTConnectStreams Root Element

1766 MTConnectStreams is the *Root Element* for the *MTConnectStreams Response Document*.
 1767

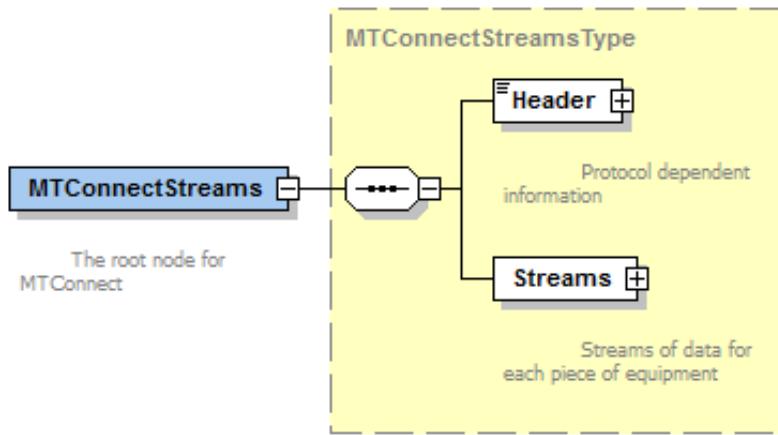


Figure 14: MTConnectStreams Structure

1768 MTConnectStreams **MUST** contain two *Child Elements* - Header and Streams.

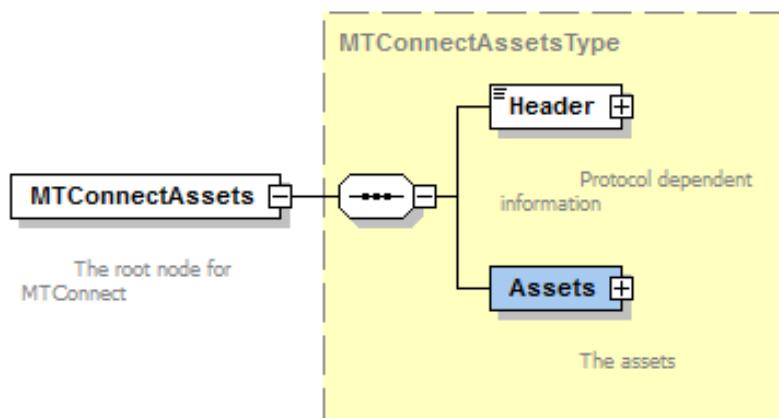
1769 Details for Header are defined in *Section 6.5 - Document Header*.

1770 Streams is an XML container that represents the *Document Body* for a *MTConnectStreams Response Document* – see *Section 6.6 - Document Body*. Details for the *semantic data model* describing the contents for Streams are defined in *MTConnect Standard: Part 3.0 - Streams Information Model*.

1774 MTConnectStreams also has a number of attributes. These attributes are defined in *Section 6.4 - Schema and Namespace Declaration*.

1776 **6.3.2.1 MTConnectStreams Elements**1777 An `MTConnectStreams` element **MUST** contain a `Header` and a `Streams` element.**Table 2:** Elements for `MTConnectStreams`

| Element | Description | Occurrence |
|---------|---|------------|
| Header | An XML container in an <i>MTConnect Response Document</i> that provides information from an <i>Agent</i> defining version information, storage capacity, and parameters associated with the data management within the <i>Agent</i> . | 1 |
| Streams | The XML container for the information published by an <i>Agent</i> in a <i>MTConnectStreams Response Document</i> . | 1 |

1778 **6.3.3 MTConnectAssets Root Element**1779 `MTConnectAssets` is the *Root Element* for the *MTConnectAssets Response Document*.**Figure 15:** `MTConnectAssets` Structure

1780 MTConnectAssets **MUST** contain two *Child Elements* - Header and Assets.

1781 Details for Header are defined in *Section 6.5 - Document Header*.

1782 Assets is an XML container that represents the *Document Body* for an *MTConnectAssets Response Document* – see *Section 6.6 - Document Body*. Details for the *semantic data model* describing the contents for Assets are defined in *MTConnect Standard: Part 4.0 - Assets Information Model*.

1786 MTConnectAssets also has a number of attributes. These attributes are defined in
1787 *Section 6.4 - Schema and Namespace Declaration*.

1788 6.3.3.1 MTConnectAssets Elements

1789 An MTConnectAssets element **MUST** contain a Header and an Assets element.

Table 3: Elements for MTConnectAssets

| Element | Description | Occurrence |
|---------|---|------------|
| Header | An XML container in an <i>MTConnect Response Document</i> that provides information from an <i>Agent</i> defining version information, storage capacity, and parameters associated with the data management within the <i>Agent</i> . | 1 |
| Assets | The XML container in an <i>MTConnectAssets Response Document</i> that provides information for <i>MTConnect Assets</i> associated with an <i>Agent</i> . | 1 |

1790 6.3.4 MTConnectError Root Element

1791 MTConnectError is the *Root Element* for the *MTConnectErrors Response Document*.

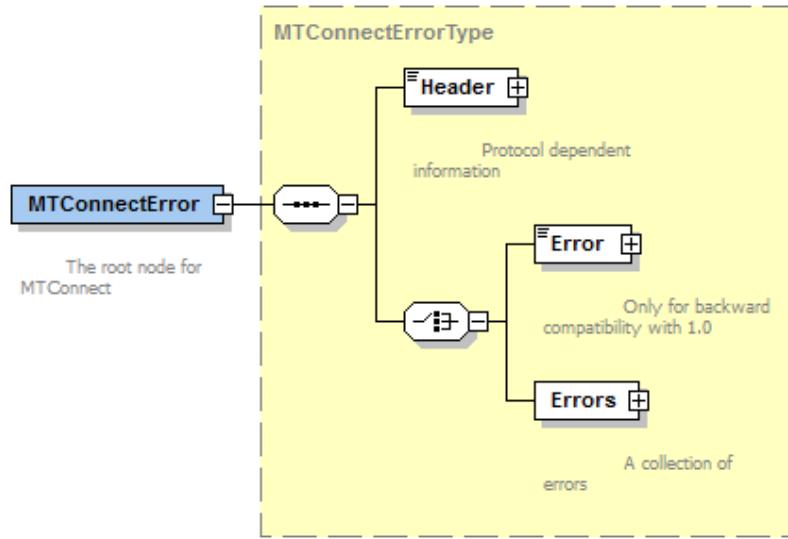


Figure 16: MTConnectError Structure

1792 MTConnectError **MUST** contain two *Child Elements* - Header and Errors.

1793 Note: When compatibility with *Version 1.0.1* and earlier of the MTConnect Standard
 1794 is required for an implementation, the *MTConnectErrors Response Document*
 1795 contains only a single *Error Data Entity* and the *Errors Child Element*
 1796 **MUST NOT** appear in the document.

1797 Details for Header are defined in *Section 6.5 - Document Header*.

1798 Errors is an XML container that represents the *Document Body* for an *MTConnectErrors Response Document* – See *Section 6.6 - Document Body*. Details for the *semantic data model* describing the contents for Errors are defined in *Section 9 - Error Information Model*.

1802 MTConnectError also has a number of attributes. These attributes are defined in *Section 6.4 - Schema and Namespace Declaration*.

1804 6.3.4.1 MTConnectError Elements

1805 An MTConnectError element **MUST** contain a Header and an Errors element.

Table 4: Elements for MTConnectError

| Element | Description | Occurrence |
|---------|---|------------|
| Header | An XML container in an <i>MTConnect Response Document</i> that provides information from an <i>Agent</i> defining version information, storage capacity, and parameters associated with the data management within the <i>Agent</i> . | 1 |
| Errors | The XML container in an <i>MTConnectErrors Response Document</i> that provides information associated with errors encountered by an <i>Agent</i> . | 1 |

1806 6.4 Schema and Namespace Declaration

1807 XML provides standard methods for declaring the *schema* and *namespace* associated with
 1808 a document encoded by XML. The declaration of the *schema* and *namespace* for MTCon-
 1809 nect *Response Documents* **MUST** be structured as attributes in the *Root Element* of the
 1810 document. XML defines these attributes as pseudo-attributes since they provide additional
 1811 information for the entire document and not just specifically for the *Root Element* itself.

1812 Note: If a *Response Document* contains sections that utilize different *schemas* and/or
 1813 *namespaces*, additional pseudo-attributes should appear in the document as de-
 1814 clared using standard conventions as defined by W3C.

1815 For further information on declarations refer to *Appendix C*.

1816 6.5 Document Header

1817 The *Document Header* is an XML container in an *MTConnect Response Document* that
 1818 provides information from an *Agent* defining version information, storage capacity, and
 1819 parameters associated with the data management within the *Agent*. This XML element is
 1820 called *Header*.

1821 *Header* **MUST** be the first XML element following the *Root Element* of any *Response*
 1822 *Document*. The *Header* XML element **MUST NOT** contain any *Child Elements*.

1823 The content of the *Header* element will be different for each type of *Response Document*.

1824 6.5.1 Header for MTConnectDevices

1825 The *Header* element for an *MTConnectDevices Response Document* defines information
 1826 regarding the creation of the document and the data storage capability of the *Agent* that
 1827 generated the document.

1828 6.5.1.1 XML Schema Structure for Header for MTConnectDevices

1829 The *XML Schema* in *Figure 17* represents the structure of the *Header* XML element that
 1830 **MUST** be provided for an *MTConnectDevices Response Document*.

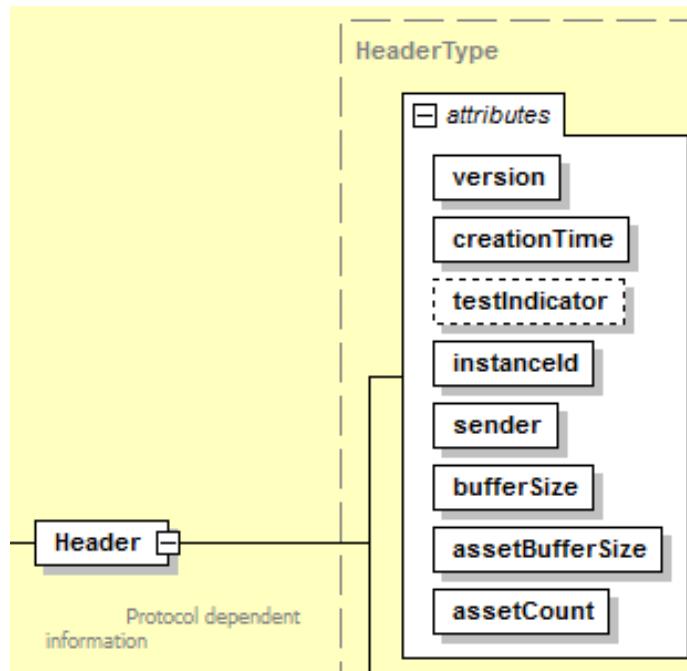


Figure 17: Header Schema Diagram for MTConnectDevices

1831 6.5.1.2 Attributes for Header for MTConnectDevices

1832 *Table 5* defines the attributes that may be used to provide additional information in the
 1833 *Header* element for an *MTConnectDevices Response Document*.

Table 5: MTConnectDevices Header

| Attribute | Description | Occurrence |
|--------------|--|------------|
| version | <p>The <i>major</i>, <i>minor</i>, and <i>revision</i> number of the MTConnect Standard that defines the <i>semantic data model</i> that represents the content of the <i>Response Document</i>. It also includes the revision number of the <i>schema</i> associated with that specific <i>semantic data model</i>.</p> <p>The value reported for <code>version</code> MUST be a series of four numeric values, separated by a decimal point, representing a <i>major</i>, <i>minor</i>, and <i>revision</i> number of the MTConnect Standard and the revision number of a specific <i>schema</i>.</p> <p>As an example, the value reported for <code>version</code> for a <i>Response Document</i> that was structured based on <i>schema</i> revision 10 associated with Version 1.4.0 of the MTConnect Standard would be: 1.4.0.10</p> <p><code>version</code> is a required attribute.</p> | 1 |
| creationTime | <p><code>creationTime</code> represents the time that an <i>Agent</i> published the <i>Response Document</i>.</p> <p><code>creationTime</code> MUST be reported in UTC (Coordinated Universal Time) format; e.g., "2010-04-01T21:22:43Z".</p> <p>Note: Z refers to UTC/GMT time, not local time.</p> <p><code>creationTime</code> is a required attribute.</p> | 1 |

| Continuation of Table 5 | | |
|-------------------------|--|------------|
| Attribute | Description | Occurrence |
| testIndicator | <p>A flag indicating that the <i>Agent</i> that published the <i>Response Document</i> is operating in a test mode. The contents of the <i>Response Document</i> may not be valid and SHOULD be used for testing and simulation purposes only.</p> <p>The values reported for <code>testIndicator</code> are:</p> <ul style="list-style-type: none"> - TRUE: The <i>Agent</i> is functioning in a test mode. - FALSE: The <i>Agent</i> is not function in a test mode. <p>If <code>testIndicator</code> is not specified, the value for <code>testIndicator</code> MUST be interpreted to be FALSE.</p> <p><code>testIndicator</code> is an optional attribute.</p> | 0..1 |
| instanceId | <p>A number indicating a specific instantiation of the <i>buffer</i> associated with the <i>Agent</i> that published the <i>Response Document</i>.</p> <p>The value reported for <code>instanceId</code> MUST be a unique unsigned 64-bit integer.</p> <p>The value for <code>instanceId</code> MUST be changed to a different unique number each time the <i>buffer</i> is cleared and a new set of data begins to be collected.</p> <p><code>instanceId</code> is a required attribute.</p> | 1 |

| Continuation of Table 5 | | |
|-------------------------|--|------------|
| Attribute | Description | Occurrence |
| sender | <p>An identification defining where the <i>Agent</i> that published the <i>Response Document</i> is installed or hosted.</p> <p>The value reported for <code>sender</code> MUST be either an IP Address or Hostname describing where the <i>Agent</i> is installed or the URL of the <i>Agent</i>; e.g., <code>http://<address>[:port]/</code>.</p> <p>Note: The port number need not be specified if it is the default HTTP port 80.</p> <p><code>sender</code> is a required attribute.</p> | 1 |
| bufferSize | <p>A value representing the maximum number of <i>Data Entities</i> that MAY be retained in the <i>Agent</i> that published the <i>Response Document</i> at any point in time.</p> <p>The value reported for <code>bufferSize</code> MUST be a number representing an unsigned 32-bit integer.</p> <p><code>bufferSize</code> is a required attribute.</p> <p>Note 1: <code>bufferSize</code> represents the maximum number of sequence numbers that MAY be stored in the <i>Agent</i>.</p> <p>Note 2: The implementer is responsible for allocating the appropriate amount of storage capacity required to accommodate the <code>bufferSize</code>.</p> | 1 |

| Continuation of Table 5 | | |
|-------------------------|---|------------|
| Attribute | Description | Occurrence |
| assetBufferSize | <p>A value representing the maximum number of <i>Asset Documents</i> that can be stored in the <i>Agent</i> that published the <i>Response Document</i>.</p> <p>The value reported for <code>assetBufferSize</code> MUST be a number representing an unsigned 32-bit integer.</p> <p><code>assetBufferSize</code> is a required attribute.</p> <p>Note: The implementer is responsible for allocating the appropriate amount of storage capacity required to accommodate the <code>assetBufferSize</code>.</p> | 1 |
| assetCount | <p>A number representing the current number of <i>Asset Documents</i> that are currently stored in the <i>Agent</i> as of the <code>creationTime</code> that the <i>Agent</i> published the <i>Response Document</i>.</p> <p>The value reported for <code>assetCount</code> MUST be a number representing an unsigned 32-bit integer and MUST NOT be larger than the value reported for <code>assetBufferSize</code>.</p> <p><code>assetCount</code> is a required attribute.</p> | 1 |

1834 Example 3 is an example of a Header XML element for an *MTConnectDevices Response*
 1835 *Document*:

Example 3: Example of Header XML Element for MTConnectDevices

```

1  <Header creationTime="2017-02-16T16:44:27Z"
2    sender="MyAgent" instanceId="1268463594"
3    bufferSize="131072" version="1.4.0.10"
4    assetCount="54" assetBufferSize="1024"/>

```

1840 6.5.2 Header for MTConnectStreams

1841 The Header element for an *MTConnectStreams Response Document* defines informa-
 1842 tion regarding the creation of the document and additional information necessary for an
 1843 application to interact and retrieve data from the *Agent*.

1844 6.5.2.1 XML Schema Structure for Header for MTConnectStreams

1845 The XML Schema in *Figure 18* represents the structure of the Header XML element that
1846 **MUST** be provided for an *MTConnectStreams Response Document*.

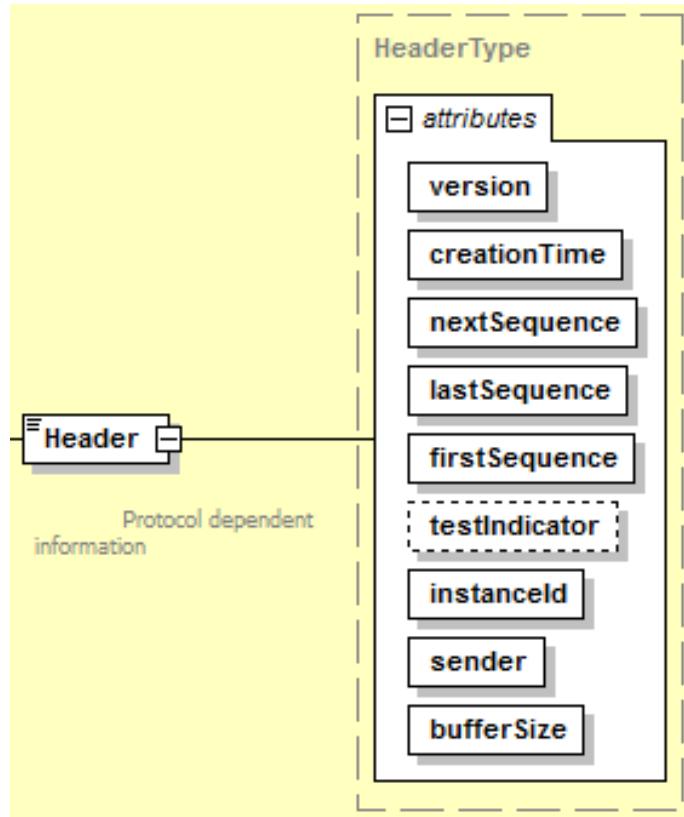


Figure 18: Header Schema Diagram for MTConnectStreams

1847 6.5.2.2 Attributes for MTConnectStreams Header

1848 *Table 6* defines the attributes that may be used to provide additional information in the
1849 Header element for an *MTConnectStreams Response Document*.

Table 6: MTConnectStreams Header

| Attribute | Description | Occurrence |
|--------------|--|------------|
| version | <p>The <i>major</i>, <i>minor</i>, and <i>revision</i> number of the MTConnect Standard that defines the <i>semantic data model</i> that represents the content of the <i>Response Document</i>. It also includes the revision number of the <i>schema</i> associated with that specific <i>semantic data model</i>.</p> <p>The value reported for <i>version</i> MUST be a series of four numeric values, separated by a decimal point, representing a <i>major</i>, <i>minor</i>, and <i>revision</i> number of the MTConnect Standard and the revision number of a specific <i>schema</i>.</p> <p>As an example, the value reported for <i>version</i> for a <i>Response Document</i> that was structured based on <i>schema</i> revision 10 associated with Version 1.4.0 of the MTConnect Standard would be: 1.4.0.10</p> <p><i>version</i> is a required attribute.</p> | 1 |
| creationTime | <p><i>creationTime</i> represents the time that an <i>Agent</i> published the <i>Response Document</i>.</p> <p><i>creationTime</i> MUST be reported in UTC (Coordinated Universal Time) format; e.g., "2010-04-01T21:22:43Z".</p> <p>Note: Z refers to UTC/GMT time, not local time.</p> <p><i>creationTime</i> is a required attribute.</p> | 1 |

| Continuation of Table 6 | | |
|-------------------------|--|------------|
| Attribute | Description | Occurrence |
| nextSequence | <p>A number representing the <i>sequence number</i> of the piece of <i>Streaming Data</i> that is the next piece of data to be retrieved from the <i>buffer</i> of the <i>Agent</i> that was not included in the Response Document published by the <i>Agent</i>.</p> <p>If the <i>Streaming Data</i> included in the Response Document includes the last piece of data stored in the <i>buffer</i> of the <i>Agent</i> at the time that the document was published, then the value reported for nextSequence MUST be equal to lastSequence + 1.</p> <p>The value reported for nextSequence MUST be a number representing an unsigned 64-bit integer.</p> <p>nextSequence is a required attribute.</p> | 1 |
| lastSequence | <p>A number representing the <i>sequence number</i> assigned to the last piece of <i>Streaming Data</i> that was added to the <i>buffer</i> of the <i>Agent</i> immediately prior to the time that the <i>Agent</i> published the Response Document.</p> <p>The value reported for lastSequence MUST be a number representing an unsigned 64-bit integer.</p> <p>lastSequence is a required attribute.</p> | 1 |
| firstSequence | <p>A number representing the <i>sequence number</i> assigned to the oldest piece of <i>Streaming Data</i> stored in the <i>buffer</i> of the <i>Agent</i> immediately prior to the time that the <i>Agent</i> published the Response Document.</p> <p>The value reported for firstSequence MUST be a number representing an unsigned 64-bit integer.</p> <p>firstSequence is a required attribute.</p> | 1 |

| Continuation of Table 6 | | |
|-------------------------|---|------------|
| Attribute | Description | Occurrence |
| testIndicator | <p>A flag indicating that the <i>Agent</i> that published the <i>Response Document</i> is operating in a test mode. The contents of the <i>Response Document</i> may not be valid and SHOULD be used for testing and simulation purposes only.</p> <p>The values reported for <code>testIndicator</code> are:</p> <ul style="list-style-type: none"> - TRUE: The <i>Agent</i> is functioning in a test mode. - FALSE: The <i>Agent</i> is not function in a test mode. <p>If <code>testIndicator</code> is not specified, the value for <code>testIndicator</code> MUST be interpreted to be FALSE.</p> <p><code>testIndicator</code> is an optional attribute.</p> | 0..1 |
| instanceId | <p>A number indicating a specific instantiation of the <i>buffer</i> associated with the <i>Agent</i> that published the <i>Response Document</i>.</p> <p>The value reported for <code>instanceId</code> MUST be a unique unsigned 64-bit integer.</p> <p>The value for <code>instanceId</code> MUST be changed to a different unique number each time the <i>buffer</i> is cleared and a new set of data begins to be collected.</p> <p><code>instanceId</code> is a required attribute.</p> | 1 |

| Continuation of Table 6 | | |
|-------------------------|---|------------|
| Attribute | Description | Occurrence |
| sender | <p>An identification defining where the <i>Agent</i> that published the <i>Response Document</i> is installed or hosted.</p> <p>The value reported for <code>sender</code> MUST be either an IP Address or Hostname describing where the <i>Agent</i> is installed or the URL of the <i>Agent</i>; e.g., <code>http://<address>[:port]/</code>.</p> <p>Note: The port number need not be specified if it is the default HTTP port 80.</p> <p><code>sender</code> is a required attribute.</p> | 1 |
| bufferSize | <p>A value representing the maximum number of <i>Data Entities</i> that MAY be retained in the <i>Agent</i> that published the <i>Response Document</i> at any point in time.</p> <p>The value reported for <code>bufferSize</code> MUST be a number representing an unsigned 32-bit integer.</p> <p><code>bufferSize</code> is a required attribute.</p> <p>Note 1: <code>bufferSize</code> represents the maximum number of <i>sequence numbers</i> that MAY be stored in the <i>Agent</i>.</p> <p>Note 2: The implementer is responsible for allocating the appropriate amount of storage capacity required to accommodate the <code>bufferSize</code>.</p> | 1 |

1850 Example 4 is an example of a Header XML element for an *MTConnectStreams Response*
 1851 *Document*:

Example 4: Example of Header XML Element for MTConnectStreams

```

1852 1 <Header creationTime="2017-02-16T16:44:27Z"
1853 2   sender="MyAgent" instanceId="1268463594"
1854 3   bufferSize="131072" version="1.4.0.10"
1855 4   assetCount="54" assetBufferSize="1024"/>

```

1856 6.5.3 Header for MTConnectAssets

1857 The **Header** element for an *MTConnectAssets Response Document* defines information
 1858 regarding the creation of the document and the storage of *Asset Documents* in the *Agent*
 1859 that generated the document.

1860 6.5.3.1 XML Schema Structure for Header for MTConnectAssets

1861 The *XML Schema* in *Figure 19* represents the structure of the **Header** XML element that
 1862 **MUST** be provided for an *MTConnectAssets Response Document*.

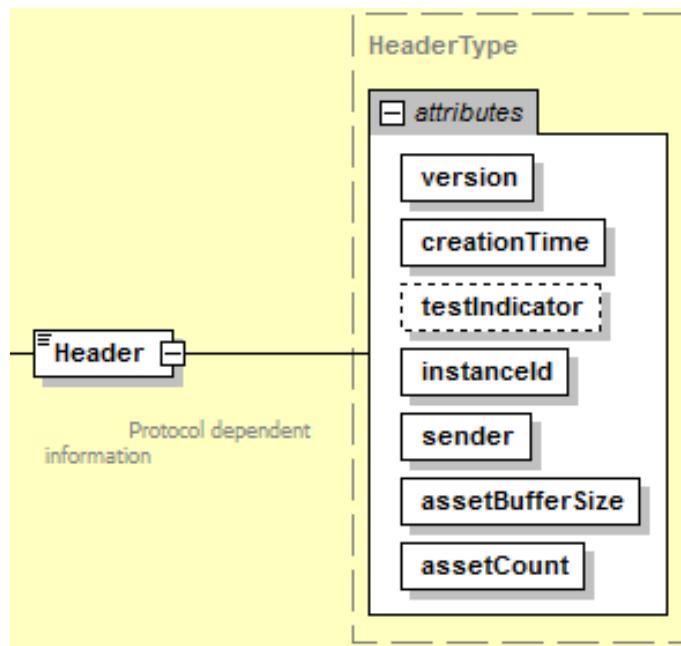


Figure 19: Header Schema Diagram for MTConnectAssets

1863 6.5.3.2 Attributes for Header for MTConnectAssets

1864 *Table 7* defines the attributes that may be used to provide additional information in the
 1865 **Header** element for an *MTConnectAssets Response Document*.

Table 7: MTConnectAssets Header

| Attribute | Description | Occurrence |
|--------------|--|------------|
| version | <p>The <i>major</i>, <i>minor</i>, and <i>revision</i> number of the MTConnect Standard that defines the <i>semantic data model</i> that represents the content of the <i>Response Document</i>. It also includes the revision number of the <i>schema</i> associated with that specific <i>semantic data model</i>.</p> <p>The value reported for version MUST be a series of four numeric values, separated by a decimal point, representing a <i>major</i>, <i>minor</i>, and <i>revision</i> number of the MTConnect Standard and the revision number of a specific <i>schema</i>.</p> <p>As an example, the value reported for version for a <i>Response Document</i> that was structured based on <i>schema</i> revision 10 associated with Version 1.4.0 of the MTConnect Standard would be: 1.4.0.10</p> <p>version is a required attribute.</p> | 1 |
| creationTime | <p>creationTime represents the time that an <i>Agent</i> published the <i>Response Document</i>.</p> <p>creationTime MUST be reported in UTC (Coordinated Universal Time) format; e.g., "2010-04-01T21:22:43Z".</p> <p>Note: Z refers to UTC/GMT time, not local time.</p> <p>creationTime is a required attribute.</p> | 1 |

| Continuation of Table 7 | | |
|-------------------------|--|------------|
| Attribute | Description | Occurrence |
| testIndicator | <p>A flag indicating that the <i>Agent</i> that published the <i>Response Document</i> is operating in a test mode. The contents of the <i>Response Document</i> may not be valid and SHOULD be used for testing and simulation purposes only.</p> <p>The values reported for <code>testIndicator</code> are:</p> <ul style="list-style-type: none"> - TRUE: The <i>Agent</i> is functioning in a test mode. - FALSE: The <i>Agent</i> is not function in a test mode. <p>If <code>testIndicator</code> is not specified, the value for <code>testIndicator</code> MUST be interpreted to be FALSE.</p> <p><code>testIndicator</code> is an optional attribute.</p> | 0..1 |
| instanceId | <p>A number indicating a specific instantiation of the <i>buffer</i> associated with the <i>Agent</i> that published the <i>Response Document</i>.</p> <p>The value reported for <code>instanceId</code> MUST be a unique unsigned 64-bit integer.</p> <p>The value for <code>instanceId</code> MUST be changed to a different unique number each time the <i>buffer</i> is cleared and a new set of data begins to be collected.</p> <p><code>instanceId</code> is a required attribute.</p> | 1 |

| Continuation of Table 7 | | |
|-------------------------|---|------------|
| Attribute | Description | Occurrence |
| sender | <p>An identification defining where the <i>Agent</i> that published the <i>Response Document</i> is installed or hosted.</p> <p>The value reported for <code>sender</code> MUST be either an IP Address or Hostname describing where the <i>Agent</i> is installed or the URL of the <i>Agent</i>; e.g., <code>http://<address>[:port]/</code>.</p> <p>Note: The port number need not be specified if it is the default HTTP port 80.</p> <p><code>sender</code> is a required attribute.</p> | 1 |
| assetBufferSize | <p>A value representing the maximum number of <i>Asset Documents</i> that can be stored in the <i>Agent</i> that published the <i>Response Document</i>.</p> <p>The value reported for <code>assetBufferSize</code> MUST be a number representing an unsigned 32-bit integer.</p> <p><code>assetBufferSize</code> is a required attribute.</p> <p>Note: The implementer is responsible for allocating the appropriate amount of storage capacity required to accommodate the <code>assetBufferSize</code>.</p> | 1 |
| assetCount | <p>A number representing the current number of <i>Asset Documents</i> that are currently stored in the <i>Agent</i> as of the <code>creationTime</code> that the <i>Agent</i> published the <i>Response Document</i>.</p> <p>The value reported for <code>assetCount</code> MUST be a number representing an unsigned 32-bit integer and MUST NOT be larger than the value reported for <code>assetBufferSize</code>.</p> <p><code>assetCount</code> is a required attribute.</p> | 1 |

1866 Example 5 is an example of a Header XML element for an *MTConnectAssets Response*
 1867 *Document*:

Example 5: Example of Header XML Element for MTConnectAssets

```

1868 1 <Header creationTime="2017-02-16T16:44:27Z"
1869 2   sender="MyAgent" instanceId="1268463594"
1870 3   version="1.4.0.10" assetCount="54"
1871 4   assetBufferSize="1024"/>

```

1872 6.5.4 Header for MTConnectError

1873 The `Header` element for an *MTConnectErrors Response Document* defines information
 1874 regarding the creation of the document and the data storage capability of the *Agent* that
 1875 generated the document.

1876 6.5.4.1 XML Schema Structure for Header for MTConnectError

1877 The *XML Schema* in *Figure 20* represents the structure of the `Header` XML element that
 1878 **MUST** be provided for an *MTConnectErrors Response Document*.

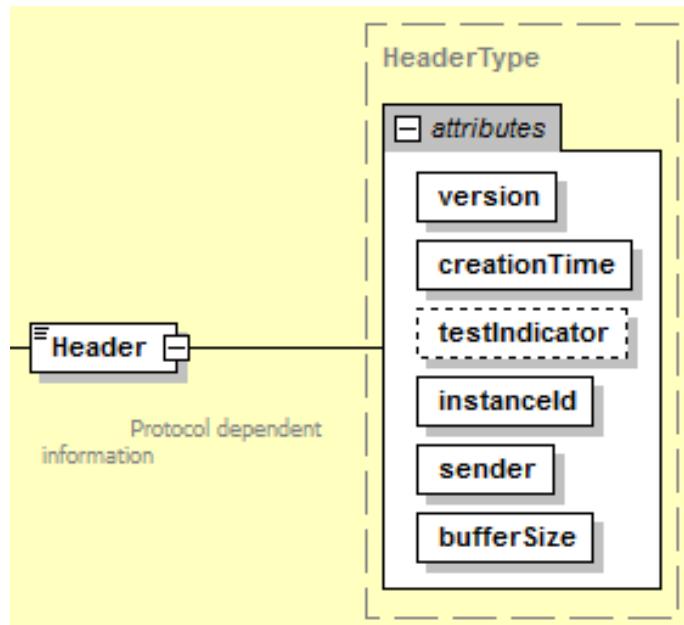


Figure 20: Header Schema Diagram for MTConnectError

1879 6.5.4.2 Attributes for Header for MTConnectError

1880 *Table 8* defines the attributes that may be used to provide additional information in the
 1881 `Header` element for an *MTConnectErrors Response Document*.

Table 8: MTConnectError Header

| Attribute | Description | Occurrence |
|--------------|--|------------|
| version | <p>The <i>major</i>, <i>minor</i>, and <i>revision</i> number of the MTConnect Standard that defines the <i>semantic data model</i> that represents the content of the <i>Response Document</i>. It also includes the revision number of the <i>schema</i> associated with that specific <i>semantic data model</i>.</p> <p>The value reported for version MUST be a series of four numeric values, separated by a decimal point, representing a <i>major</i>, <i>minor</i>, and <i>revision</i> number of the MTConnect Standard and the revision number of a specific <i>schema</i>.</p> <p>As an example, the value reported for version for a <i>Response Document</i> that was structured based on <i>schema</i> revision 10 associated with Version 1.4.0 of the MTConnect Standard would be: 1.4.0.10</p> <p>version is a required attribute.</p> | 1 |
| creationTime | <p>creationTime represents the time that an <i>Agent</i> published the <i>Response Document</i>.</p> <p>creationTime MUST be reported in UTC (Coordinated Universal Time) format; e.g., "2010-04-01T21:22:43Z".</p> <p>Note: Z refers to UTC/GMT time, not local time.</p> <p>creationTime is a required attribute.</p> | 1 |

| Continuation of Table 8 | | |
|-------------------------|--|------------|
| Attribute | Description | Occurrence |
| testIndicator | <p>A flag indicating that the <i>Agent</i> that published the <i>Response Document</i> is operating in a test mode. The contents of the <i>Response Document</i> may not be valid and SHOULD be used for testing and simulation purposes only.</p> <p>The values reported for <code>testIndicator</code> are:</p> <ul style="list-style-type: none"> - TRUE: The <i>Agent</i> is functioning in a test mode. - FALSE: The <i>Agent</i> is not function in a test mode. <p>If <code>testIndicator</code> is not specified, the value for <code>testIndicator</code> MUST be interpreted to be FALSE.</p> <p><code>testIndicator</code> is an optional attribute.</p> | 0..1 |
| instanceId | <p>A number indicating a specific instantiation of the <i>buffer</i> associated with the <i>Agent</i> that published the <i>Response Document</i>.</p> <p>The value reported for <code>instanceId</code> MUST be a unique unsigned 64-bit integer.</p> <p>The value for <code>instanceId</code> MUST be changed to a different unique number each time the <i>buffer</i> is cleared and a new set of data begins to be collected.</p> <p><code>instanceId</code> is a required attribute.</p> | 1 |

| Continuation of Table 8 | | |
|-------------------------|--|------------|
| Attribute | Description | Occurrence |
| sender | <p>An identification defining where the <i>Agent</i> that published the <i>Response Document</i> is installed or hosted.</p> <p>The value reported for <code>sender</code> MUST be either an IP Address or Hostname describing where the <i>Agent</i> is installed or the URL of the <i>Agent</i>; e.g., <code>http://<address>[:port]/</code>.</p> <p>Note: The port number need not be specified if it is the default HTTP port 80.</p> <p><code>sender</code> is a required attribute.</p> | 1 |
| bufferSize | <p>A value representing the maximum number of <i>Data Entities</i> that MAY be retained in the <i>Agent</i> that published the <i>Response Document</i> at any point in time.</p> <p>The value reported for <code>bufferSize</code> MUST be a number representing an unsigned 32-bit integer.</p> <p><code>bufferSize</code> is a required attribute.</p> <p>Note 1: <code>bufferSize</code> represents the maximum number of sequence numbers that MAY be stored in the <i>Agent</i>.</p> <p>Note 2: The implementer is responsible for allocating the appropriate amount of storage capacity required to accommodate the <code>bufferSize</code>.</p> | 1 |

1882 Example 6 is an example of a Header XML element for an *MTConnectErrors Response*
 1883 *Document*:

Example 6: Example of Header XML Element for MTConnectError

```
1884 1  <Header creationTime="2017-02-16T16:44:27Z"
1885 2    sender="MyAgent" instanceId="1268463594"
1886 3    bufferSize="131072" version="1.4.0.10"/>
```

1887 6.6 Document Body

1888 The *Document Body* contains the information that is published by an *Agent* in response
 1889 to a *Request* from a client software application. Each *Response Document* has a different
 1890 XML element that represents the *Document Body*.

1891 The structure of the content of the XML element representing the *Document Body* is de-
 1892 fined by the *semantic data models* defined for each *Response Document*.

1893 *Table 9* defines the relationship between each of the *Response Documents*, the XML ele-
 1894 ment that represents the *Document Body* for each document, and the *semantic data model*
 1895 that defines the structure for the content of each of the *Response Documents*:

Table 9: Relationship between Response Document and Semantic Data Model

| Response Document | XML Element for Document Body | Semantic Data Model |
|---|--|---|
| <i>MTConnectDevices Response Document</i> | Devices | <i>MTConnect Standard: Part 2.0 - Devices Information Model</i> |
| <i>MTConnectStreams Response Document</i> | Streams | <i>MTConnect Standard: Part 3.0 - Streams Information Model</i> |
| <i>MTConnectAssets Response Document</i> | Assets | <i>MTConnect Standard: Part 4.0 - Assets Information Model</i> |
| <i>MTConnectErrors Response Document</i> | Errors Note: Errors MUST NOT be used when backwards compatibility with MTConnect Standard Version 1.0.1 and earlier is required. | <i>MTConnect Standard Part 1.0 - Overview and Fundamentals</i> |

1896 6.7 Extensibility

1897 MTConnect is an extensible standard, which means that implementers **MAY** extend the
 1898 *Data Models* defined in the various sections of the MTConnect Standard to include in-
 1899 formation required for a specific implementation. When these *Data Models* are encoded
 1900 using XML, the methods for extending these *Data Models* are defined by the rules estab-
 1901 lished for extending any XML schema (see the W3C website for more details on extending
 1902 XML data models).

1903 The following are typical extensions that **MAY** be considered in the MTConnect *Data*
 1904 *Models*:

- 1905 • Additional `type` and `subType` values for *Data Entities*.
- 1906 • Additional *Structural Elements* as containers.
- 1907 • Additional Composition elements.
- 1908 • New *Asset* types that are sub-typed from the abstract *Asset* type.
- 1909 • *Child Elements* that may be added to specific XML elements contained within the
 1910 *MTConnect Information Models*. These extended elements **MUST** be identified in
 1911 a separate *namespace*.

1912 When extending an MTConnect *Data Model*, there are some basic rules restricting changes
 1913 to the MTConnect *Data Models*.

1914 When extending an MTConnect *Data Model*, an implementer:

- 1915 • **MUST NOT** add new value for category for *Data Entities*,
- 1916 • **MUST NOT** add new *Root Elements*,
- 1917 • **SHOULD NOT** add new *Top Level Components*, and
- 1918 • **MUST NOT** add any new attributes or include any sub-elements to Composi-
 1919 tion.

1920 Note: Throughout the documents additional information is provided where
 1921 extensibility may be acceptable or unacceptable to maintain compliance with
 1922 the MTConnect Standard.

1923 When a *schema* representing a *Data Model* is extended, the *schema* and *namespace* declaration at the beginning of the corresponding *Response Document* **MUST** be updated to
 1924 reflect the new *schema* and *namespace* so that a client software application can properly
 1925 validate the *Response Document*.

1927 An XML example of a *schema* and *namespace* declaration, including an extended *schema*
 1928 and *namespace*, is shown in *Example 7*:

Example 7: Example of extended schema and namespace in declaration

```
1929 1  <?xml version="1.0" encoding="UTF-8"?>
1930 2    <MTConnectDevices
1931 3      xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
1932 4      xmlns="urn:mtconnect.org:MTConnectDevices:1.3"
1933 5      xmlns:m="urn:mtconnect.org:MTConnectDevices:1.3"
1934 6      xmlns:x="urn:MyLocation:MyFile:MyVersion"
1935 7      xsi:schemaLocation="urn:MyLocation:MyFile:MyVersion /schemas/MyFileName.xsd" />
```

1936 In this example:

- *xmlns:x* is added in Line 6 to identify the *XML Schema* instance for the extended *schema*. *Element Names* identified with an "x" prefix are associated with this specific *XML Schema* instance.

1940 Note: The "x" prefix **MAY** be replaced with any prefix that the implementer
 1941 chooses for identifying the extended *schema* and *namespace*.

- *xsi:schemaLocation* is modified in Line 7 to associate the *namespace* URN with the URL specifying the location of *schema* file.
- *MyLocation*, *MyFile*, *MyVersion*, and *MyFileName* in Lines 6 and 7 **MUST** be replaced by the actual name, version, and location of the extended *schema*.

1946 When an extended *schema* is implemented, each *Structural Element*, *Data Entity*, and
 1947 *MTConnect Asset* defined in the extended *schema* **MUST** be identified in each respective
 1948 *Response Document* by adding a prefix to the XML *Element Name* associated with that
 1949 *Structural Element*, *Data Entity*, or *MTConnect Asset*. The prefix identifies the *schema*
 1950 and *namespace* where that XML Element is defined.

1951 7 Protocol and Messaging

1952 An *Agent* performs two *major* communications tasks. It collects information from pieces
 1953 of equipment and it publishes MTConnect *Response Documents* in response to *Requests*
 1954 from client software applications.

1955 The MTConnect Standard does not address the method used by an *Agent* to collect in-
 1956 formation from a piece of equipment. The relationship between the *Agent* and a piece of
 1957 equipment is implementation dependent. The *Agent* may be fully integrated into the piece
 1958 of equipment or the *Agent* may be independent of the piece of equipment. Implementation
 1959 of the relationship between a piece of equipment and an *Agent* is the responsibility of the
 1960 supplier of the piece of equipment and/or the implementer of the *Agent*.

1961 The communications mechanism between an *Agent* and a client software application re-
 1962 quires the following primary components:

- 1963 • *Physical Connection*: The network transmission technologies that physically inter-
 1964 connect an *Agent* and a client software application. Examples of a *Physical Con-*
 1965 *nnection* would be an Ethernet network or a wireless connection.

- 1966 • *Transport Protocol*: A set of capabilities that provide the rules and procedures used
 1967 to transport information between an *Agent* and a client software application through
 1968 a *Physical Connection*.

- 1969 • *Application Programming Interface*: The *Request* and *Response* interactions that
 1970 occur between an *Agent* and a client software application.

- 1971 • *Message*: The content of the information that is exchanged. The *Message* includes
 1972 both the content of the MTConnect *Response Document* and any additional informa-
 1973 tion required for the client software application to interpret the *Response Document*.

1974 Note: The *Physical Connections*, *Transport Protocols*, and *Application Pro-*
 1975 *gramming Interface* supported by an *Agent* are independent of the *Message* it-
 1976 self; i.e., the information contained in the MTConnect *Response Documents* is
 1977 not changed based on the methods used to transport those documents to a client
 1978 software application.

1979 An *Agent* **MAY** support multiple methods for communicating with client software ap-
 1980 plications. The MTConnect Standard specifies one methodology for communicating that
 1981 **MUST** be supported by every *Agent*. This methodology is a REST, which defines a state-
 1982 less, client-server communications architecture. This REST interface is the architectural
 1983 pattern that specifies the exchange of information between an *Agent* and a client software

1984 application. REST dictates that a server has no responsibility for tracking or coordinating
1985 with a client software application regarding which information or how much information
1986 the client software application may request from a server. This removes the burden for
1987 a server to keep track of client sessions. An *Agent* **MUST** be implemented as a server
1988 supporting the RESTful interface.

1989 8 HTTP Messaging Supported by an Agent

1990 This section describes the application of *HTTP Messaging* applied to a REST interface that
 1991 **MUST** be supported by an *Agent* to realize the MTConnect *Request/Response* information
 1992 exchange functionality.

1993 8.1 REST Interface

1994 An *Agent* **MUST** provide a REST interface that supports HTTP version 1.0 to commu-
 1995 nicate with client applications. This interface **MUST** support HTTP (RFC7230) and use
 1996 URIs (RFC3986) to identify specific information requested from an *Agent*. HTTP is most
 1997 often implemented on top of the Transmission Control Protocol (TCP) that provides an
 1998 ordered byte stream of data and the Internet Protocol (IP) that provides unified address-
 1999 ing and routing between computers. However, additional interfaces to an *Agent* may be
 2000 implemented in conjunction with any other communications technologies.

2001 The REST interface supports an *Application Programming Interface* (API) that adheres
 2002 to the architectural principles of a stateless, uniform interface to retrieve data and other
 2003 information related to either pieces of equipment or *MTConnect Assets*. The API allows
 2004 for access, but not modification of data stored within the *Agent* and is nullipotent, meaning
 2005 it will not produce any side effects on the information stored in an *Agent* or the function
 2006 of the *Agent* itself.

2007 *HTTP Messaging* is comprised of two basic functions – an *HTTP Request* and an *HTTP*
 2008 *Response*. A client software application forms a *Request* for information from an *Agent*
 2009 by specifying a specific set of information using an *HTTP Request*. In response, an *Agent*
 2010 provides either an *HTTP Response* or replies with an *HTTP Error Message* as defined
 2011 below.

2012 8.2 HTTP Request

2013 The MTConnect Standard defines that an *Agent* **MUST** support the HTTP GET verb – no
 2014 other HTTP methods are required to be supported.

2015 An *HTTP Request* **MAY** include three sections:

2016 • an *HTTP Request Line*

2017 • *HTTP Header Fields*

2018 • an *HTTP Body*

2019 The MTConnect Standard defines that an *HTTP Request* issued by a client application
2020 **SHOULD** only have two sections:

2021 • an *HTTP Request Line*

2022 • *HTTP Header Fields*

2023 The *HTTP Request Line* identifies the specific information being requested by the client
2024 software application. If an *Agent* receives any information in an *HTTP Request* that is not
2025 specified in the MTConnect Standard, the *Agent* **MAY** ignore it.

2026 The structure of an *HTTP Request Line* consists of the following portions:

2027 • *HTTP Request Method*: GET

2028 • *HTTP Request URL*: http://<authority>/<path>[?<query>]

2029 • *HTTP Version*: HTTP/1.0

2030 For the following discussion, the *HTTP Request URL* will only be considered since the
2031 Method will always be GET and the MTConnect Standard only requires HTTP/1.0.

2032 8.2.1 authority Portion of an HTTP Request Line

2033 The authority portion consists of the DNS name or IP address associated with an
2034 *Agent* and an optional TCP port number [:port] that the *Agent* is listening to for incoming
2035 *Requests* from client software applications. If the port number is the default Port 80, port
2036 is not required.

2037 Example forms for authority are:

2038 • http://machine/

2039 • http://machine:5000/

2040 • http://192.168.1.2:5000/

2041 **8.2.2 path Portion of an HTTP Request Line**

2042 The <Path> portion of the *HTTP Request Line* has the follow segments:

- 2043 • /<name or uuid>/<request>

2044 In this portion of the *HTTP Request Line*, name or uuid designates that the information to
2045 be returned in a *Response Document* is associated with a specific piece of equipment that
2046 has published data to the *Agent*. See Part 2 - *Devices Information Model* for details on
2047 name or uuid for a piece of equipment.

2048 Note: If name or uuid are not specified in the *HTTP Request Line*, an *Agent* **MUST**
2049 return the information for all pieces of equipment that have published data to
2050 the *Agent* in the *Response Document*.

2051 In the <Path> portion of the *HTTP Request Line*, <request> designates one of the
2052 *Requests* defined in *Section 5.4 - Request/Response Information Exchange*. The value
2053 for <request> **MUST** be probe, current, sample, or asset(s) representing the
2054 *Probe Request*, *Current Request*, *Sample Request*, and *Asset Request* respectively.

2055 **8.2.3 query Portion of an HTTP Request Line**

2056 The [?<query>] portion of the *HTTP Request Line* designates an *HTTP Query*. *Query* is
2057 a string of parameters that define filters used to refine the content of a *Response Document*
2058 published in response to an *HTTP Request*.

2059 **8.3 MTConnect Request/Response Information Exchange Implemented
2060 with HTTP**

2061 An *Agent* **MUST** support *Probe Requests*, *Current Requests*, *Sample Requests*, and *Asset
2062 Requests*.

2063 The following sections define how the *HTTP Request Line* is structured to support each of
2064 these types of *Requests* and the information that an *Agent* **MUST** provide in response to
2065 these *Requests*.

2066 8.3.1 Probe Request Implemented Using HTTP

2067 An *Agent* responds to a *Probe Request* with an *MTConnectDevices Response Document*
 2068 that contains the *Equipment Metadata* for pieces of equipment that are requested and cur-
 2069 rently represented in the *Agent*.

2070 There are two forms of the *Probe Request*:

- 2071 • The first form includes an *HTTP Request Line* that does not specify a specific path
 2072 portion (name or uuid). In response to this *Request*, the *Agent* returns an *MT-*
 2073 *ConnectDevices Response Document* with information for all pieces of equipment
 2074 represented in the *Agent*.

2075 1. `http://<authority>/probe`

- 2076 • The second form includes an *HTTP Request Line* that specifies a specific path por-
 2077 tion that defines either a name or uuid. In response to this *Request*, the *Agent*
 2078 returns an *MTConnectDevices Response Document* with information for only the
 2079 one piece of equipment associated with that name or uuid.

2080 1. `http://<authority>/<name or uuid>/probe`

2081 8.3.1.1 Path Portion of the HTTP Request Line for a Probe Request

2082 The following segments of path **MUST** be supported in an *HTTP Request Line* for a
 2083 *Probe Request*:

Table 10: Path of the HTTP Request Line for a Probe Request

| Path Segments | Description |
|---------------|--|
| name or uuid | If present, specifies that only the <i>Equipment Metadata</i> for the piece of equipment represented by the name or uuid will be published. If not present, <i>Metadata</i> for all pieces of equipment associated with the <i>Agent</i> will be published. |
| <request> | probe MUST be provided. |

2084 8.3.1.2 Query Portion of the HTTP Request Line for a Probe Request

2085 The *HTTP Request Line* for a *Probe Request* **SHOULD NOT** contain a query. If the

2086 *Request* does contain a `query`, the **Agent MUST** ignore the `query`.

2087 **8.3.1.3 Response to a Probe Request**

2088 The *Response* to a *Probe Request* **SHOULD** be an *MTConnectDevices Response Document* for one or more pieces of equipment as designated by the path portion of the
2089 *Request*.

2091 The *Response Document* returned in response to a *Probe Request* **MUST** always provide
2092 the most recent information available to an **Agent**.

2093 The *Response* **MUST** also include an *HTTP Status Code*. If problems are encountered by
2094 an **Agent** while responding to a *Probe Request*, the **Agent MUST** also publish an *MTConnectErrors Response Document*.
2095

2096 **8.3.1.4 HTTP Status Codes for a Probe Request**

2097 The following *HTTP Status Codes* **MUST** be supported as possible responses to a *Probe Request*:

Table 11: HTTP Status Codes for a Probe Request

| HTTP Status Code | Code Name | Description |
|------------------|-------------|--|
| 200 | OK | The <i>Request</i> was handled successfully. |
| 400 | Bad Request | The <i>Request</i> could not be interpreted. The Agent MUST return a 400 <i>HTTP Status Code</i> . Also, the Agent MUST publish an <i>MTConnectErrors Response Document</i> that identifies either <code>INVALID_URI</code> or <code>INVALID_REQUEST</code> as the <code>errorCode</code> . |
| 404 | Not Found | The <i>Request</i> could not be interpreted. The Agent MUST return a 404 <i>HTTP Status Code</i> . Also, the Agent MUST publish an <i>MTConnectErrors Response Document</i> that identifies <code>NO_DEVICE</code> as the <code>errorCode</code> . |

| Continuation of Table 11 | | |
|--------------------------|---------------------------------|---|
| HTTP Status Code | Code Name | Description |
| 405 | Method Not Allowed | <p>A method other than GET was specified in the <i>Request</i> or the piece of equipment specified in the <i>Request</i> could not be found.</p> <p>The <i>Agent</i> MUST return a 405 <i>HTTP Status Code</i>. Also, the <i>Agent</i> MUST publish an <i>MTConnectErrors Response Document</i> that identifies UNSUPPORTED as the errorCode.</p> |
| 406 | Not Acceptable | <p>The <i>HTTP Accept Header</i> in the <i>Request</i> was not one of the supported representations.</p> <p>The <i>Agent</i> MUST return a 406 <i>HTTP Status Code</i>. Also, the <i>Agent</i> MUST publish an <i>MTConnectErrors Response Document</i> that identifies UNSUPPORTED as the errorCode.</p> |
| 431 | Request Header Fields Too Large | <p>The fields in the <i>HTTP Request</i> exceed the limit of the implementation of the <i>Agent</i>.</p> <p>The <i>Agent</i> MUST return a 431 <i>HTTP Status Code</i>. Also, the <i>Agent</i> MUST publish an <i>MTConnectErrors Response Document</i> that identifies INVALID_REQUEST as the errorCode.</p> |
| 500 | Internal Server Error | <p>There was an unexpected error in the <i>Agent</i> while responding to a <i>Request</i>.</p> <p>The <i>Agent</i> MUST return a 500 <i>HTTP Status Code</i>. Also, the <i>Agent</i> MUST publish an <i>MTConnectErrors Response Document</i> that identifies INTERNAL_ERROR as the errorCode.</p> |

2099 8.3.2 Current Request Implemented Using HTTP

2100 An *Agent* responds to a *Current Request* with an *MTConnectStreams Response Document*
 2101 that contains the current value of *Data Entities* associated with each piece of *Streaming*
 2102 *Data* available from the *Agent*, subject to any filtering defined in the *Request*.

2103 There are two forms of the *Current Request*:

- 2104 • The first form is given without a specific path portion (name or uuid). In response
 2105 to this *Request*, the *Agent* returns an *MTConnectStreams Response Document* with
 2106 information for all pieces of equipment represented in the *buffer* of the *Agent*.

2107 1. `http://<authority>/current [?query]`

- 2108 • The second form includes a specific path portion that defines either a name or uuid.
 2109 In response to this *Request*, the *Agent* returns an *MTConnectStreams Response Doc-
 2110 ument* with information for only the one piece of equipment associated with the
 2111 name or uuid defined in the *Request*.

2112 1. `http://<authority>/<name or uuid>/current [?query]`

2113 8.3.2.1 Path Portion of the HTTP Request Line for a Current Request

2114 The following segments of path **MUST** be supported for an *HTTP Request Line* for a
 2115 *Current Request*:

Table 12: Path of the HTTP Request Line for a Current Request

| Path Segments | Description |
|---------------|--|
| name or uuid | If present, specifies that only the <i>Equipment Metadata</i> for the piece of equipment represented by the name or uuid will be published. If not present, <i>Metadata</i> for all pieces of equipment associated with the <i>Agent</i> will be published. |
| <request> | current MUST be provided. |

2116 8.3.2.2 Query Portion of the HTTP Request Line for a Current Request

2117 A *Query* may be used to more precisely define the specific information to be included
 2118 in a *Response Document*. Multiple parameters may be used in a *Query* to further refine

2119 the information to be included. When multiple parameters are provided, each parameter
 2120 is separated by an ampersand (&) character and each parameter appears only once in the
 2121 *Query*. The parameters within the *Query* may appear in any sequence.

2122 The following `query` parameters **MUST** be supported in an *HTTP Request Line* for a
 2123 *Current Request*:

Table 13: Query Parameters of the HTTP Request Line for a Current Request

| Query Parameters | Description |
|------------------|---|
| path | <p>An XPath that defines specific information or a set of information to be included in an <i>MTConnectStreams Response Document</i>.</p> <p>The value for the XPath is the location of the information defined in the <i>Devices Information Model</i> that represents the <i>Structural Element(s)</i> and/or the specific <i>Data Entities</i> to be included in the <i>MTConnectStreams Response Document</i>.</p> <p>When a <i>Component</i> element is referenced by the XPath, all <i>Lower Level</i> components and the <i>Data Entities</i> associated with those elements MUST be included in the <i>MTConnectStreams Response Document</i>.</p> |

| Continuation of Table 13 | |
|--------------------------|--|
| Query Parameters | Description |
| at | <p>Requests that the <i>MTConnect Response Documents</i> MUST include the current value for all <i>Data Entities</i> relative to the time that a specific <i>sequence number</i> was recorded.</p> <p>The value associated with the <i>at</i> parameter references a specific <i>sequence number</i>. The value MUST be an unsigned 64-bit value.</p> <p>The <i>at</i> parameter MUST NOT be used in conjunction with the <i>interval</i> parameter since this would cause an <i>Agent</i> to repeatedly return the same data.</p> <p>If the value provided for the <i>at</i> parameter is a negative number or is not a, the <i>Request</i> MUST be determined to be invalid. The <i>Agent</i> MUST return a 400 <i>HTTP Status Code</i>. Also, the <i>Agent</i> MUST publish an <i>MTConnectErrors Response Document</i> that identifies an <code>INVALID_REQUEST</code> <i>errorCode</i>.</p> <p>If the value provided for the <i>at</i> parameter is either lower than the value of <i>firstSequence</i> or greater than the value of <i>lastSequence</i>, the <i>Request</i> MUST be determined to be invalid. The <i>Agent</i> MUST return a 404 <i>HTTP Status Code</i>. The <i>Agent</i> MUST also publish an <i>MTConnectErrors Response Document</i> that identifies an <code>OUT_OF_RANGE</code> <i>errorCode</i>.</p> <p>Note: Some information stored in the <i>buffer</i> of an <i>Agent</i> may not be returned for a <i>Current Request</i> with a <i>Query</i> containing an <i>at</i> parameter if the <i>sequence number</i> associated with the most current value for that information is greater than the <i>sequence number</i> specified in the <i>Query</i>.</p> |

| Continuation of Table 13 | |
|--------------------------|---|
| Query Parameters | Description |
| interval | <p>When a <i>Current Request</i> includes a <i>Query</i> with the <code>interval</code> parameter, an <i>Agent</i> MUST respond to this <i>Request</i> by repeatedly publishing the required Response Document at the time <code>interval</code> (period) defined by the value provided for the <code>interval</code> parameter.</p> <p>The value provided for <code>interval</code> MUST be expressed in milliseconds and MUST be a positive value greater than 0.</p> <p>The <code>interval</code> parameter MUST NOT be used in conjunction with the <code>at</code> parameter since this would cause an <i>Agent</i> to repeatedly return the same data.</p> <p>If a <i>Request</i> contains a <i>Query</i> with an <code>interval</code> parameter, it MUST remain in effect until the client software application terminates its connection to the <i>Agent</i>.</p> |

2124 8.3.2.3 Response to a Current Request

2125 The *Response* to a *Current Request* **SHOULD** be an *MTConnectStreams Response Document* for one or more pieces of equipment designated by the `path` portion of the *Request*.

2127 The *Response* to a *Current Request* **MUST** always provide the most recent information
2128 available to an *Agent* or, when the `at` parameter is specified, the value of the data at the
2129 given *sequence number*.

2130 The *Data Entities* provided in the *MTConnectStreams Response Document* will be limited
2131 to those specified in the combination of the `path` segment of the *Current Request* and the
2132 value of the XPath defined for the `path` attribute provided in the `query` segment of that
2133 *Request*.

2134 8.3.2.4 HTTP Status Codes for a Current Request

2135 The following *HTTP Status Codes* **MUST** be supported as possible responses to a *Current*
2136 *Request*:

Table 14: HTTP Status Codes for a Current Request

| HTTP Status Code | Code Name | Description |
|------------------|--------------------|---|
| 200 | OK | The <i>Request</i> was handled successfully. |
| 400 | Bad Request | <p>The <i>Request</i> could not be interpreted.</p> <p>The Agent MUST return a 400 <i>HTTP Status Code</i>. Also, the Agent MUST publish an <i>MTConnectErrors Response Document</i> that identifies either INVALID_URI, INVALID_REQUEST, or INVALID_XPATH as the errorCode.</p> <p>If the query parameters do not contain a valid value or include an invalid parameter, the Agent MUST return a 400 <i>HTTP Status Code</i>. Also, the Agent MUST publish an <i>MTConnectErrors Response Document</i> that identifies QUERY_ERROR as the errorCode.</p> |
| 404 | Not Found | <p>The <i>Request</i> could not be interpreted.</p> <p>The Agent MUST return a 404 <i>HTTP Status Code</i>. Also, the Agent MUST publish an <i>MTConnectErrors Response Document</i> that identifies NO_DEVICE as the errorCode.</p> <p>If the value of the at parameter was greater than the lastSequence or is less than the firstSequence, the Agent MUST return a 404 <i>HTTP Status Code</i>. Also, the Agent MUST publish an <i>MTConnectErrors Response Document</i> that identifies OUT_OF_RANGE as the errorCode.</p> |
| 405 | Method Not Allowed | <p>A method other than GET was specified in the <i>Request</i> or the piece of equipment specified in the <i>Request</i> could not be found.</p> <p>The Agent MUST return a 405 <i>HTTP Status Code</i>. Also, the Agent MUST publish an <i>MTConnectErrors Response Document</i> that identifies UNSUPPORTED as the errorCode.</p> |

| Continuation of Table 14 | | |
|--------------------------|---------------------------------|--|
| HTTP Status Code | Code Name | Description |
| 406 | Not Acceptable | <p>The <i>HTTP Accept Header</i> in the <i>Request</i> was not one of the supported representations.</p> <p>The <i>Agent MUST</i> return a 406 <i>HTTP Status Code</i>. Also, the <i>Agent MUST</i> publish an <i>MTConnectErrors Response Document</i> that identifies UNSUPPORTED as the <i>errorCode</i>.</p> |
| 431 | Request Header Fields Too Large | <p>The fields in the <i>HTTP Request</i> exceed the limit of the implementation of the <i>Agent</i>.</p> <p>The <i>Agent MUST</i> return a 431 <i>HTTP Status Code</i>. Also, the <i>Agent MUST</i> publish an <i>MTConnectErrors Response Document</i> that identifies INVALID_REQUEST as the <i>errorCode</i>.</p> |
| 500 | Internal Server Error | <p>There was an unexpected error in the <i>Agent</i> while responding to a <i>Request</i>.</p> <p>The <i>Agent MUST</i> return a 500 <i>HTTP Status Code</i>. Also, the <i>Agent MUST</i> publish an <i>MTConnectErrors Response Document</i> that identifies INTERNAL_ERROR as the <i>errorCode</i>.</p> |

2137 8.3.3 Sample Request Implemented Using HTTP

2138 An *Agent* responds to a *Sample Request* with an *MTConnectStreams Response Document*
 2139 that contains a set of values for *Data Entities* currently available for *Streaming Data* from
 2140 the *Agent*, subject to any filtering defined in the *Request*.

2141 There are two forms to the *Sample Request*:

- 2142 • The first form is given without a specific path portion (name or uuid). In re-
 2143 sponse to this *Request*, the *Agent* returns an *MTConnectStreams Response Docu-*
 2144 *ment* with information for all pieces of equipment represented in the *Agent*.

2145 1. `http://<authority>/sample[?query]`

- 2146 • The second form includes a specific path portion that defines either a name or
 2147 uuid.

2148 In response to this *Request*, the *Agent* returns an *MTConnectStreams Response Doc-*
 2149 *ument* with information for only the one piece of equipment associated with the
 2150 name or uuid defined in the *Request*.

2151 1. http://<authority>/<name or uuid>/sample?query

2152 8.3.3.1 Path Portion of the HTTP Request Line for a Sample Request

2153 The following segments of path **MUST** be supported in the *HTTP Request Line* for a
 2154 *Sample Request*:

Table 15: Path of the HTTP Request Line for a Sample Request

| Path Segments | Description |
|---------------|--|
| name or uuid | If present, specifies that only the <i>Equipment Metadata</i> for the piece of equipment represented by the name or uuid will be published. If not present, <i>Metadata</i> for all pieces of equipment associated with the <i>Agent</i> will be published. |
| <request> | sample MUST be provided. |

2155 8.3.3.2 Query Portion of the HTTP Request Line for a Sample Request

2156 A *Query* may be used to more precisely define the specific information to be included
 2157 in a *Response Document*. Multiple parameters may be used in a *Query* to further refine
 2158 the information to be included. When multiple parameters are provided, each parameter
 2159 is separated by an & character and each parameter appears only once in the *Query*. The
 2160 parameters within the *Query* may appear in any sequence.

2161 The following query parameters **MUST** be supported in an *HTTP Request Line* for a
 2162 *Sample Request*:

Table 16: Query Parameters of the HTTP Request Line for a Sample Request

| Query Parameters | Description |
|------------------|---|
| path | <p>An XPath that defines specific information or a set of information to be included in an <i>MTConnectStreams Response Document</i>.</p> <p>The value for the XPath is the location of the information defined in the <i>Devices Information Model</i> that represents the <i>Structural Element(s)</i> and/or the specific <i>Data Entities</i> to be included in the <i>MTConnectStreams Response Document</i>.</p> <p>When a <i>Component</i> element is referenced by the XPath, all <i>Lower Level</i> components and the <i>Data Entities</i> associated with those elements MUST be included in the <i>MTConnectStreams Response Document</i>.</p> |

| Continuation of Table 16 | |
|--------------------------|--|
| Query Parameters | Description |
| from | <p>The <code>from</code> parameter designates the <i>sequence number</i> of the first <i>Data Entity</i> in the <i>buffer</i> of the <i>Agent</i> that MUST be included in the <i>Response Document</i>.</p> <p>The value for <code>from</code> MUST be an unsigned 64-bit integer.</p> <p>The <code>from</code> parameter is typically provided in conjunction with the <code>count</code> parameter. However, this is not required.</p> <p>If the <i>sequence number</i> provided as the value for the <code>from</code> parameter is 0, the information provided in the <i>Response Document</i> MUST be provided starting with the information located in the <i>buffer</i> of an <i>Agent</i> defined by <code>firstSequence</code>.</p> <p>If no <i>sequence number</i> is provided as the value for the <code>from</code> parameter, the information provided in the <i>Response Document</i> MUST be provided starting with the information located in the <i>buffer</i> of an <i>Agent</i> defined by <code>firstSequence</code>.</p> <p>If the <i>sequence number</i> provided as the value for the <code>from</code> parameter is a negative number, the request MUST be determined to be invalid and the <i>Agent</i> MUST return a 400 <i>HTTP Status Code</i>. Also, the <i>Agent</i> MUST publish an <i>MTConnectErrors Response Document</i> that identifies an <code>INVALID_REQUEST</code> <code>errorCode</code>.</p> <p>If the value provided for the <code>from</code> parameter is either lower than the value of <code>firstSequence</code> or greater than the value of <code>lastSequence</code>, the request MUST be determined to be invalid and the <i>Agent</i> MUST return a 404 <i>HTTP Status Code</i>. Also, the <i>Agent</i> MUST publish an <i>MTConnectErrors Response Document</i> that identifies an <code>OUT_OF_RANGE</code> <code>errorCode</code>.</p> |

| Continuation of Table 16 | |
|--------------------------|--|
| Query Parameters | Description |
| interval | <p>When a <i>Sample Request</i> includes a <i>Query</i> with the <code>interval</code> parameter, an <i>Agent</i> MUST respond to this <i>Request</i> by repeatedly publishing the required <i>Response Document</i> at the time interval (period) defined by the value provided for the <code>interval</code> parameter.</p> <p>The value provided for <code>interval</code> MUST be expressed in milliseconds and MUST be a positive value greater than 0.</p> <p>The <code>interval</code> parameter MUST NOT be used in conjunction with the <code>at</code> parameter since this would cause an <i>Agent</i> to repeatedly return the same data.</p> <p>If the value for the <code>interval</code> parameter is 0, the <i>Agent</i> MUST provide successive <i>Response Documents</i> at the fastest rate that the <i>Agent</i> can support.</p> <p>If a <code>count</code> parameter is not provided in conjunction with an <code>interval</code> parameter, an <i>Agent</i> SHOULD use a default value of 100 for <code>count</code>.</p> <p>If a <i>Request</i> contains a <i>Query</i> with an <code>interval</code> parameter, it MUST remain in effect until the client software application terminates its connection to the <i>Agent</i>.</p> <p>An <i>Agent</i> MUST NOT publish a <i>Response Document</i> if no new data associated with the <i>Response Document</i> is available in the <i>buffer</i>. However, if new data associated with the <i>Response Document</i> is received by the <i>Agent</i> at a point in time after the value of the <code>interval</code> parameter is exceeded, the <i>Agent</i> MUST then publish a new version of the <i>Response Document</i> immediately.</p> |

| Continuation of Table 16 | |
|--------------------------|---|
| Query Parameters | Description |
| count | <p>The <code>count</code> parameter designates the total number of <i>Data Entities</i> to be published from the <i>buffer</i> of the <i>Agent</i> in the <i>Response Document</i>.</p> <p>The <code>count</code> parameter is typically provided in conjunction with the <code>from</code> parameter. However, this is not required.</p> <p>If the value provided for the <code>count</code> parameter defines information located in the <i>buffer</i> of an <i>Agent</i> that would be a <i>sequence number</i> greater than the value of <code>lastSequence</code>, the information provided MUST be limited only to the information available in the <i>buffer</i>.</p> <p>If no value is provided for the <code>count</code> parameter, the information provided in the <i>Response Document</i> MUST default to <code>count=100</code>.</p> <p>If the value provided for the <code>count</code> parameter is 0 or a negative number, the request MUST be determined to be invalid. The <i>Agent</i> must return a 400 <i>HTTP Status Code</i>. Also, the <i>Agent</i> MUST publish an <i>MTConnectErrors Response Document</i> that identifies an <code>INVALID_REQUEST</code> <code>errorCode</code>.</p> |
| heartbeat | <p>Sets the time period for the <i>heartbeat</i> function in an <i>Agent</i>.</p> <p>The value for <code>heartbeat</code> represents the amount of time after a <i>Response Document</i> has been published until a new <i>Response Document</i> MUST be published, even when no new data is available.</p> <p>The value for <code>heartbeat</code> is defined in milliseconds.</p> <p>If no value is defined for <code>heartbeat</code>, the value SHOULD default to 10 seconds.</p> <p><code>heartbeat</code> MUST only be specified if <code>interval</code> is also specified.</p> |

2163 8.3.3.3 Response to a Sample Request

2164 The *Response* to a *Sample Request* **SHOULD** be an *MTConnectStreams Response Document* for one or more pieces of equipment designated by the path portion of the *Request*.

2166 The *Response* to a *Sample Request* **MUST** always provide the most recent information

2167 available to an *Agent* or, when the `at` parameter is specified, the value of the data at the
 2168 given *sequence number*.

2169 The *Data Entities* provided in the *MTConnectStreams Response Document* will be limited
 2170 to those specified in the combination of the path segment of the *Sample Request* and the
 2171 value of the XPath defined for the path attribute provided in the query segment of that
 2172 *Request*.

2173 When the value of `from` references the value of the next *sequence number* (`nextSequence`) and there are no additional *Data Entities* available in the buffer, the response
 2174 document will have an empty `<Streams/>` element in the *MTConnectStreams* doc-
 2175 ument to indicate no data is available at the point in time that the *Agent* published the
 2176 *Response Document*.

2178 **8.3.3.4 HTTP Status Codes for a Sample Request**

2179 The following *HTTP Status Codes* **MUST** be supported as possible responses to a *Sample*
 2180 *Request*:

Table 17: HTTP Status Codes for a Sample Request

| HTTP Status Code | Code Name | Description |
|------------------|-------------|--|
| 200 | OK | The <i>Request</i> was handled successfully. |
| 400 | Bad Request | <p>The <i>Request</i> could not be interpreted.</p> <p>The <i>Agent</i> MUST return a 400 <i>HTTP Status Code</i>. Also, the <i>Agent</i> MUST publish an <i>MTConnectErrors Response Document</i> that identifies either <code>INVALID_URI</code>, <code>INVALID_REQUEST</code>, or <code>INVALID_XPATH</code> as the <code>errorCode</code>.</p> <p>If the <code>query</code> parameters do not contain a valid value or include an invalid parameter, the <i>Agent</i> MUST return a 400 <i>HTTP Status Code</i>. Also, the <i>Agent</i> MUST publish an <i>MTConnectErrors Response Document</i> that identifies <code>QUERY_ERROR</code> as the <code>errorCode</code>.</p> |

| Continuation of Table 17 | | |
|--------------------------|---------------------------------|---|
| HTTP Status Code | Code Name | Description |
| 404 | Not Found | <p>The <i>Request</i> could not be interpreted.</p> <p>The Agent MUST return a 404 <i>HTTP Status Code</i>. Also, the Agent MUST publish an <i>MTConnectErrors Response Document</i> that identifies NO_DEVICE as the <i>errorCode</i>.</p> <p>If the value of the <i>at</i> parameter was greater than the <i>lastSequence</i> or is less than the <i>firstSequence</i>, the Agent MUST return a 404 <i>HTTP Status Code</i>. Also, the Agent MUST publish an <i>MTConnectErrors Response Document</i> that identifies OUT_OF_RANGE as the <i>errorCode</i>.</p> |
| 405 | Method Not Allowed | <p>A method other than GET was specified in the <i>Request</i> or the piece of equipment specified in the <i>Request</i> could not be found.</p> <p>The Agent MUST return a 405 <i>HTTP Status Code</i>. Also, the Agent MUST publish an <i>MTConnectErrors Response Document</i> that identifies UNSUPPORTED as the <i>errorCode</i>.</p> |
| 406 | Not Acceptable | <p>The <i>HTTP Accept Header</i> in the <i>Request</i> was not one of the supported representations.</p> <p>The Agent MUST return a 406 <i>HTTP Status Code</i>. Also, the Agent MUST publish an <i>MTConnectErrors Response Document</i> that identifies UNSUPPORTED as the <i>errorCode</i>.</p> |
| 431 | Request Header Fields Too Large | <p>The fields in the <i>HTTP Request</i> exceed the limit of the implementation of the Agent.</p> <p>The Agent MUST return a 431 <i>HTTP Status Code</i>. Also, the Agent MUST publish an <i>MTConnectErrors Response Document</i> that identifies INVALID_REQUEST as the <i>errorCode</i>.</p> |

| Continuation of Table 17 | | |
|--------------------------|-----------------------|--|
| HTTP Status Code | Code Name | Description |
| 500 | Internal Server Error | <p>There was an unexpected error in the <i>Agent</i> while responding to a <i>Request</i>.</p> <p>The <i>Agent</i> MUST return a 500 <i>HTTP Status Code</i>. Also, the <i>Agent</i> MUST publish an <i>MTConnectErrors Response Document</i> that identifies INTERNAL_ERROR as the errorCode.</p> |

2181 8.3.4 Asset Request Implemented Using HTTP

2182 An *Agent* responds to an *Asset Request* with an *MTConnectAssets Response Document* that contains information for *MTConnect Assets* from the *Agent*, subject to any filtering
2183 defined in the *Request*.

2185 There are multiple forms to the *Asset Request*:

2186 • The first form is given without a specific path portion (name or uuid). In re-
2187 sponse to this *Request*, the *Agent* returns an *MTConnectAssets Response Document*
2188 that contains information for all *Asset Document* represented in the *Agent*.

2189 1. `http://<authority>/assets`

2190 • The second form includes a specific path portion that defines the identity (as-
2191 set_id) for one or more specific *Asset Documents*. In response to this *Request*,
2192 the *Agent* returns an *MTConnectAssets Response Document* that contains informa-
2193 tion for the specific Assets represented in the *Agent* and defined by each of the
2194 asset_id values provided in the *Request*. Each asset_id is separated by a ";".

2195 1. `http://<authority>/asset/asset_id;asset_id;asset_id....`

2196 Note: An *HTTP Request Line* may include combinations of path and query to
2197 achieve the desired set of *Asset Documents* to be included in a specific *MT-*
2198 *ConnectAssets Response Document*.

2199 **8.3.4.1 Path Portion of the HTTP Request Line for an Asset Request**

2200 The following segments of path **MUST** be supported in the *HTTP Request Line* for an
 2201 *Asset Request*:

Table 18: Path of the HTTP Request Line for an Asset Request

| Path Segments | Description |
|---------------|--|
| <request> | asset or assets MUST be provided. |
| asset_id | Identifies the id attribute of an <i>MTConnect Asset</i> to be provided by an <i>Agent</i> . |

2202 **8.3.4.2 Query Portion of the HTTP Request Line for an Asset Request**

2203 A *Query* may be used to more precisely define the specific information to be included
 2204 in a *Response Document*. Multiple parameters may be used in a *Query* to further refine
 2205 the information to be included. When multiple parameters are provided, each parameter
 2206 is separated by an & character and each parameter appears only once in the *Query*. The
 2207 parameters within the *Query* may appear in any sequence.

2208 The following *query* parameters **MUST** be supported in an *HTTP Request Line* for an
 2209 *Asset Request*:

Table 19: Query Parameters of the HTTP Request Line for an Asset Request

| Query Parameters | Description |
|------------------|--|
| type | <p>Defines the type of <i>MTConnect Asset</i> to be returned in the <i>MTConnectAssets Response Document</i>.</p> <p>The type for an <i>Asset</i> is the term used in the <i>Asset Information Model</i> to describe different types of <i>Assets</i>. It is the term that is substituted for the <i>Asset</i> container and describes the highest-level element in the <i>Asset</i> hierarchy. See <i>MTConnect Standard: Part 4.0 - Assets Information Model</i>, Section 3.2.3 for more information on the type of an <i>Asset</i>.</p> |

| Continuation of Table 19 | |
|--------------------------|---|
| Query Parameters | Description |
| removed | <p><i>Assets</i> can have an attribute that indicates whether the <i>Asset</i> has been removed from a piece of equipment.</p> <p>The valid values for removed are <code>true</code> or <code>false</code>.</p> <p>If the value of the <code>removed</code> parameter in the query is <code>true</code>, then <i>Asset Documents</i> for <i>Assets</i> that have been marked as removed from a piece of equipment will be included in the <i>Response Document</i>.</p> <p>If the value of the <code>removed</code> parameter in the query is <code>false</code>, then <i>Asset Documents</i> for <i>Assets</i> that have been marked as removed from a piece of equipment will not be included in the <i>Response Document</i>.</p> <p>If <code>removed</code> is not defined in a query, the default value for <code>removed</code> MUST be determined to be <code>false</code>.</p> |
| count | <p>Defines the maximum number of <i>Asset Documents</i> to return in an <i>MTConnectAssets Response Document</i>.</p> <p>If <code>count</code> is not defined in the query, the default value for <code>count</code> MUST be determined to be 100.</p> |

2210 8.3.4.3 Response to an Asset Request

2211 The *Response to an Asset Request* **SHOULD** be an *MTConnectAssets Response Document* containing information for one or more *Asset Documents* designated by the *Request*. The *Response to an Asset Request* **MUST** always provide the most recent information available to an *Agent*.

2215 The *Asset Documents* provided in the *MTConnectAssets Response Document* will be limited to those specified in the combination of the *path* segment of the *Asset Request* and the parameters provided in the *query* segment of that *Request*.

2218 If the `removed` query parameter is not provided with a value of `true`, *Asset Documents* for *Assets* that have been marked as removed will not be provided in the response.

2220 **8.3.4.4 HTTP Status Codes for a Asset Request**

2221 The following *HTTP Status Codes* **MUST** be supported as possible responses to an *Asset Request*:

Table 20: HTTP Status Codes for an Asset Request

| HTTP Status Code | Code Name | Description |
|------------------|--------------------|--|
| 200 | OK | The <i>Request</i> was handled successfully. |
| 400 | Bad Request | <p>The <i>Request</i> could not be interpreted.</p> <p>The Agent MUST return a 400 <i>HTTP Status Code</i>. Also, the Agent MUST publish an <i>MTConnectErrors Response Document</i> that identifies either <code>INVALID_URI</code> or <code>INVALID_REQUEST</code> as the <code>errorCode</code>.</p> <p>If the query parameters do not contain a valid value or include an invalid parameter, the Agent MUST return a 400 <i>HTTP Status Code</i>. Also, the Agent MUST publish an <i>MTConnectErrors Response Document</i> that identifies <code>QUERY_ERROR</code> as the <code>errorCode</code>.</p> |
| 404 | Not Found | <p>The <i>Request</i> could not be interpreted.</p> <p>The Agent MUST return a 404 <i>HTTP Status Code</i>. Also, the Agent MUST publish an <i>MTConnectErrors Response Document</i> that identifies <code>NO_DEVICE</code> or <code>ASSET_NOT_FOUND</code> as the <code>errorCode</code>.</p> |
| 405 | Method Not Allowed | <p>A method other than GET was specified in the <i>Request</i> or the piece of equipment specified in the <i>Request</i> could not be found.</p> <p>The Agent MUST return a 405 <i>HTTP Status Code</i>. Also, the Agent MUST publish an <i>MTConnectErrors Response Document</i> that identifies <code>UNSUPPORTED</code> as the <code>errorCode</code>.</p> |

| Continuation of Table 20 | | |
|--------------------------|---------------------------------|---|
| HTTP Status Code | Code Name | Description |
| 406 | Not Acceptable | <p>The <i>HTTP Accept Header</i> in the <i>Request</i> was not one of the supported representations.</p> <p>The <i>Agent</i> MUST return a 406 <i>HTTP Status Code</i>. Also, the <i>Agent</i> MUST publish an <i>MTConnectErrors Response Document</i> that identifies UNSUPPORTED as the errorCode.</p> |
| 431 | Request Header Fields Too Large | <p>The fields in the <i>HTTP Request</i> exceed the limit of the implementation of the <i>Agent</i>.</p> <p>The <i>Agent</i> MUST return a 431 <i>HTTP Status Code</i>. Also, the <i>Agent</i> MUST publish an <i>MTConnectErrors Response Document</i> that identifies INVALID_REQUEST as the errorCode.</p> |
| 500 | Internal Server Error | <p>There was an unexpected error in the <i>Agent</i> while responding to a <i>Request</i>.</p> <p>The <i>Agent</i> MUST return a 500 <i>HTTP Status Code</i>. Also, the <i>Agent</i> MUST publish an <i>MTConnectErrors Response Document</i> that identifies INTERNAL_ERROR as the errorCode.</p> |

2223 8.3.5 HTTP Errors

2224 When an *Agent* receives an *HTTP Request* that is incorrectly formatted or is not supported
 2225 by the *Agent*, the *Agent* **MUST** publish an *HTTP Error Message* which includes a specific
 2226 status code from the tables above indicating that the *Request* could not be handled by the
 2227 *Agent*.

2228 Also, if the *Agent* experiences an internal error and is unable to provide the requested
 2229 *Response Document*, it **MUST** publish an *HTTP Error Message* that includes a specific
 2230 status code from the table above.

2231 When an *Agent* encounters an error in interpreting or responding to an *HTTP Request*,
 2232 the *Agent* **MUST** also publish an *MTConnectErrors Response Document* that provides
 2233 additional details about the error. See *Section 9 - Error Information Model* for details on
 2234 the *MTConnectErrors Response Document*.

2235 8.3.6 Streaming Data

2236 *HTTP Data Streaming* is a method for a server to provide a continuous stream of information
 2237 in response to a single *Request* from a client software application. *Data Streaming* is
 2238 a version of a *Publish/Subscribe* method of communications.

2239 When an *HTTP Request* includes an `interval <query>` parameter, an *Agent* **MUST**
 2240 provide data with a minimum delay between the end of one data transmission and the
 2241 beginning of the next data transmission defined by the value (in milliseconds) provided
 2242 for `interval` parameter. A value of zero (0) for the `interval` parameter indicates
 2243 that the *Agent* should deliver data at the highest rate possible.

2244 The format of the response **MUST** use a MIME encoded message with each section separated
 2245 by a MIME boundary. Each section **MUST** contain an entire *MTConnectStreams Response Document*.

2247 If there are no available *Data Entities* to be published after the `interval` time has
 2248 elapsed, an *Agent* **MUST** wait until additional information is available to be published.
 2249 If no new information is available to be published within the time defined by the
 2250 `heartbeat` parameter, the *Agent* **MUST** then send a new section to ensure the receiver
 2251 that the *Agent* is functioning correctly. In this case, the content of the *MTConnect- Streams* document **MUST** be empty since no data is available.

2253 For more information on MIME see IETF RFC 1521 and RFC 822.

2254 An example of the format for a *HTTP Request* that includes an `interval` parameter is:

Example 8: Example for HTTP Request with interval parameter

2255 1 `http://localhost:5000/sample?interval=1000`

2256 **HTTP Response Header:**

Example 9: HTTP Response header

2257 1 `HTTP/1.1 200 OK`
 2258 2 `Connection: close`
 2259 3 `Date: Sat, 13 Mar 2010 08:33:37 UTC`
 2260 4 `Status: 200 OK`
 2261 5 `Content-Disposition: inline`

```

2262 6 X-Runtime: 144ms
2263 7 Content-Type: multipart/x-mixed-replace;boundary=
2264 8 a8e12eced4fb871ac096a99bf9728425
2265 9 Transfer-Encoding: chunked

```

2266 Lines 1-9 in *Example 9* represent a standard header for a MIME multipart/x-mixed-replace message. The boundary is a separator for each section of the stream. Lines 7-8 indicate this is a multipart MIME message and the boundary between sections.

2269 With streaming protocols, the Content-length **MUST** be omitted and Transfer-Encoding **MUST** be set to chunked (line 9). See IETF RFC 7230 for a full description 2270 of the HTTP protocol and chunked encoding.

Example 10: HTTP Response header 2

```

2272 10 --a8e12eced4fb871ac096a99bf9728425
2273 11 Content-type: text/xml
2274 12 Content-length: 887
2275 13
2276 14 <?xml version="1.0" encoding="UTF-8"?>
2277 15 <MTConnectStreams ...>...

```

2278 Each section of the document begins with a boundary preceded by two hyphens (-). The 2279 Content-type and Content-length MIME header fields **MUST** be provided for 2280 each section and **MUST** be followed by <CR><LF><CR><LF> (ASCII code for <CR> is 2281 13 and <LF> is 10) before the XML document. The header and the <CR><LF><CR><LF> 2282 **MUST NOT** be included in the computation of the content length.

2283 An *Agent* **MUST** continue to stream results until the client closes the connection. The 2284 *Agent* **MUST NOT** stop the streaming for any other reason other than the *Agent* process 2285 shutting down or the client application becoming unresponsive and not receiving data (as 2286 indicated by not consuming data and the write operation blocking).

2287 8.3.6.1 Heartbeat

2288 When *Streaming Data* is requested from a *Sample Request*, an *Agent* **MUST** support a 2289 *heartbeat* to indicate to a client application that the HTTP connection is still viable during 2290 times when there is no new data available to be published. The *heartbeat* is indicated by 2291 an *Agent* by sending an *MTConnect Response Document* with an empty Streams container 2292 (See *MTConnect Standard: Part 3.0 - Streams Information Model, Section 4.1 Streams* for 2293 more details on the Streams container) to the client software application.

2294 The *heartbeat* **MUST** occur on a periodic basis given by the optional heartbeat query 2295 parameter and **MUST** default to 10 seconds. An *Agent* **MUST** maintain a separate *heart-*

2296 *beat* for each client application for which the *Agent* is responding to a *Data Streaming*
 2297 *Request*.

2298 An *Agent* **MUST** begin calculating the interval for the time-period of the *heartbeat* for
 2299 each client application immediately after a *Response Document* is published to that spe-
 2300 specific client application.

2301 The *heartbeat* remains in effect for each client software application until the *Data Stream-
 2302 ing Request* is terminated by either the *Agent* or the client application.

2303 8.3.7 References

2304 A *Structural Element* **MAY** include a set of *References* of the following types that **MAY**
 2305 alter the content of the *MTConnectStreams Response Documents* published in response to
 2306 a *Current Request* or a *Sample Request* as specified:

- 2307 • A *Component Reference* (*ComponentRef*) modifies the set of resulting *Data Enti-
 2308 ties*, limited by a path query parameter of a *Current Request* or *Sample Request*,
 2309 to include the *Data Entities* associated with the *Structural Element* whose value for
 2310 its *id* attribute matches the value provided for the *idRef* attribute of the *Compo-
 2311 nentRef* element. Additionally, *Data Entities* defined for any *Lower Level Struc-
 2312 tural Element(s)* associated with the identified *Structural Element* **MUST** also be
 2313 returned. The result is equivalent to appending `//[@id=<"idRef">]` to the path
 2314 query parameters of the *Current Request* or *Sample Request*. See *Section 8.3.2 -
 2315 Current Request Implemented Using HTTP* for more details on path queries.
- 2316 • A *Data Item Reference* (*DataItemRef*) modifies the set of resulting *Data Enti-
 2317 ties*, limited by a path query parameter of a *Current Request* or *Sample Request*, to
 2318 include the *Data Entity* whose value for its *id* attribute matches the value provided
 2319 for the *idRef* attribute of the *DataItemRef* element. The result is equivalent
 2320 to appending `//[@id=<"idRef">]` to the path query parameters of the *Current
 2321 Request* or *Sample Request*. See *Section 8.3.2 - Current Request Implemented Using
 2322 HTTP* for more details on path queries.

2323 9 Error Information Model

2324 The *Error Information Model* establishes the rules and terminology that describes the *Re-
2325 sponse Document* returned by an *Agent* when it encounters an error while interpreting a
2326 *Request* for information from a client software application or when an *Agent* experiences
2327 an error while publishing the *Response* to a *Request* for information.

2328 An *Agent* provides the information regarding errors encountered when processing a *Re-
2329 quest* for information by publishing an *MTConnectErrors Response Document* to the client
2330 software application that made the *Request* for information.

2331 9.1 MTConnectError Response Document

2332 The *MTConnectErrors Response Document* is comprised of two sections: Header and
2333 Errors.

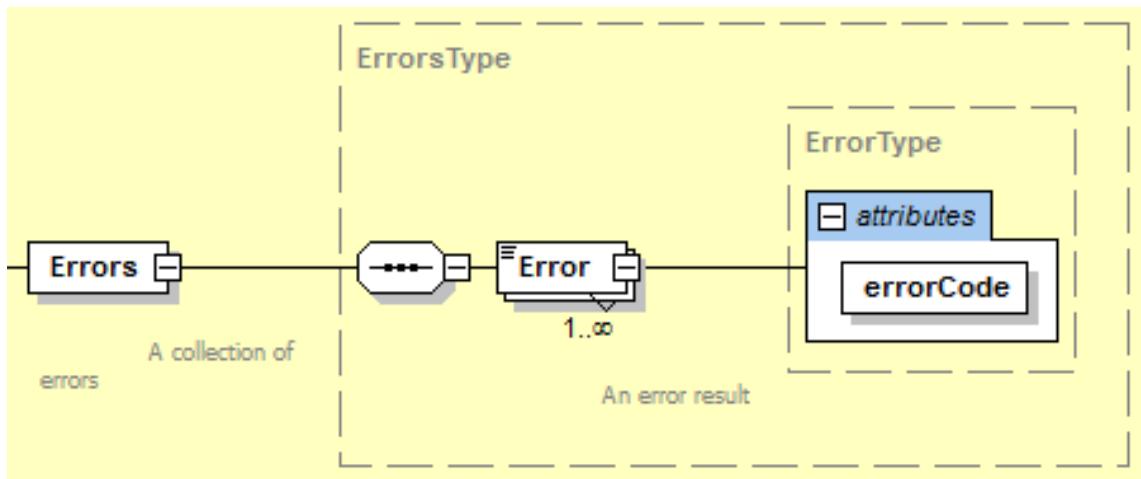
2334 The Header section contains information defining the creation of the document and the
2335 data storage capability of the *Agent* that generated the document. (See *Section 6.5.4 -
2336 Header for MTConnectError*)

2337 The Errors section of the *MTConnectErrors Response Document* is a *Structural Element*
2338 that organizes *Data Entities* describing each of the errors reported by an *Agent*.

2339 9.1.1 Structural Element for MTConnectError

2340 *Structural Elements* are XML elements that form the logical structure for an XML docu-
2341 ment. The *MTConnectErrors Response Document* has only one *Structural Element*. This
2342 *Structural Element* is Errors. Errors is an XML container element that organizes the
2343 information and data associated with all errors relevant to a specific *Request* for informa-
2344 tion.

2345 The following *XML Schema* represents the structure of the Errors XML element.

**Figure 21:** Errors Schema Diagram**Table 21:** MTConnect Errors Element

| Element | Description | Occurrence |
|---------|--|------------|
| Errors | <p>An XML container element in an <i>MTConnectErrors Response Document</i> provided by an <i>Agent</i> when an error is encountered associated with a <i>Request</i> for information from a client software application.</p> <p>There MUST be only one Errors element in an <i>MTConnectErrors Response Document</i>.</p> <p>The Errors element MUST contain at least one Error Data Entity element.</p> | 1 |

2346 Note: When compatibility with Version 1.0.1 and earlier of the MTConnect Standard
 2347 is required for an implementation, the *MTConnectErrors Response Document*
 2348 contains only a single Error Data Entity and the Errors Structural Element
 2349 **MUST NOT** appear in the document.

2350 **9.1.2 Error Data Entity**

2351 When an *Agent* encounters an error when responding to a *Request* for information from
 2352 a client software application, the information describing the error(s) is reported as a *Data*
 2353 *Entity* in an *MTConnectErrors Response Document*. *Data Entities* are organized in the
 2354 *Errors* XML container.

2355 There is only one type of *Data Entity* defined for an *MTConnectErrors Response Document*. That *Data Entity* is called *Error*.

2357 The following is an illustration of the structure of an XML document demonstrating how
 2358 *Error Data Entities* are reported in an *MTConnectErrors Response Document*:

Example 11: Example of Error in MTConnectError

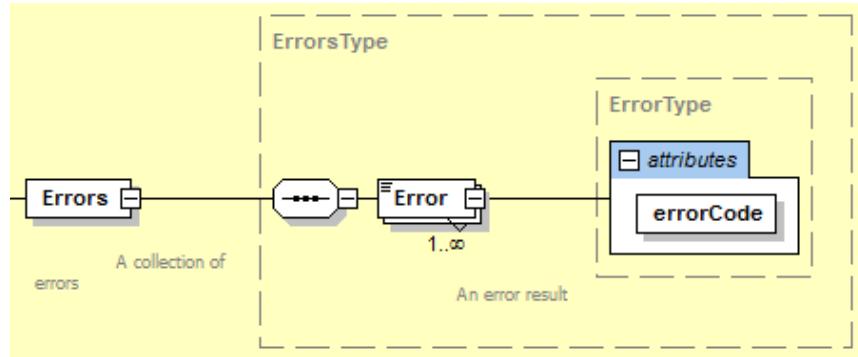
```
2359 1 <MTConnectError>
2360 2   <Header/>
2361 3   <Errors>
2362 4     <Error/>
2363 5     <Error/>
2364 6     <Error/>
2365 7   </Errors>
2366 8 </MTConnectError>
```

2367 The *Errors* element **MUST** contain at least one *Data Entity*. Each *Data Entity* describes
 2368 the details for a specific error reported by an *Agent* and is represented by the XML element
 2369 named *Error*.

2370 *Error* XML elements **MAY** contain both attributes and CDATA that provide details further defining a specific error. The CDATA **MAY** provide the complete text provided by an
 2371 *Agent* for the specific error.

2373 **9.1.2.1 XML Schema Structure for Error**

2374 The *XML Schema* in *Figure 22* represents the structure of an *Error* XML element showing
 2375 the attributes defined for *Error*.

**Figure 22:** Error Schema Diagram

2376 9.1.2.2 Attributes for Error

2377 Error has one attribute. *Table 22* defines this attribute that provides additional information for an Error XML element.

Table 22: Attributes for Error

| Attribute | Description | Occurrence |
|------------------------|---|------------|
| <code>errorCode</code> | Provides a descriptive code that indicates the type of error that was encountered by an <i>Agent</i> when attempting to respond to a <i>Request</i> for information. <code>errorCode</code> is a required attribute. | 1 |

2379 **9.1.2.3 Values for errorCode**

2380 There is a limited vocabulary defined for `errorCode`. The value returned for `error-`
 2381 `Code` **MUST** be one of the following:

Table 23: Values for errorCode

| Value for errorCode | Description |
|---------------------|---|
| ASSET_NOT_FOUND | The <i>Request</i> for information specifies an <i>MTConnect Asset</i> that is not recognized by the <i>Agent</i> . |
| INTERNAL_ERROR | The <i>Agent</i> experienced an error while attempting to published the requested information. |
| INVALID_REQUEST | The <i>Request</i> contains information that was not recognized by the <i>Agent</i> . |
| INVALID_URI | The URI provided was incorrect. |
| INVALID_XPATH | The XPath identified in the <i>Request</i> for information could not be parsed correctly by the <i>Agent</i> . This could be caused by an invalid syntax or the XPath did not match a valid identify for any information stored in the <i>Agent</i> . |
| NO_DEVICE | The identity of the piece of equipment specified in the <i>Request</i> for information is not associated with the <i>Agent</i> . |
| OUT_OF_RANGE | The <i>Request</i> for information specifies <i>Streaming Data</i> that includes sequence number(s) for pieces of data that are beyond the end of the <i>buffer</i> . |
| QUERY_ERROR | The <i>Agent</i> was unable to interpret the <i>Query</i> . The <i>Query</i> parameters do not contain valid values or include an invalid parameter. |
| TOO_MANY | The <code>count</code> parameter provided in the <i>Request</i> for information requires either of the following: <ul style="list-style-type: none"> - <i>Streaming Data</i> that includes more pieces of data than the <i>Agent</i> is capable of organizing in an <i>MTConnectStreams Response Document</i>. - Assets that include more <i>Asset Documents</i> in an <i>MTConnectAssets Response Document</i> than the <i>Agent</i> is capable of handling. |

| Continuation of Table 23 | |
|--------------------------|--|
| Value for errorCode | Description |
| UNAUTHORIZED | The <i>Requester</i> does not have sufficient permissions to access the requested information. |
| UNSUPPORTED | A valid <i>Request</i> was provided, but the <i>Agent</i> does not support the feature or type of <i>Request</i> . |

2382 9.1.2.4 CDATA for Error

2383 The CDATA for Error contains a textual description of the error and any additional
 2384 information an *Agent* is capable of providing regarding a specific error. The *Valid Data*
 2385 *Value* returned for Error **MAY** be any text string.

2386 9.1.3 Examples for MTConnectError

2387 *Example 12* is an example demonstrating the structure of an *MTConnectErrors Response*
 2388 *Document*:

Example 12: Example of structure for MTConnectError

```

2389 1  <?xml version="1.0" encoding="UTF-8"?>
2390 2    <MTConnectError
2391 3      xmlns="urn:mtconnect.org:MTConnectError:1.4"
2392 4      xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
2393 5      xsi:schemaLocation="urn:mtconnect.org:MTConnectError
2394 6        :1.4/schemas/MTConnectError_1.4.xsd">
2395 7      <Header creationTime="2010-03-12T12:33:01Z"
2396 8        sender="MyAgent" version="1.4.1.10"
2397 9        bufferSize="131000" instanceId="1383839" />
2398 10     <Errors>
2399 11       <Error errorCode="OUT_OF_RANGE" >Argument was
2400 12         out of range</Error>
2401 13       <Error errorCode="INVALID_XPATH" >Bad
2402 14         path</Error>
2403 15     </Errors>
2404 16   </MTConnectError>
```

2405 *Example 13* is an example demonstrating the structure of an *MTConnectErrors Response*
 2406 *Document* when backward compatibility with Version 1.0.1 and earlier of the MTConnect
 2407 Standard is required. In this case, the *Document Body* contains only a single *Error Data*
 2408 *Entity* and the *Errors Structural Element* **MUST NOT** appear in the document.

Example 13: Example of structure for MTConnectError when backward compatibility is required

```
2409 1  <?xml version="1.0" encoding="UTF-8"?>
2410 2  <MTConnectError
2411 3      xmlns="urn:mtconnect.org:MTConnectError:1.1"
2412 4      xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
2413 5      xsi:schemaLocation="urn:mtconnect.org:MTConnectError
2414 6          :1.1/schemas/MTConnectError_1.1.xsd">
2415 7      <Header creationTime="2010-03-12T12:33:01Z"
2416 8          sender="MyAgent" version="1.1.0.10"
2417 9          bufferSize="131000" instanceId="1383839" />
2418 10     <Error errorCode="OUT_OF_RANGE" >Argument was out
2419 11         of range</Error>
2420 12 </MTConnectError>
```

2421 Appendices

2422 A Bibliography

- 2423 Engineering Industries Association. *EIA Standard - EIA-274-D*, Interchangeable Variable,
2424 Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically
2425 Controlled Machines. Washington, D.C. 1979.
- 2426 ISO TC 184/SC4/WG3 N1089. *ISO/DIS 10303-238*: Industrial automation systems and
2427 integration Product data representation and exchange Part 238: Application Protocols: Ap-
2428 plication interpreted model for computerized numerical controllers. Geneva, Switzerland,
2429 2004.
- 2430 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
2431 tems and integration – Physical device control – Data model for computerized numerical
2432 controllers – Part 10: General process data. Geneva, Switzerland, 2004.
- 2433 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
2434 tems and integration – Physical device control – Data model for computerized numerical
2435 controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.
- 2436 International Organization for Standardization. *ISO 6983/1* – Numerical Control of ma-
2437 chines – Program format and definition of address words – Part 1: Data format for posi-
2438 tioning, line and contouring control systems. Geneva, Switzerland, 1982.
- 2439 Electronic Industries Association. *ANSI/EIA-494-B-1992*, 32 Bit Binary CL (BCL) and
2440 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines.
2441 Washington, D.C. 1992.
- 2442 National Aerospace Standard. *Uniform Cutting Tests* - NAS Series: Metal Cutting Equip-
2443 ment Specifications. Washington, D.C. 1969.
- 2444 International Organization for Standardization. *ISO 10303-11*: 1994, Industrial automa-
2445 tion systems and integration Product data representation and exchange Part 11: Descrip-
2446 tion methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.
- 2447 International Organization for Standardization. *ISO 10303-21*: 1996, Industrial automa-
2448 tion systems and integration – Product data representation and exchange – Part 21: Imple-
2449 mentation methods: Clear text encoding of the exchange structure. Geneva, Switzerland,
2450 1996.
- 2451 H.L. Horton, F.D. Jones, and E. Oberg. *Machinery's Handbook*. Industrial Press, Inc.

2452 New York, 1984.

2453 International Organization for Standardization. *ISO 841-2001: Industrial automation sys-*
2454 *tems and integration - Numerical control of machines - Coordinate systems and motion*
2455 *nomenclature.* Geneva, Switzerland, 2001.

2456 *ASME B5.59-2 Version 9c: Data Specification for Properties of Machine Tools for Milling*
2457 *and Turning.* 2005.

2458 *ASME/ANSI B5.54: Methods for Performance Evaluation of Computer Numerically Con-*
2459 *trolled Lathes and Turning Centers.* 2005.

2460 OPC Foundation. *OPC Unified Architecture Specification, Part 1: Concepts Version 1.00.*
2461 *July 28, 2006.*

2462 View the following site for RFC references: <http://www.faqs.org/rfcs/>.

2463 B Fundamentals of Using XML to Encode Response Documents

2464 The MTConnect Standard specifies the structures and constructs that are used to encode
2465 *Response Documents*. When these *Response Documents* are encoded using XML, there
2466 are additional rules defined by the XML standard that apply for creating an XML compli-
2467 ant document. An implementer should refer to the W3C website for additional information
2468 on XML documentation and implementation details - <http://www.w3.org/XML>.

2469 The following provides specific terms and guidelines referenced in the MTConnect Stan-
2470 dard for forming *Response Documents* with XML:

- 2471 • **tag:** A tag is an XML construct that forms the foundation for an XML expression.
2472 It defines the scope (beginning and end) of an XML expression. The main types of
2473 tags are:

- 2474 • **start-tag:** Designates the beginning on an XML element; e.g., `<Element Name>`
- 2475 • **end-tag:** Designates the end on an XML element; e.g., `</Element Name>`.

2476 Note: If an element has no *Child Elements* or CDATA, the end-tag may be
2477 shortened to `>`.

- 2478 • **Element:** An element is an XML statement that is the primary building block
2479 for a document encoded using XML. An element begins with a start-tag and
2480 ends with a matching end-tag. The characters between the start-tag and the
2481 end-tag are the element's content. The content may contain attributes, CDATA,
2482 and/or other elements. If the content contains additional elements, these elements
2483 are called *Child Elements*.

2484 An example would be: `<Element Name>Content of the Element</Element Name>`.

- 2485 • **Child Element:** An XML element that is contained within a higher-level *Parent El-*
2486 *ement*. A *Child Element* is also known as a sub-element. XML allows an unlimited
2487 hierarchy of *Parent Element-Child Element* relationships that establishes the struc-
2488 ture that defines how the various pieces of information in the document relate to
2489 each other. A *Parent Element* may have multiple associated *Child Elements*.

- 2490 • **Element Name:** A descriptive identifier contained in both the start-tag and
2491 end-tag that provides the name of an XML element.

- 2492 • **Attribute:** A construct consisting of a name-value pair that provides additional
2493 information about that XML element. The format for an attribute is `name="value"`;
2494 where the value for the attribute is enclosed in a set of quotation ("") marks. An XML
2495 attribute **MUST** only have a single value and each attribute can appear at most once
2496 in each element. Also, each attribute **MUST** be defined in a *schema* to either be
2497 required or optional.

- 2498 • An example of attributes for an XML element is *Example 14*:

Example 14: Example of attributes for an element

```
2499 1 <DataItem category="SAMPLE" id="S1load"
2500 2   nativeUnits="PERCENT" type="LOAD"
2501 3   units="PERCENT"/>
```

2502 In this example, DataItem is the ElementName. category, id, nativeU-
 2503 nits, type, and units are the names of the attributes. “SAMPLE”, “S1load”,
 2504 “PERCENT”, “LOAD”, and “PERCENT” are the values for each of the respective
 2505 attributes.

- 2506 • **CDATA:** CDATA is an XML term representing *Character Data*. *Character Data*
 2507 contains a value(s) or text that is associated with an XML element. CDATA can be
 2508 restricted to certain formats, patterns, or words.

2509 An example of CDATA associated with an XML element would be *Example 15*:

Example 15: Example of cdata associated with element

```
2510 1 <Message id="M1">This is some text</Message>
```

2511 In this example, Message is the ElementName and This is some text is
 2512 the CDATA.

- 2513 • **namespace:** An XML *namespace* defines a unique vocabulary for named elements
 2514 and attributes in an XML document. An XML document may contain content that is
 2515 associated with multiple *namespaces*. Each *namespace* has its own unique identifier.

2516 Elements and attributes are associated with a specific *namespace* by placing a pre-
 2517 fix on the name of the element or attribute that associates that name to a specific
 2518 *namespace*; e.g., x:MyTarget associates the element name MyTarget with the
 2519 *namespace* designated by x: (the prefix).

2520 *namespaces* are used to avoid naming conflicts within an XML document. The
 2521 naming convention used for elements and attributes may be associated with either
 2522 the default *namespace* specified in the *Header* of an XML document or they may
 2523 be associated with one or more alternate *namespaces*. All elements or attributes
 2524 associated with a *namespace* that is not the default *namespace*, must include a prefix
 2525 (e.g., x:) as part of the name of the element or attribute to associate it with the proper
 2526 *namespace*. See *Appendix C* for details on the structure for XML *Headers*.

2527 The names of the elements and attributes declared in a *namespace* may be identified
 2528 with a different prefix than the prefix that signifies that specific *namespace*. These
 2529 prefixes are called *namespace* aliases. As an example, MTConnect Standard spe-
 2530 cific *namespaces* are designated as m: and the names of the elements and attributes
 2531 defined in that *namespace* have an alias prefix of mt : which designates these names
 2532 as MTConnect Standard specific vocabulary; e.g., mt :MTConnectDevices.

2533 XML documents are encoded with a hierarchy of elements. In general, XML elements
 2534 may contain *Child Elements*, CDATA, or both. However, in the MTConnect Standard,
 2535 an element **MUST NOT** contain mixed content; meaning it cannot contain both *Child*
 2536 *Elements* and CDATA.

2537 The *semantic data model* defined for each *Response Document* specifies the elements and
 2538 *Child Elements* that may appear in a document. The *semantic data model* also defines the
 2539 number of times each element and *Child Element* may appear in the document.

2540 *Example 16* demonstrates the hierarchy of XML elements and *Child Elements* used to
 2541 form an XML document:

Example 16: Example of hierarchy of XML elements

```
2542 1 <Root Level>      (Parent Element)
2543 2   <First Level>    (Child Element to Root Level and
2544 3     Parent Element to Second Level)
2545 4     <Second Level>  (Child Element to First Level
2546 5       and Parent Element to Third Level)
2547 6       <Third Level name="N1"></Third Level>
2548 7         (Child Element to Second Level)
2549 8         <Third Level name="N2"></Third Level>
2550 9           (Child Element to Second Level)
2551 10          <Third Level name="N3"></Third Level>
2552 11            (Child Element to Second Level)
2553 12          </Second Level>   (end-tag for Second Level)
2554 13        </First Level>    (end-tag for First Level)
2555 14      </Root Level>    (end-tag for Root Level)
```

2556 In the *Example 16*, *Root Level* and *First Level* have one *Child Element* (sub-elements)
 2557 each and *Second Level* has three *Child Elements*; each called *Third Level*. Each *Third*
 2558 *Level* element has a different name attribute. Each level in the structure is an element and
 2559 each lower level element is a *Child Element*.

2560 C Schema and Namespace Declaration Information

2561 There are four pseudo-attributes typically included in the *Header* of a *Response Document*
 2562 that declare the *schema* and *namespace* for the document. Each of these pseudo-attributes
 2563 provides specific information for a client software application to properly interpret the
 2564 content of the *Response Document*.

2565 The pseudo-attributes include:

- 2566 • `xmlns:xsi` – The `xsi` portion of this attribute name stands for *XML Schema*
 2567 instance. An *XML Schema* instance provides information that may be used by a
 2568 software application to interpret XML specific information within a document. See
 2569 the W3C website for more details on `xmlns:xsi`.

- 2570 • `xmlns` – Declares the default *namespace* associated with the content of the *Re-*
 2571 *sponse Document*. The default *namespace* is considered to apply to all elements and
 2572 attributes whenever the name of the element or attribute does not contain a prefix
 2573 identifying an alternate *namespace*.

2574 The value of this attribute is an URN identifying the name of the file that defines
 2575 the details of the *namespace* content. This URN provides a unique identify for the
 2576 *namespace*.

- 2577 • `xmlns:m` – Declares the MTConnect specific *namespace* associated with the con-
 2578 tent of the *Response Document*. There may be multiple *namespaces* declared for
 2579 an XML document. Each may be associated to the default *namespace* or it may be
 2580 totally independent. The `:m` designates that this is a specific MTConnect *namespace*
 2581 which is directly associated with the default *namespace*.

2582 Note: See *Section 6.7 - Extensibility* for details regarding extended *namespaces*.

2583 The value associated with this attribute is an URN identifying the name of the file
 2584 that defines the details of the *namespace* content.

- 2585 • `xsi:schemaLocation` - Declares the name for the *schema* associated with the
 2586 *Response Document* and the location of the file that contains the details of the
 2587 *schema* for that document.

2588 The value associated with this attribute has two parts:

2589 - A URN identifying the name of the specific *XML Schema* instance associated
 2590 with the *Response Document*.

2591 - The path to the location where the file describing the specific *XML Schema*
 2592 instance is located. If the file is located in the same root directory where the *Agent*
 2593 is installed, then the local path MAY be declared. Otherwise, a fully qualified URL
 2594 must be declared to identify the location of the file.

2595 Note: In the format of the value associated with `xsi:schemaLocation`, the
 2596 URN and the path to the *schema* file **MUST** be separated by a “space”.

2597 In *Example 17*, the first line is the *XML Declaration*. The second line is a *Root Element*
 2598 called `MTConnectDevices`. The remaining four lines are the pseudo-attributes of
 2599 `MTConnectDevices` that declare the XML *schema* and *namespace* associated with an
 2600 *MTConnectDevices Response Document*.

Example 17: Example of schema and namespace declaration

```
2601 1  <?xml version="1.0" encoding="UTF-8"?>
2602 2    <MTConnectDevices
2603 3      xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
2604 4      xmlns="urn:mtconnect.org:MTConnectDevices:1.3"
2605 5      xmlns:m="urn:mtconnect.org:MTConnectDevices:1.3"
2606 6      xsi:schemaLocation="urn:mtconnect.org:
2607 7          MTConnectDevices:1.3 /schemas/MTConnectDevices\_1.3.xsd">
```

2608 The format for the values provided for each of the pseudo-attributes **MUST** reference
 2609 the *semantic data model* (e.g., `MTConnectDevices`, `MTConnectStreams`, `MTCon-`
 2610 `nnectAssets`, or `MTConnectError`) and the version (i.e.; 1.1, 1.2, 1.3, etc.) of
 2611 the `MTConnect` Standard that depict the *schema* and *namespace(s)* associated with a spe-
 2612 cific *Response Document*.

2613 When an implementer chooses to extend an `MTConnect Data Model` by adding custom
 2614 data types or additional *Structural Elements*, the *schema* and *namespace* for that *Data*
 2615 *Model* should be updated to reflect the additional content. When this is done, the *names-
 2616 pace* and *schema* information in the *Header* should be updated to reflect the URI for the
 2617 extended *namespace* and *schema*.



MTConnect® Standard

Part 2.0 – Devices Information Model

Version 1.5.0

Prepared for: MTConnect Institute
Prepared on: December 2, 2019

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/>.

MTConnect Specification and Materials

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 | 10 |
| 2.3 MTConnect References | 10 |
| 3 Devices Information Model | 11 |
| 4 Structural Elements for MTConnectDevices | 13 |
| 4.1 Devices | 16 |
| 4.2 Device | 17 |
| 4.2.1 XML Schema Structure for Device | 18 |
| 4.2.2 Attribute for Device | 18 |
| 4.2.3 Elements for Device | 20 |
| 4.2.3.1 Description for Device | 21 |
| 4.2.3.2 Configuration for Device | 23 |
| 4.2.3.3 DataItems for Device | 24 |
| 4.2.3.4 Components within Device | 25 |
| 4.2.3.5 Compositions for Device | 25 |
| 4.2.3.6 References for Device | 25 |
| 4.3 Components | 25 |
| 4.4 Component | 26 |
| 4.4.1 XML Schema Structure for Component | 26 |
| 4.4.2 Attribute for Component | 28 |
| 4.4.3 Elements of Component | 31 |
| 4.4.3.1 Description for Component | 31 |
| 4.4.3.2 Configuration for Component | 33 |
| 4.4.3.3 DataItems for Component | 34 |
| 4.4.3.4 Components within Component | 35 |
| 4.4.3.5 Compositions for Component | 35 |
| 4.4.3.6 References for Component | 35 |
| 4.5 Compositions | 35 |
| 4.6 Composition | 36 |
| 4.6.1 XML Schema Structure for Composition | 37 |
| 4.6.2 Attributes for Composition | 38 |
| 4.6.3 Elements of Composition | 39 |
| 4.6.3.1 Description for Composition | 40 |
| 4.7 References | 41 |
| 4.8 Reference | 42 |
| 4.8.1 ComponentRef | 43 |

| | |
|--|-----------|
| 4.8.2 DataItemRef | 44 |
| 4.9 Relationships | 46 |
| 4.10 Relationship | 46 |
| 4.10.1 DeviceRelationship | 48 |
| 4.10.2 ComponentRelationship | 53 |
| 5 Component Structural Elements | 56 |
| 5.1 Axes | 58 |
| 5.1.1 Linear | 58 |
| 5.1.2 Rotary | 59 |
| 5.1.2.1 Chuck | 59 |
| 5.2 Controller | 59 |
| 5.2.1 Path | 60 |
| 5.3 Systems | 60 |
| 5.3.1 Hydraulic System | 61 |
| 5.3.2 Pneumatic System | 61 |
| 5.3.3 Coolant System | 61 |
| 5.3.4 Lubrication System | 61 |
| 5.3.5 Electric System | 61 |
| 5.3.6 Enclosure System | 62 |
| 5.3.7 Protective System | 62 |
| 5.3.8 ProcessPower System | 62 |
| 5.3.9 Feeder System | 62 |
| 5.3.10 Dielectric System | 62 |
| 5.3.11 EndEffector System | 63 |
| 5.4 Auxiliaries | 63 |
| 5.4.1 Loader System | 63 |
| 5.4.2 WasteDisposal System | 63 |
| 5.4.3 ToolingDelivery System | 63 |
| 5.4.4 BarFeeder System | 64 |
| 5.4.5 Environmental System | 64 |
| 5.4.6 Sensor System | 64 |
| 5.4.7 Deposition System | 64 |
| 5.5 Resources | 64 |
| 5.5.1 Materials | 65 |
| 5.5.1.1 Stock | 65 |
| 5.6 Interfaces | 65 |
| 5.7 Other Components | 65 |
| 5.7.1 Actuator | 66 |
| 5.7.2 Door | 66 |
| 5.7.3 Sensor | 67 |

| | |
|---|------------|
| 6 Composition Type Structural Elements | 68 |
| 7 Data Entities for Device | 72 |
| 7.1 DataItems | 73 |
| 7.2 DataItem | 73 |
| 7.2.1 XML Schema Structure for DataItem | 73 |
| 7.2.2 Attributes for DataItem | 75 |
| 7.2.2.1 name Attribute for DataItem | 79 |
| 7.2.2.2 id Attribute for DataItem | 79 |
| 7.2.2.3 type and subType Attributes for DataItem | 80 |
| 7.2.2.4 statistic Attribute for DataItem | 80 |
| 7.2.2.5 units Attribute for DataItem | 82 |
| 7.2.2.6 nativeUnits Attribute for DataItem | 83 |
| 7.2.2.7 nativeScale Attribute for DataItem | 85 |
| 7.2.2.8 category Attribute for DataItem | 85 |
| 7.2.2.9 coordinateSystem Attribute for DataItem | 87 |
| 7.2.2.10 compositionId Attribute for DataItem | 87 |
| 7.2.2.11 sampleRate Attribute for DataItem | 88 |
| 7.2.2.12 representation Attribute for DataItem | 88 |
| 7.2.2.13 significantDigits Attribute for DataItem | 90 |
| 7.2.2.14 discrete Attribute for DataItem | 90 |
| 7.2.3 Elements for DataItem | 90 |
| 7.2.3.1 Source Element for DataItem | 91 |
| 7.2.3.1.1 Attributes for Source | 92 |
| 7.2.3.2 Constraints Element for DataItem | 93 |
| 7.2.3.2.1 Schema for Constraints | 93 |
| 7.2.3.3 Filters Element for DataItem | 96 |
| 7.2.3.3.1 Filter | 97 |
| 7.2.3.4 InitialValue Element for DataItem | 98 |
| 7.2.3.5 ResetTrigger Element for DataItem | 98 |
| 8 Listing of Data Items | 101 |
| 8.1 Data Items in category SAMPLE | 102 |
| 8.2 Data Items in category EVENT | 121 |
| 8.3 Data Items in category CONDITION | 146 |
| 9 Sensor | 148 |
| 9.1 Sensor Data | 148 |
| 9.2 Sensor Unit | 149 |
| 9.3 Sensor Configuration | 151 |
| 9.3.1 Elements for SensorConfiguration | 153 |
| 9.3.1.1 Attributes for Channel | 154 |
| 9.3.1.2 Elements for Channel | 155 |

| | |
|-----------------------------|------------|
| Appendices | 158 |
| A Bibliography | 158 |

Table of Figures

| | |
|--|------------|
| Figure 1: Example Device Structural Elements | 15 |
| Figure 2: Example Composition Structural Elements | 16 |
| Figure 3: Device Diagram | 18 |
| Figure 4: Description Diagram | 22 |
| Figure 5: Configuration Diagram | 24 |
| Figure 6: Component Diagram | 27 |
| Figure 7: Description of Component Diagram | 32 |
| Figure 8: Component Configuration Diagram | 34 |
| Figure 9: Composition Diagram | 38 |
| Figure 10:Description of Composition Diagram | 40 |
| Figure 11:Reference Diagram | 43 |
| Figure 12:ComponentRef Diagram | 43 |
| Figure 13:DataItemRef Diagram | 45 |
| Figure 14:Relationship Diagram | 47 |
| Figure 15:DeviceRelationship Diagram | 49 |
| Figure 16:ComponentRelationship Diagram | 53 |
| Figure 17:Axes Example with Two Linear Axes and One Rotary Axis | 58 |
| Figure 18:Example Data Entities for Device (DataItem) | 72 |
| Figure 19:DataItem Diagram | 74 |
| Figure 20:Source Diagram | 92 |
| Figure 21:Constraints Diagram | 94 |
| Figure 22:Filter Diagram | 97 |
| Figure 23:Sensor Data Associations | 149 |
| Figure 24:SensorConfiguration Diagram | 152 |

List of Tables

| | |
|--|----|
| Table 1: MTConnect Devices Element | 16 |
| Table 2: MTConnect Device Element | 17 |
| Table 3: Attributes for Device | 19 |
| Table 4: Elements for Device | 21 |
| Table 5: Attributes for Description | 22 |
| Table 6: MTConnect Configuration Element | 23 |
| Table 7: MTConnect Components Element | 25 |
| Table 8: MTConnect Component Element | 26 |
| Table 9: Attributes for Component | 28 |
| Table 10: Elements for Component | 31 |
| Table 11: Attributes for Description for Component | 32 |
| Table 12: MTConnect Configuration Element for Component | 33 |
| Table 13: MTConnect Compositions Element | 36 |
| Table 14: MTConnect Composition Element | 37 |
| Table 15: Attributes for Composition | 38 |
| Table 16: Elements for Composition | 39 |
| Table 17: Attributes for Description for Composition | 40 |
| Table 18: MTConnect References Element | 42 |
| Table 19: Attributes for ComponentRef | 44 |
| Table 20: Attributes for DataItemRef | 45 |
| Table 21: MTConnect Relationships Element | 46 |
| Table 22: Attributes for DeviceRelationship | 50 |
| Table 23: Attributes for ComponentRelationship | 54 |
| Table 24: Top Level Component Elements | 56 |
| Table 25: Composition type Elements | 68 |
| Table 26: MTConnect DataItems Element | 73 |
| Table 27: MTConnect DataItem Element | 73 |
| Table 28: Attributes for DataItem | 75 |
| Table 29: DataItem attribute statistic type | 81 |
| Table 30: DataItem attribute units type | 82 |
| Table 31: DataItem attribute nativeunits type | 84 |
| Table 32: DataItem attribute coordinateSystem type | 87 |
| Table 33: DataItem attribute representation type | 88 |
| Table 34: Elements for DataItem | 90 |
| Table 35: Attributes for Source | 92 |
| Table 36: Elements for Constraints | 95 |
| Table 37: MTConnect Filters Element | 96 |
| Table 38: DataItem Element Filter type | 97 |
| Table 39: MTConnect ResetTrigger Element | 99 |
| Table 40: DataItem Element ResetTrigger type | 99 |

| | |
|--|-----|
| Table 41: DataItem type subType for category SAMPLE | 102 |
| Table 42: DataItem type subType for category EVENT | 122 |
| Table 43: DataItem type for category CONDITION | 146 |
| Table 44: MTConnect SensorConfiguration Element | 153 |
| Table 45: Elements for SensorConfiguration | 153 |
| Table 46: Attributes for Channel | 155 |
| Table 47: Elements for Channel | 155 |

1 Purpose of This Document

2 This document, *MTConnect Standard: Part 2.0 - Devices Information Model* of the *MT-*
3 *Connect Standard*, establishes the rules and terminology to be used by designers to de-
4 scribe the function and operation of a piece of equipment and to define the data that is
5 provided by an *Agent* from the equipment. The *Devices Information Model* also defines
6 the structure for the XML document that is returned from an *Agent* in response to a *Probe*
7 *Request*.

8 In the MTConnect Standard, equipment represents any tangible property that is used in the
9 operations of a manufacturing facility. Examples of equipment are machine tools, ovens,
10 sensor units, workstations, software applications, and bar feeders.

11 Note: See *MTConnect Standard: Part 3.0 - Streams Information Model* of the MT-
12 *Connect Standard* for details on the XML documents that are returned from an
13 *Agent* in response to a *Sample Request* or *Current Request*.

14 2 Terminology and Conventions

15 Refer to *Section 3 of MTConnect Standard Part 1.0 - Overview and Fundamentals* for a
 16 dictionary of terms, reserved language, and document conventions used in the MTConnect
 17 Standard.

18 2.1 Glossary

19 CDATA

20 General meaning:

21 An abbreviation for Character Data.

22 CDATA is used to describe a value (text or data) published as part of an XML ele-
 23 ment.

24 For example, "This is some text" is the CDATA in the XML element:

25 <Message ...>This is some text</Message>

26 Appears in the documents in the following form: CDATA

27 HTTP

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

30 Note: HTTP is an IETF standard and is defined in RFC 7230.

31 See <https://tools.ietf.org/html/rfc7230> for more information.

32 NMTOKEN

33 The data type for XML identifiers.

34 Note: The identifier must start with a letter, an underscore "_" or a colon. The next
 35 character must be a letter, a number, or one of the following ".", "-", "_", ":". The
 36 identifier must not have any spaces or special characters.

37 Appears in the documents in the following form: NMTOKEN.

38 XML

39 Stands for eXtensible Markup Language.

40 XML defines a set of rules for encoding documents that both a human-readable and
 41 machine-readable.

42 XML is the language used for all code examples in the MTConnect Standard.

43 Refer to <http://www.w3.org/XML> for more information about XML.

44 **Agent**

45 Refers to an MTConnect Agent.

46 Software that collects data published from one or more piece(s) of equipment, orga-
47 nizes that data in a structured manner, and responds to requests for data from client
48 software systems by providing a structured response in the form of a *Response Doc-
49 ument* that is constructed using the *semantic data models* defined in the Standard.

50 Appears in the documents in the following form: *Agent*.

51 **Asset**

52 General meaning:

53 Typically referred to as an *MTConnect Asset*.

54 An *MTConnect Asset* is something that is used in the manufacturing process, but is
55 not permanently associated with a single piece of equipment, can be removed from
56 the piece of equipment without compromising its function, and can be associated
57 with other pieces of equipment during its lifecycle.

58 Used to identify a storage area in an Agent:

59 See description of *buffer*.

60 Used as an Information Model:

61 Used to describe an *Information Model* that contains the rules and terminology that
62 describe information that may be included in electronic documents representing *MT-
63 Connect Assets*.

64 The *Asset Information Models* defines the structure for the *Assets Response Docu-
65 ment*.

66 Individual *Information Models* describe the structure of the *Asset Documents* rep-
67 resent each type of *MTConnect Asset*. Appears in the documents in the following
68 form: *Asset Information Models* or (asset type) *Information Model*.

69 Used when referring to an MTConnect Asset:

70 Refers to the information related to an *MTConnect Asset* or a group of *MTConnect
71 Assets*.

72 Appears in the documents in the following form: *Asset* or *Assets*.

73 Used as an XML container or element:

- 74 • When used as an XML container that consists of one or more types of Asset
75 XML elements.

76 Appears in the documents in the following form: *Assets*.

- 77 ● When used as an abstract XML element. It is replaced in the XML document
 78 by types of *Asset* elements representing individual *Asset* entities.
 79 Appears in the documents in the following form: *Asset*.

80 **Used to describe information stored in an *Agent*:**

81 Identifies an electronic document published by a data source and stored in the *assets*
 82 *buffer* of an *Agent*.

83 Appears in the documents in the following form: *Asset Document*.

84 **Used as an XML representation of an *MTConnect Response Document*:**

85 Identifies an electronic document encoded in XML and published by an *Agent* in
 86 response to a *Request* for information from a client software application relating to
 87 *MTConnect Assets*.

88 Appears in the documents in the following form: *MTConnectAssets*.

89 **Used as an *MTConnect Request*:**

90 Represents a specific type of communications request between a client software ap-
 91 plication and an *Agent* regarding *MTConnect Assets*.

92 Appears in the documents in the following form: *Asset Request*.

93 **Used as part of an *HTTP Request*:**

94 Used in the path portion of an *HTTP Request Line*, by a client software applica-
 95 tion, to initiate an *Asset Request* to an *Agent* to publish an *MTConnectAssets*
 96 document.

97 Appears in the documents in the following form: *asset*.

98 ***Asset Document***

99 An electronic document published by an *Agent* in response to a *Request* for infor-
 100 mation from a client software application relating to Assets.

101 ***buffer***

102 **General meaning:**

103 A section of an *Agent* that provides storage for information published from pieces
 104 of equipment.

105 **Used relative to *Streaming Data*:**

106 A section of an *Agent* that provides storage for information relating to individual
 107 pieces of *Streaming Data*.

108 Appears in the documents in the following form: *buffer*.

109 **Used relative to *MTConnect Assets*:**

110 A section of an *Agent* that provides storage for *Asset Documents*.
111 Appears in the documents in the following form: *assets buffer*.

112 ***Child Element***

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

116 ***Current Request***

117 An HTTP request to the *Agent* for returning latest known values for the `DataItem`
118 as an `MTConnectStreams` XML document

119 ***Data Entity***

120 A primary data modeling element that represents all elements that either describe
121 data items that may be reported by an *Agent* or the data items that contain the actual
122 data published by an *Agent*.
123 Appears in the documents in the following form: *Data Entity*.

124 ***Data Set***

125 A set of *key-value pairs* where each entry is uniquely identified by the *key*.

126 ***Devices Information Model***

127 A set of rules and terms that describes the physical and logical configuration for a
128 piece of equipment and the data that may be reported by that equipment.
129 Appears in the documents in the following form: *Devices Information Model*.

130 ***Document***

131 General meaning:

132 A piece of written, printed, or electronic matter that provides information.

133 Used to represent an *MTConnect Document*:

134 Refers to printed or electronic document(s) that represent a *Part(s)* of the MTCon-
135 nect Standard.

136 Appears in the documents in the following form: *MTConnect Document*.

137 Used to represent a specific representation of an *MTConnect Document*:

138 Refers to electronic document(s) associated with an *Agent* that are encoded using
139 `XML`; *Response Documents* or *Asset Documents*.

140 Appears in the documents in the following form: *MTConnect XML Document*.

141 Used to describe types of information stored in an *Agent*:

142 In an implementation, the electronic documents that are published from a data source
143 and stored by an *Agent*.

144 Appears in the documents in the following form: *Asset Document*.

145 Used to describe information published by an *Agent*:

146 A document published by an *Agent* based upon one of the *semantic data models*
147 defined in the MTConnect Standard in response to a request from a client.

148 Appears in the documents in the following form: *Response Document*.

149 ***Equipment Metadata***

150 See *Metadata*

151 ***HTTP Request***

152 In the MTConnect Standard, a communications command issued by a client soft-
153 ware application to an *Agent* requesting information defined in the *HTTP Request*
154 *Line*.

155 Appears in the documents in the following form: *HTTP Request*.

156 ***HTTP Request Line***

157 In the MTConnect Standard, the first line of an *HTTP Request* describing a specific
158 *Response Document* to be published by an *Agent*.

159 Appears in the documents in the following form: *HTTP Request Line*.

160 ***Information Model***

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

163 For example, an information model is used to define the structure for each *MTCon-*
164 *nnect Response Document*; the definition of each piece of information within those
165 documents and the relationship between pieces of information.

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

167 ***Interaction Model***

168 The definition of information exchanged to support the interactions between pieces
169 of equipment collaborating to complete a task.

170 Appears in the documents in the following form: *Interaction Model*.

171 **Interface**

172 General meaning:

173 The exchange of information between pieces of equipment and/or software systems.

174 Appears in the documents in the following form: *interface*.

175 Used as an *Interaction Model*:

176 An *Interaction Model* that describes a method for inter-operations between pieces
177 of equipment.

178 Appears in the documents in the following form: *Interface*.

179 Used as an XML container or element:

180 - When used as an XML container that consists of one or more types of *Interface*
181 XML elements.

182 Appears in the documents in the following form: *Interfaces*.

183 - When used as an abstract XML element. It is replaced in the XML document
184 by types of *Interface* elements.

185 Appears in the documents in the following form: *Interface*

186 **key**

187 A unique identifier in a *key-value pair* association.

188 **key-value pair**

189 An association between an identifier referred to as the *key* and a value which taken
190 together create a *key-value pair*. When used in a set of *key-value pairs* each *key* is
191 unique and will only have one value associated with it at any point in time.

192 **Lower Level**

193 A nested element that is below a higher level element.

194 **Metadata**

195 Data that provides information about other data.

196 For example, *Equipment Metadata* defines both the *Structural Elements* that rep-
197 resent the physical and logical parts and sub-parts of each piece of equipment, the
198 relationships between those parts and sub-parts, and the definitions of the *Data En-*
199 *tities* associated with that piece of equipment.

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

201 **MTConnect Document**

202 See *Document*.

203 ***MTConnect Request***

204 A communication request for information issued from a client software application
205 to an *Agent*.

206 Appears in the documents in the following form: *MTConnect Request*.

207 ***MTConnect XML Document***

208 See *Document*.

209 ***Parent Element***

210 An XML element used to organize *Lower Level* child elements that share a common
211 relationship to the *Parent Element*.

212 Appears in the documents in the following form: *Parent Element*.

213 ***Request***

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

216 Appears in the documents in the following form: *Request*.

217 ***Response Document***

218 See *Document*.

219 ***Sample Request***

220 A request from the *Agent* for a stream of time series data.

221 ***semantic data model***

222 A methodology for defining the structure and meaning for data in a specific logical
223 way.

224 It provides the rules for encoding electronic information such that it can be inter-
225 preted by a software system.

226 Appears in the documents in the following form: *semantic data model*.

227 ***Streaming Data***

228 The values published by a piece of equipment for the *Data Entities* defined by the
229 *Equipment Metadata*.

230 Appears in the documents in the following form: *Streaming Data*.

231 ***Streams Information Model***

232 The rules and terminology (*semantic data model*) that describes the *Streaming Data*
233 returned by an *Agent* from a piece of equipment in response to a *Sample Request* or
234 a *Current Request*.

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

236 ***Structural Element***

237 General meaning:

238 An XML element that organizes information that represents the physical and logical
239 parts and sub-parts of a piece of equipment.

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

241 Used to indicate hierarchy of Components:

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

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

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

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

248 ***Top Level***

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

251 ***Valid Data Value***

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

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

255 **2.2 Acronyms**

256 ***AMT***

257 The Association for Manufacturing Technology

258 2.3 MTConnect References

- 259 [MTConnect Part 1.0] *MTConnect Standard Part 1.0 - Overview and Fundamentals*. Version 1.5.0.
- 260
- 261 [MTConnect Part 2.0] *MTConnect Standard: Part 2.0 - Devices Information Model*. Version 1.5.0.
- 262
- 263 [MTConnect Part 3.0] *MTConnect Standard: Part 3.0 - Streams Information Model*. Version 1.5.0.
- 264
- 265 [MTConnect Part 4.0] *MTConnect Standard: Part 4.0 - Assets Information Model*. Version 1.5.0.
- 266
- 267 [MTConnect Part 5.0] *MTConnect Standard: Part 5.0 - Interfaces*. Version 1.5.0.

268 3 Devices Information Model

269 The *Devices Information Model* provides a representation of the physical and logical con-
270 figuration for a piece of equipment used for a manufacturing process or for any other
271 purpose. It also provides the definition of data that may be reported by that equipment.

272 Using information defined in the *Devices Information Model*, a software application can
273 determine the configuration and reporting capabilities of a piece of equipment. To do this,
274 the software application issues a *Probe Request* (defined in *MTConnect Standard Part 1.0*
275 - *Overview and Fundamentals Section 8.1.1*) to an *Agent* associated with a piece of equip-
276 ment. An *Agent* responds to the *Probe Request* with an *MTConnectDevices* XML
277 document that contains information describing both the physical and logical structure of
278 the piece of equipment and a detailed description of each *Data Entity* that can be reported
279 by the *Agent* associated with the piece of equipment. This information allows the client
280 software application to interpret the document and to extract the data with the same mean-
281 ing, value, and context that it had at its original source.

282 The *MTConnectDevices* XML document is comprised of two sections: Header and
283 Devices.

284 The Header section contains protocol related information as defined in *MTConnect Stan-*
285 *dard Part 1.0 - Overview and Fundamentals Section 6.5.1*.

286 The Devices section of the *MTConnectDevices* document contains a *Device* XML
287 container for each piece of equipment described in the document. Each *Device* container
288 is comprised of two primary types of XML elements - *Structural Elements* and *Data Enti-*
289 *ties*.

290 *Structural Elements* are defined as XML elements that organize information that repre-
291 sents the physical and logical parts and sub-parts of a piece of equipment (See *Section 4 -*
292 *Structural Elements for MTConnectDevices* for more details).

293 *Data Entities* are defined as XML elements that describe data that can be reported by
294 a piece of equipment. In the *Devices Information Model*, *Data Entities* are defined as
295 *DataItem* elements (See *Section 7 - Data Entities for Device* and *Section 8 - Listing of*
296 *Data Items*).

297 The *Structural Elements* and *Data Entities* in the *MTConnectDevices* document pro-
298 vide information representing the physical and logical structure for a piece of equipment
299 and the types of data that the piece of equipment can report relative to that structure. The
300 *MTConnectDevices* document does not contain values for the data types reported by
301 the piece of equipment. The *MTConnectStreams* document defined in *MTConnect*

302 *Standard: Part 3.0 - Streams Information Model* provides the data values that are reported
303 by the piece of equipment. As such, most *Structural Elements* and *Data Entities* in the
304 MTConnectDevices document do not contain CDATA. XML elements that provide
305 values or information in the CDATA will be specifically identified in *Section 4 - Structural*
306 *Elements for MTConnectDevices*, *Section 7 - Data Entities for Device*, and *Section 9 -*
307 *Sensor*.

308 Note: The *MTConnect Standard* also defines the information model for *Assets*. An
309 *Asset* is something that is used in the manufacturing process, but is not perma-
310 nently associated with a single piece of equipment, can be removed from the
311 piece of equipment without compromising its function, and can be associated
312 with other pieces of equipment during its lifecycle. See *MTConnect Standard:*
313 *Part 4.0 - Assets Information Model* for more details on *Assets*.

314 4 Structural Elements for MTConnectDevices

315 *Structural Elements* are XML elements that form the logical structure for the MTCon-
316 nectDevices XML document. These elements are used to organize information that
317 represents the physical and logical architecture of a piece of equipment. Refer to *Figure 1*
318 for an overview of the *Structural Elements* used in an MTConnectDevices document.

319 A variety of *Structural Elements* are defined to describe a piece of equipment. Some
320 of these elements **MUST** always appear in the MTConnectDevices XML document,
321 while others are optional and **MAY** be used, as required, to provide additional structure.

322 The first, or highest level, *Structural Element* in a MTConnectDevices XML document
323 is Devices. Devices is a container type XML element used to group one or more
324 pieces of equipment into a single XML document. Devices **MUST** always appear in the
325 MTConnectDevices document.

326 Device is the next *Structural Element* in the MTConnectDevices XML document.
327 Device is also a container type XML element. A separate Device container is used
328 to identify each piece of equipment represented in the MTConnectDevices document.
329 Each Device container provides information on the physical and logical structure of
330 the piece of equipment and the data associated with that equipment. Device can also
331 represent any logical grouping of pieces of equipment that function as a unit or any other
332 data source that provides data through an Agent.

333 One or more Device element(s) **MUST** always appear in an MTConnectDevices
334 document.

335 Components is the next *Structural Element* in the MTConnectDevices XML doc-
336 ument. Components is also a container type XML element. Components is used to
337 group information describing *Lower Level* physical parts or logical functions of a piece of
338 equipment.

339 If the Components container appears in the XML document, it **MUST** contain one or
340 more Component type XML elements.

341 Component is the next level of *Structural Element* in the MTConnectDevices XML
342 document. Component is both an abstract type XML element and a container type ele-
343 ment.

344 As an abstract type element, Component will never appear in the XML document de-
345 scribing a piece of equipment and will be replaced by a specific Component type defined
346 in *Section 5 - Component Structural Elements*. Each Component type is also a container
347 type element. As a container, the Component type element is used to organize infor-

348 mation describing *Lower Level Structural Elements* or *Data Entities* associated with the
 349 Component.

350 If *Lower Level Structural Elements* are described, these elements are by definition child
 351 Component elements of a parent Component. At this next level, the *Lower Level* child
 352 Component elements are grouped into an XML container called Components.

353 This *Lower Level* Components container is comprised of one or more child Compo-
 354 nent XML elements representing the sub-parts of the parent Component. Just like the
 355 parent Component element, the child Component element is an abstract type XML el-
 356 ement and will never appear in the XML document – only the different *Lower Level* child
 357 Component types will appear.

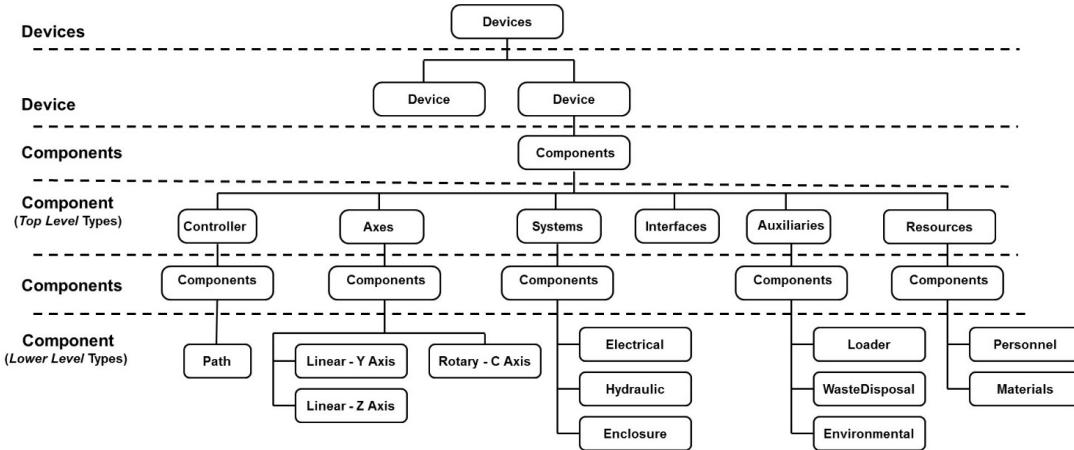
358 This parent-child relationship can continue to any depth required to fully define a piece of
 359 equipment.

360 Example 1 illustrates the relationship between a parent Component and *Lower Level*
 361 child components:

Example 1: Component Levels

```
362 1 <Devices>
363 2   <Device>
364 3     <Components>
365 4       <Axes> Parent Component
366 5       <Components>
367 6         <Rotary> Child component of Axes and Parent component of Lower Level compo-
368   nents
369 7           <Components>
370 8             <Chuck> Child Component of Rotary
```

371 Figure 1 demonstrates the various *Structural Elements* provided to describe a piece of
 372 equipment and the relationship between these elements.

**Figure 1:** Example Device Structural Elements

373 Component type XML elements **MAY** be further decomposed into Composition type
 374 XML elements. Composition elements describe the lowest level basic structural or
 375 functional building blocks contained within a Component. Any number of Composi-
 376 tion elements **MAY** be used. Data provided for a Component provides more specific
 377 meaning when it is associated with one of the Composition elements of the Compo-
 378 nent. The different Composition types that **MAY** appear in the XML document are
 379 defined in *Section 6 - Composition Type Structural Elements*.

380 The Composition elements are organized into a Compositions container. The
 381 Compositions container **MAY** appear in the XML document further describing a Com-
 382 ponent. If one or more Composition element(s) is provided to describe a Compo-
 383 nent, a Compositions container **MUST** be defined for the Component.

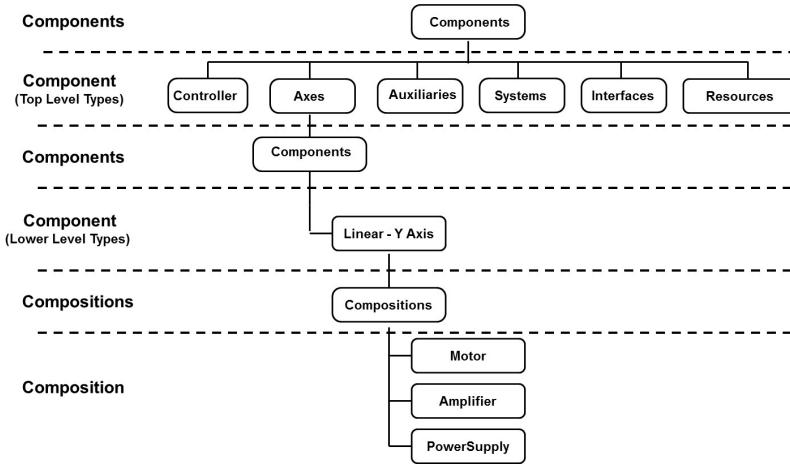
384 *Example 2* represents an XML document structure that demonstrates the relationship be-
 385 tween a parent Component and its Composition elements.

Example 2: Component levels with Composition

```

386 1 <Devices>
387 2   <Device>
388 3     <Components>
389 4       <Axes>  (Component)
390 5         <Components>
391 6           <Linear> (Component)
392 7             <Compositions>
393 8               <Composition>
394 9                 <Composition>
395 10                <Composition>
  
```

396 *Figure 2* demonstrates this relationship between a Component and some of its potential
 397 Composition elements.

**Figure 2:** Example Composition Structural Elements

398 4.1 Devices

399 Devices is a container type XML element that **MUST** contain only Device elements.
 400 Devices **MUST** contain at least one Device element, but **MAY** contain multiple De-
 401 vice elements. *Data Entities* **MAY NOT** be directly associated with the Devices con-
 402 tainer.

Table 1: MTConnect Devices Element

| Element | Description | Occurrence |
|---------|---|------------|
| Devices | The first, or highest level, <i>Structural Element</i> in a MTConnectDevices document. Devices is a container type XML element. | 1 |

403 4.2 Device

404 Device is an XML container type element that organizes the *Structural Elements* and
 405 *Data Entities* associated with a piece of equipment. *Data Entities* **MAY** be directly asso-
 406 ciated with the Device container. Device **MUST** provide the data item AVAILABIL-
 407 ITY, which represents the *Agent*'s ability to communicate with the data source.

408 In the MTConnectDevices XML document, Device is a unique type of *Structural*
 409 *Element*. Device carries all of the properties of a Component (See *Section 4.4 - Com-*
410 ponent). Additionally, Device **MUST** have a `uuid` attribute that uniquely identifies the
 411 piece of equipment. The value for the `uuid` **SHOULD NOT** change over time. The
 412 value for the `uuid` **MUST** be universally unique and **MUST** only appear once in any MT-
 413 Connect installation. All *Structural Elements* and *Data Entities* associated with a piece
 414 of equipment are therefore uniquely identified through their association with the Device
 415 container.

Table 2: MTConnect Device Element

| Element | Description | Occurrence |
|---------|---|------------|
| Device | The primary container element for each piece of equipment. Device is organized within the Devices container. There MAY be multiple Device elements in an XML document. | 1..* |

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

424 It is possible for a piece of equipment to be defined as both a Component of a Device
 425 and simultaneously function independently as a separate Device reporting data directly
 426 through an *Agent* using its own `uuid`. An example would be a temperature monitoring
 427 system that is defined as a Device reporting data about the environment within a facility
 428 and simultaneously reporting data for a Component of another piece of equipment that
 429 it is monitoring.

430 4.2.1 XML Schema Structure for Device

431 *Figure 3* represents the structure of the Device XML element showing the attributes
 432 defined for Device and the elements that may be associated with Device.

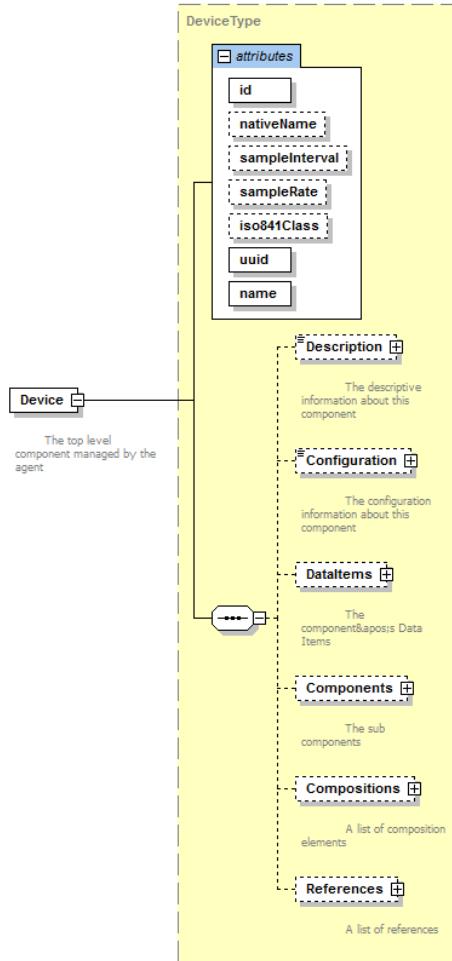


Figure 3: Device Diagram

433 4.2.2 Attribute for Device

434 *Table 3* defines the attributes that may be used to provide additional information for a
 435 Device type element.

Table 3: Attributes for Device

| Attribute | Description | Occurrence |
|-----------------------------|--|------------|
| <code>id</code> | <p>The unique identifier for this element.</p> <p><code>id</code> is a required attribute.</p> <p>An <code>id</code> MUST be unique across all the <code>id</code> attributes in the document.</p> <p>An XML ID-type.</p> | 1 |
| <code>nativeName</code> | <p>The common name normally associated with this piece of equipment.</p> <p><code>nativeName</code> is an optional attribute.</p> | 0..1 |
| <code>sampleInterval</code> | <p>An optional attribute that is an indication provided by a piece of equipment describing the interval in milliseconds between the completion of the reading of the data associated with the <code>Device</code> element until the beginning of the next sampling of that data. This indication is reported as the number of milliseconds between data captures.</p> <p>This information may be used by client software applications to understand how often information from a piece of equipment is expected to be refreshed.</p> <p>The refresh rate for all data from the piece of equipment will be the same as for the <code>Device</code> element unless specifically overridden by another <code>sampleInterval</code> provided for a Component of the <code>Device</code> element.</p> <p>If the value of <code>sampleInterval</code> is less than one millisecond, the value will be represented as a floating-point number. For example, an interval of 100 microseconds would be 0.1.</p> | 0..1 †† |
| <code>sampleRate</code> | <p>DEPRECATED in MTConnect Version 1.2. Replaced by <code>sampleInterval</code>.</p> | 0..1 ††† |
| <code>iso841Class</code> | DEPRECATED in MTConnect Version 1.1. | 0..1 ††† |

| Continuation of Table 3 | | |
|-------------------------|--|------------|
| Attribute | Description | Occurrence |
| uuid | <p>A unique identifier for this XML element.</p> <p>uuid is a required attribute.</p> <p>The uuid MUST be unique amongst all uuid identifiers used in an MTConnect installation.</p> <p>For example, this may be a combination of the manufacturer's code and serial number. The uuid SHOULD be alphanumeric and not exceed 255 characters.</p> <p>An NMOKEN XML type.</p> | 1 † |
| name | <p>The name of the piece of equipment represented by the Device element.</p> <p>name is a required attribute.</p> <p>This name MUST be unique for each Device XML element defined in the MTConnectDevices document.</p> <p>An NMOKEN XML type.</p> | 1 |

436 Notes: †A uuid **MUST** be provided for each Device element. It is optional for all
 437 other *Structural Elements*.

438 ††The sampleInterval is used to aid a client software application in interpreting values provided by some *Data Entities*. This is the desired sample
 439 interval and may vary depending on the capabilities of the piece of equipment.

440 †††Remains in schema for backwards compatibility.

442 4.2.3 Elements for Device

443 Table 4 lists the elements defined to provide additional information for a Device element.
 444 These elements are organized in the Device container.

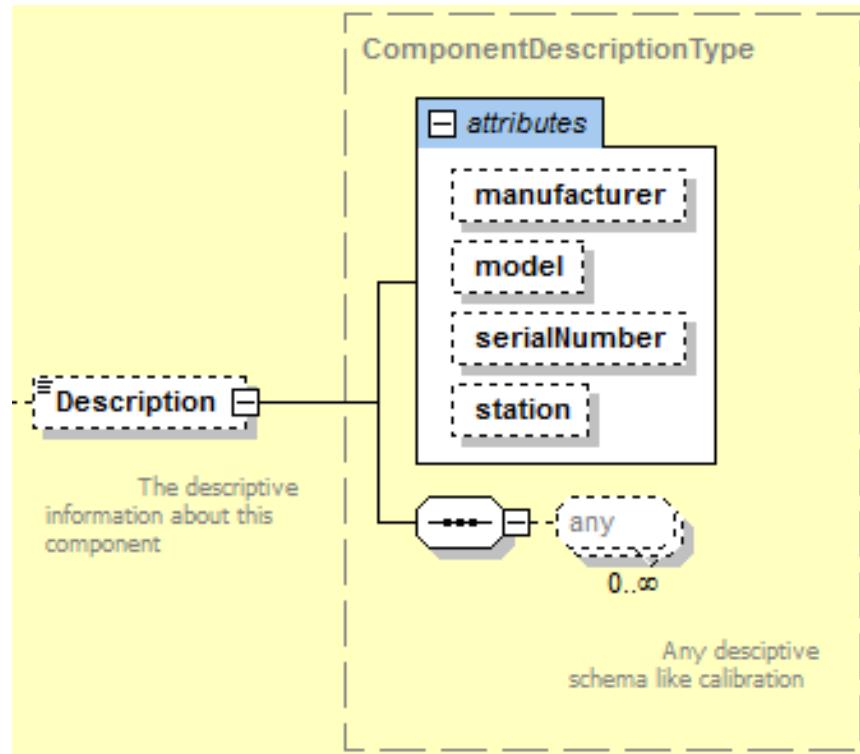
Table 4: Elements for Device

| Element | Description | Occurrence |
|---------------|---|----------------|
| Description | An XML element that can contain any descriptive content. | 0..1 |
| Configuration | An XML element that contains technical information about a piece of equipment describing its physical layout or functional characteristics. | 0..1 |
| DataItems | A container for the <i>Data Entities</i> (See <i>Section 7 - Data Entities for Device</i> and <i>Section 8 - Listing of Data Items</i> for more detail) provided by this <i>Device</i> element. | 1 [†] |
| Components | A container for the <i>Component</i> elements associated with this <i>Device</i> element. | 0..1 |
| Compositions | A container for the <i>Composition</i> elements associated with this <i>Device</i> element. | 0..1 |
| References | A container for the <i>Reference</i> elements associated with this <i>Device</i> element. | 0..1 |

445 Note: [†]DataItems **MUST** be provided since every piece of equipment **MUST**
 446 report AVAILABILITY.

447 **4.2.3.1 Description for Device**

448 *Figure 4* shows the structure of the *Description* XML element showing the attributes
 449 defined for *Description*. *Description* can contain any descriptive content for this
 450 piece of equipment. This element is defined to contain mixed content and additional XML
 451 elements (indicated by the *any* element) **MAY** be added to extend the schema for De-
 452 scription.

**Figure 4:** Description Diagram

453 *Table 5* lists the attributes defined for the `Description` XML element.

Table 5: Attributes for Description

| Attribute | Description | Occurrence |
|---------------------------|---|------------|
| <code>manufacturer</code> | The name of the manufacturer of the piece of equipment represented by the <code>Device</code> element. <code>manufacturer</code> is an optional attribute. | 0..1 |
| <code>model</code> | The model description of the piece of equipment represented by the <code>Device</code> element. <code>model</code> is an optional attribute. | 0..1 |
| <code>serialNumber</code> | The serial number associated with piece of equipment represented by the <code>Device</code> element. <code>serialNumber</code> is an optional attribute. | 0..1 |

| Continuation of Table 5 | | |
|-------------------------|---|------------|
| Attribute | Description | Occurrence |
| station | The station where the equipment represented by the Device element is located when it is part of a manufacturing unit or cell with multiple stations. station is an optional attribute. | 0..1 |

454 The content of Description **MAY** include any additional descriptive information the
 455 implementer chooses to include regarding a piece of equipment. This content **SHOULD**
 456 be limited to information not included elsewhere in the MTConnectDevices XML doc-
 457 ument.

Example 3: Example of Description

```
458 1 <Description manufacturer="Example Co"
459 2     serialNumber="A124FFF" station="2"> Example Co
460 3     Simulated Vertical 3 Axis Machining center.
461 4 </Description>
```

4.2.3.2 Configuration for Device

463 The Configuration XML element contains technical information about a piece of
 464 equipment. Configuration **MAY** include any information describing the physical
 465 layout or functional characteristics of the piece of equipment, such as capabilities, testing,
 466 installation, operation, calibration, or maintenance. Configuration **MAY** also include
 467 information representing the inter-relationships between pieces of equipment.

Table 6: MTConnect Configuration Element

| Element | Description | Occurrence |
|---------------|---|------------|
| Configuration | An XML element that contains technical information about a piece of equipment describing its physical layout, functional characteristics, and relationships with other pieces of equipment. | 0..1 |

468 Configuration data for Device is structured in the MTConnectDevices XML doc-
 469 ument as shown in *Figure 5*. AbstractConfiguration is an abstract type XML
 470 element. It will never appear in the XML document representing a piece of equipment.

471 When Configuration is provided for a piece of equipment, that type of Configuration
472 will appear in the XML document.

473 SensorConfiguration is described in detail in *Section 9.3 - Sensor Configuration*.

474 Relationships is described in detail in *Section 4.9 - Relationships*.

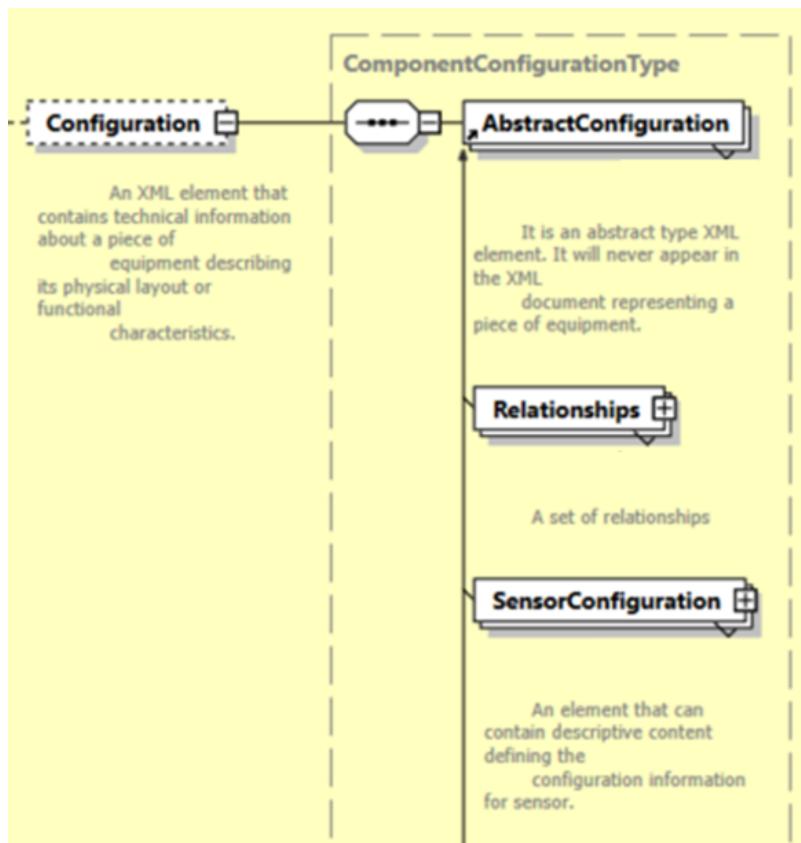


Figure 5: Configuration Diagram

4.2.3.3 DataItems for Device

476 DataItems is an XML container that provides structure for organizing the data reported
477 by a piece of equipment that is associated with the Device element.

478 DataItems **MUST** be provided since every piece of equipment **MUST** report the data
479 item AVAILABILITY.

480 See *Section 7 - Data Entities for Device* and *Section 8 - Listing of Data Items* for details
481 on the DataItems XML element.

482 **4.2.3.4 Components within Device**

483 The use of the XML container `Components` within a `Device` element provides the
 484 ability to break down the structure of a `Device` element into *Top Level* and *Lower Level*
 485 physical and logical sub-parts. If a `Components` XML element is provided, then only
 486 one `Components` element **MUST** be defined for a `Device` element.

487 **4.2.3.5 Compositions for Device**

488 `Compositions` is an XML container used to organize `Composition` elements asso-
 489 ciated with a `Device` element. See *Section 4.5 - Compositions* for details on Composi-
 490 tions.

491 **4.2.3.6 References for Device**

492 `References` is an XML container used to organize `References` elements associated
 493 with a `Device` element. See *Section 4.7 - References* for details on References.

494 **4.3 Components**

495 `Components` is an XML container used to group information describing physical parts
 496 or logical functions of a piece of equipment. `Components` contains one or more `Com-`
 497 `ponent` XML elements.

Table 7: MTConnect Components Element

| Element | Description | Occurrence |
|-------------------------|--|------------|
| <code>Components</code> | An XML container that consists of one or more types of <code>Component</code> XML elements. If a <code>Components</code> XML element is provided, then only one <code>Components</code> element MUST be defined for a <code>Device</code> element. | 0..1 |

498 4.4 Component

499 A Component XML element is a container type XML element used to organize information
 500 describing a physical part or logical function of a piece of equipment. It also provides
 501 structure for describing the *Lower Level Structural Elements* associated with the Compo-
 502 nent. Component is an abstract type XML element and will never appear directly in
 503 the MTConnect XML document. As an abstract type XML element, Component will be
 504 replaced in the XML document by specific Component types. XML elements represent-
 505 ing Component are described in *Section 5 - Component Structural Elements* and include
 506 elements such as Axes, Controller, and Systems.

Table 8: MTConnect Component Element

| Element | Description | Occurrence |
|-----------|--|------------|
| Component | <p>An abstract XML element. Replaced in the XML document by types of Component elements representing physical parts and logical functions of a piece of equipment.</p> <p>There can be multiple types of Component XML elements in the document.</p> | 1..* |

507 4.4.1 XML Schema Structure for Component

508 *Figure 6* represents the structure of a Component XML element showing the attributes
 509 defined for Component and the elements that **MAY** be associated with Component.

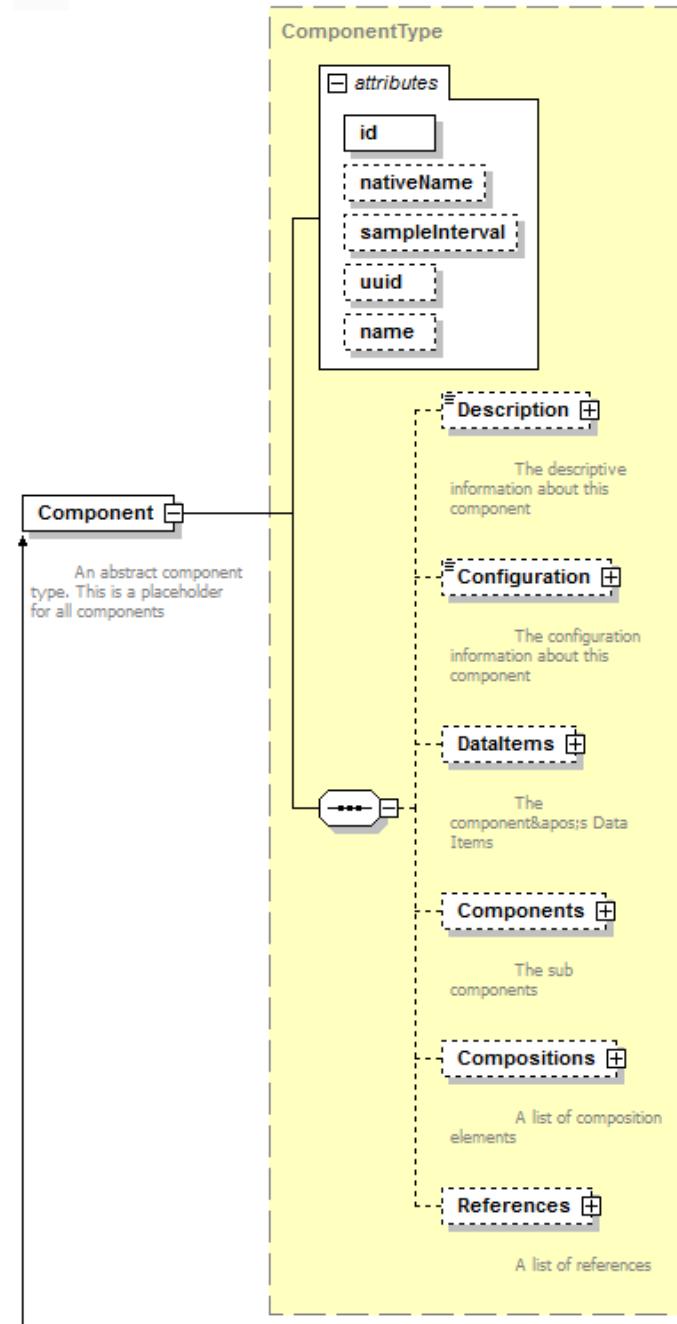


Figure 6: Component Diagram

510 4.4.2 Attribute for Component

511 *Table 9* defines the attributes that may be used to provide additional information for a
 512 Component type XML element.

Table 9: Attributes for Component

| Attribute | Description | Occurrence |
|-------------------------|--|------------|
| <code>id</code> | <p>The unique identifier for this element. <code>id</code> is a required attribute. An <code>id</code> MUST be unique across all the <code>id</code> attributes in the document. An XML ID-type.</p> | 1 |
| <code>nativeName</code> | <p>The common name normally associated with a specific physical or logical part of a piece of equipment. <code>nativeName</code> is an optional attribute.</p> | 0..1 |

| Continuation of Table 9 | | |
|-------------------------|---|------------|
| Attribute | Description | Occurrence |
| sampleInterval | <p>An optional attribute that is an indication provided by a piece of equipment describing the interval in milliseconds between the completion of the reading of the data associated with the Component element until the beginning of the next sampling of that data. This indication is reported as the number of milliseconds between data captures.</p> <p>This information may be used by client software applications to understand how often information from a piece of equipment for a specific Component element is expected to be refreshed.</p> <p>The refresh rate for data from all <i>Lower Level</i> Component elements will be the same as for the parent Component element unless specifically overridden by another sampleInterval provided for the <i>Lower Level</i> Component element.</p> <p>If the value of sampleInterval is less than one millisecond, the value will be represented as a floating-point number. For example, an interval of 100 microseconds would be 0.1.</p> | 0..1 †† |
| sampleRate | DEPRECATED in MTConnect Version 1.2. Replaced by sampleInterval. | 0..1 ††† |

| Continuation of Table 9 | | |
|-------------------------|---|-------------------|
| Attribute | Description | Occurrence |
| uuid | <p>A unique identifier for this XML element.</p> <p>uuid is an optional attribute.</p> <p>The value provided for the uuid MUST be unique amongst all uuid identifiers used in an MTConnect installation.</p> <p>For example, this may be a combination of the manufacturer's code and serial number. The uuid SHOULD be alphanumeric and not exceed 255 characters.</p> <p>An NMOKEN XML type.</p> | 0..1 [†] |
| name | <p>The name of the Component element.</p> <p>name is an optional attribute.</p> <p>However, if there are multiple <i>Lower Level</i> components that have the same parent and are of the same component type (example Linear), then the name attribute MUST be provided for all <i>Lower Level</i> components of the same element type to differentiate between the similar components.</p> <p>When provided, name MUST be unique for all <i>Lower Level</i> components of a parent Component.</p> <p>An NMOKEN XML type.</p> | 0..1 |

513 Notes: [†]While uuid **MUST** be provided for the Device element, it is optional for
 514 Component elements.

515 ^{††}The sampleInterval is used to aid a client software application in interpreting values provided by some *Data Entities*. This is the desired sample
 516 interval and may vary depending on the capabilities of the piece of equipment.

517 ^{†††}Remains in schema for backwards compatibility.

519 4.4.3 Elements of Component

520 *Table 10* lists the elements defined to provide additional information for a Component
 521 type XML element.

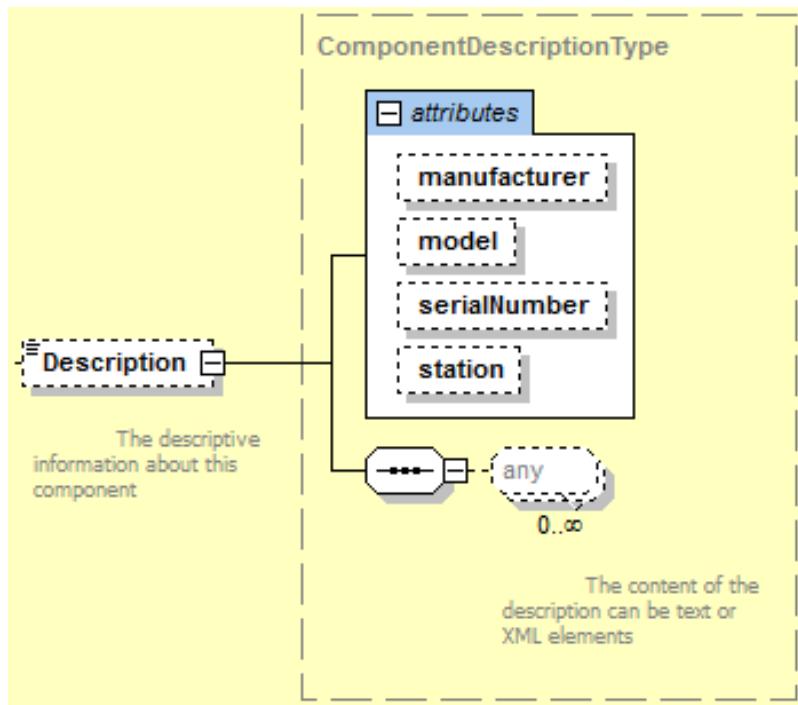
Table 10: Elements for Component

| Element | Description | Occurrence |
|---------------|--|-------------------|
| Description | An element that can contain any descriptive content. | 0..1 |
| Configuration | An XML element that contains technical information about a piece of equipment describing its physical layout or functional characteristics. | 0..1 |
| DataItems | A container for the <i>Data Entities</i> (defined in <i>Section 8 - Listing of Data Items</i>) associated with this Component element. | 0..1 [†] |
| Components | A container for <i>Lower Level</i> Component XML elements associated with this parent Component. | 0..1 [†] |
| Compositions | A container for the Composition elements (defined in <i>Section 6 - Composition Type Structural Elements</i>) associated with this Component element. | 0..1 |
| References | A container for the Reference elements associated with this Component element. | 0..1 [†] |

522 Note: [†]At least one of Components, DataItems, or References **MUST** be
 523 provided.

524 4.4.3.1 Description for Component

525 *Figure 7* illustrates the structure of the Description XML element showing the attributes defined for Description. Description can contain any descriptive content
 526 of this Component. This element is defined to contain mixed content and additional
 527 XML elements (indicated by the any element) **MAY** be added to extend the schema for
 528 Description.
 529

**Figure 7:** Description of Component Diagram

530 *Table 11* lists the attributes defined for the `Description` XML element.

Table 11: Attributes for Description for Component

| Attribute | Description | Occurrence |
|---------------------------|---|------------|
| <code>manufacturer</code> | The name of the manufacturer of the physical or logical part of a piece of equipment represented by the <code>Component</code> element. <code>manufacturer</code> is an optional attribute. | 0..1 |
| <code>model</code> | The model description of the physical part or logical function of a piece of equipment represented by the <code>Component</code> element. <code>model</code> is an optional attribute. | 0..1 |
| <code>serialNumber</code> | The serial number associated with the physical part or logical function of a piece of equipment represented by the <code>Component</code> element. <code>serialNumber</code> is an optional attribute. | 0..1 |

| Continuation of Table 11 | | |
|--------------------------|---|------------|
| Attribute | Description | Occurrence |
| station | <p>The station where the physical part or logical function of a piece of equipment represented by the Component element is located when it is part of a manufacturing unit or cell with multiple stations.</p> <p>station is an optional attribute.</p> | 0..1 |

531 The content of Description **MAY** include any additional descriptive information the
 532 implementer chooses to include regarding the Component element. This content **SHOULD**
 533 be limited to information not included elsewhere in the MTConnectDevices XML doc-
 534 ument.

Example 4: Example of Description

```
535 1 <Description manufacturer="Example Co"
536 2     serialNumber="EXCO-TT-099PP-XXXX"> Advanced Pulse
537 3     watt-hour transducer with pulse output
538 4 </Description>
```

539 **4.4.3.2 Configuration for Component**

540 The Configuration XML element contains technical information about a component.
 541 Configuration **MAY** include any information describing the physical layout or func-
 542 tional characteristics of a component, such as capabilities, testing, installation, operation,
 543 calibration, or maintenance. Configuration **MAY** also include information represent-
 544 ing the inter-relationships between components within a piece of equipment.

Table 12: MTConnect Configuration Element for Component

| Element | Description | Occurrence |
|---------------|---|------------|
| Configuration | An XML element that contains technical information about a component describing its physical layout, functional characteristics, and relationships with other components within a piece of equipment. | 0..1 |

545 Configuration data for Component is structured in the MTConnectDevices XML

546 document as shown in *Figure 8*. `AbstractConfiguration` is an abstract type XML
 547 element. It will never appear in the XML document representing a piece of equipment.
 548 When `Configuration` is provided for a component, that type of Configuration
 549 will appear in the XML document.

550 `SensorConfiguration` is described in detail in *Section 9.3 - Sensor Configuration*.

551 `Relationships` is described in detail in *Section 4.9 - Relationships*.

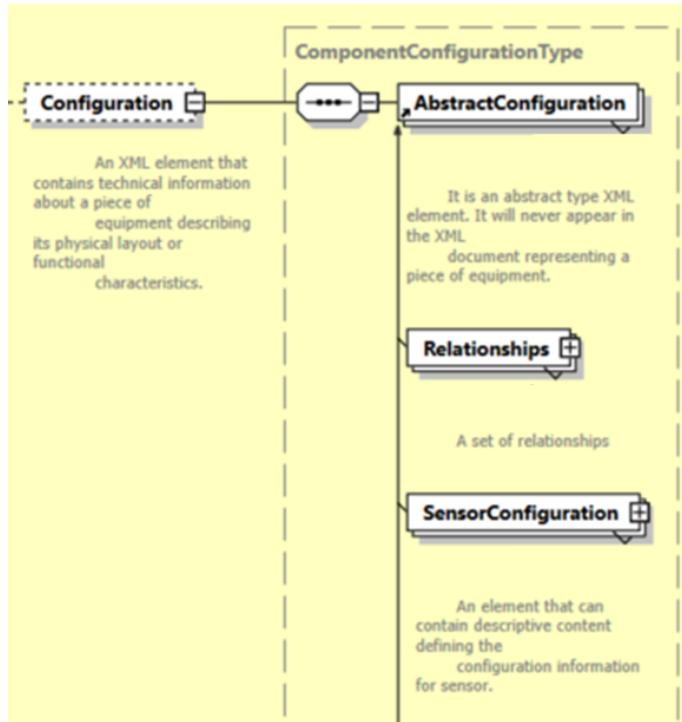


Figure 8: Component Configuration Diagram

552 4.4.3.3 DataItems for Component

553 `DataItems` is an XML container that provides structure for organizing the data reported
 554 by a piece of equipment that is associated with the Component.

555 See *Section 7 - Data Entities for Device* for details on the `DataItems` XML element.

556 **4.4.3.4 Components within Component**

557 The use of the XML container `Components` within a `Component` element provides
 558 the ability to further break down the structure of a `Component` element into even *Lower*
 559 *Level* physical and logical sub-parts. These *Lower Level* elements can add more clarity
 560 and granularity to the physical or logical structure of a piece of equipment and the data
 561 associated with that equipment.

562 This parent-child relationship can be extended down to any level necessary to fully de-
 563 scribe a piece of equipment. These *Lower Level* `Component` elements use the same XML
 564 structure as `Component` defined in *Section 4.4.1 - XML Schema Structure for Component*.

Example 5: Example of parent Component and Child Elements

```
565 1 <Devices>
566 2   <Device>
567 3     <Components>
568 4       <Axes> (Component)
569 5       <Components>
570 6         <Linear> (Component)
571 7         <Components>
572 8           <Etc. > (Component)
```

573 **4.4.3.5 Compositions for Component**

574 `Compositions` is an XML container used to organize the lowest level structural build-
 575 ing blocks contained within a `Component` as defined below.

576 **4.4.3.6 References for Component**

577 `References` is an XML container used to organize `Reference` elements associated
 578 with a `Component` element. See *Section 4.7 - References* for details on `References`.

579 **4.5 Compositions**

580 `Compositions` is an XML container that defines the lowest level structural building
 581 blocks contained within a `Component` element.

582 `Compositions` contains one or more `Composition` XML elements.

Table 13: MTConnect Compositions Element

| Element | Description | Occurrence |
|--------------|--|------------|
| Compositions | An XML container consisting of one or more types of Composition XML elements. Only one Compositions container MAY appear for a Component element. | 0..1 |

583 4.6 Composition

584 Composition XML elements are used to describe the lowest level physical building
 585 blocks of a piece of equipment contained within a Component.

586 Like Component elements, Composition elements provide the ability to organize in-
 587 formation describing *Lower Level* sub-parts of a higher-level Component element. How-
 588 ever, unlike Component, Composition **MUST NOT** be further sub-divided and *Data*
 589 *Entities* **MUST NOT** be assigned to Composition elements.

590 Composition elements are used to add more clarity and granularity to the data being
 591 retrieved from a piece of equipment. The meaning of the data associated with a Com-
 592 ponent may be enhanced by designating a specific Composition element associated
 593 with that data.

594 An example of the additional detail provided when using Composition elements would
 595 be:

596 A TEMPERATURE associated with a Linear type axis may be further clarified by ref-
 597 erencing the MOTOR or AMPLIFIER type Composition element associated with that
 598 axis, which differentiates the temperature of the motor from the temperature of the ampli-
 599 fier.

600 Composition is a typed XML element and will always define a specific type of struc-
 601 tural building block contained within a Component. XML elements representing the
 602 types of Composition elements are described in *Section 6 - Composition Type Struc-*
603 tural Elements and include elements describing such basic building blocks as motors, am-
 604 plifiers, filters, and pumps.

Example 6: Example of parent Component and child Composition elements

```
605 1 <Devices>
606 2   <Device>
607 3     <Components>
```

```

608 4      <Axes> (Component)
609 5      <Components>
610 6      <Linear> (Component)
611 7      <Compositions>
612 8      <Composition>
613 9      <Composition>
614 10     <Composition>

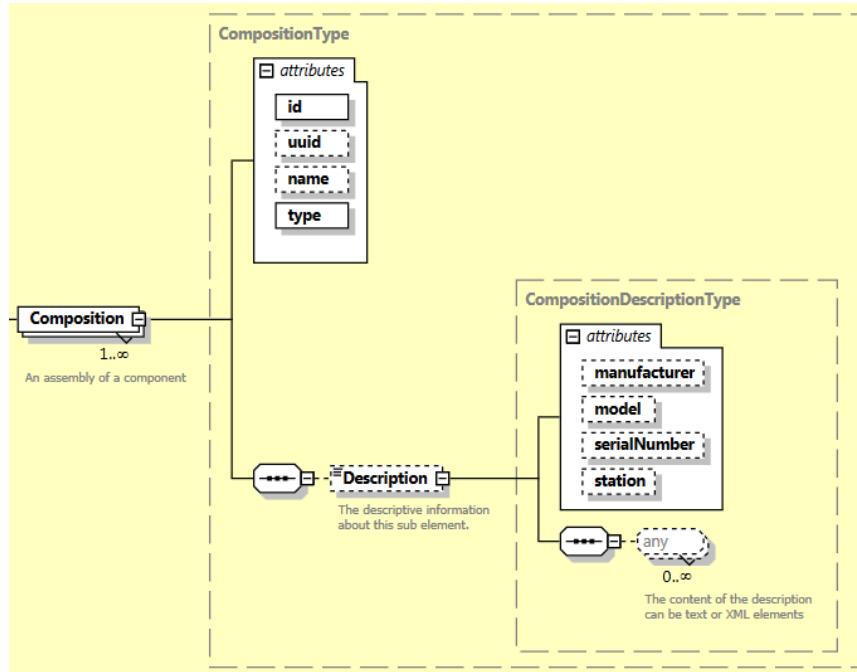
```

Table 14: MTConnect Composition Element

| Element | Description | Occurrence |
|-------------|---|------------|
| Composition | <p>An XML element used to describe the lowest level structural building blocks contained within a Component element.</p> <p>Composition is a typed XML element.</p> <p>There can be multiple types of Composition XML elements defined for a Component element.</p> | 1..* |

615 **4.6.1 XML Schema Structure for Composition**

616 *Figure 9* illustrates a Composition XML element showing the attributes defined for
 617 Composition and the elements that may be associated with Composition type XML
 618 elements.

**Figure 9:** Composition Diagram

619 4.6.2 Attributes for Composition

620 *Table 15* defines the attributes that may be used to provide additional information for a
621 Composition type XML element.

Table 15: Attributes for Composition

| Attribute | Description | Occurrence |
|-----------------|--|------------|
| <code>id</code> | <p>The unique identifier for this element. <code>id</code> is a required attribute. An <code>id</code> MUST be unique across all the <code>id</code> attributes in the document. An XML ID-type.</p> | 1 |

| Continuation of Table 15 | | |
|--------------------------|--|------------|
| Attribute | Description | Occurrence |
| uuid | <p>A unique identifier for this XML element.</p> <p>uuid is an optional attribute.</p> <p>The uuid MUST be unique amongst all uuid identifiers used in an MTConnect installation.</p> <p>For example, this may be a combination of the manufacturer's code and serial number. The uuid SHOULD be alphanumeric and not exceed 255 characters.</p> <p>An NMTOKEN XML type.</p> | 0..1 |
| name | <p>The name of the Composition element.</p> <p>name is an optional attribute.</p> <p>If provided, name MUST be unique within a Component element.</p> <p>An NMTOKEN XML type.</p> | 0..1 |
| type | <p>The type of Composition element.</p> <p>type is a required attribute.</p> <p>Examples of types are MOTOR, FILTER, PUMP, and AMPLIFIER.</p> <p>Refer to <i>Section 6 - Composition Type Structural Elements</i> for a list of currently defined types.</p> | 1 |

622 4.6.3 Elements of Composition

623 Table 16 lists the elements defined to provide additional information for a Composition
 624 type XML element.

Table 16: Elements for Composition

| Element | Description | Occurrence |
|-------------|--|------------|
| Description | An element that can contain any descriptive content. | 0..1 |

625 4.6.3.1 Description for Composition

626 *Figure 10* represents the structure of the Description XML element showing the attributes defined for Description. Description can contain any descriptive content
 627 for this Composition element. This element is defined to contain mixed content and
 628 additional XML elements (indicated by the any element) **MAY** be added to extend the
 629 schema for Description.
 630

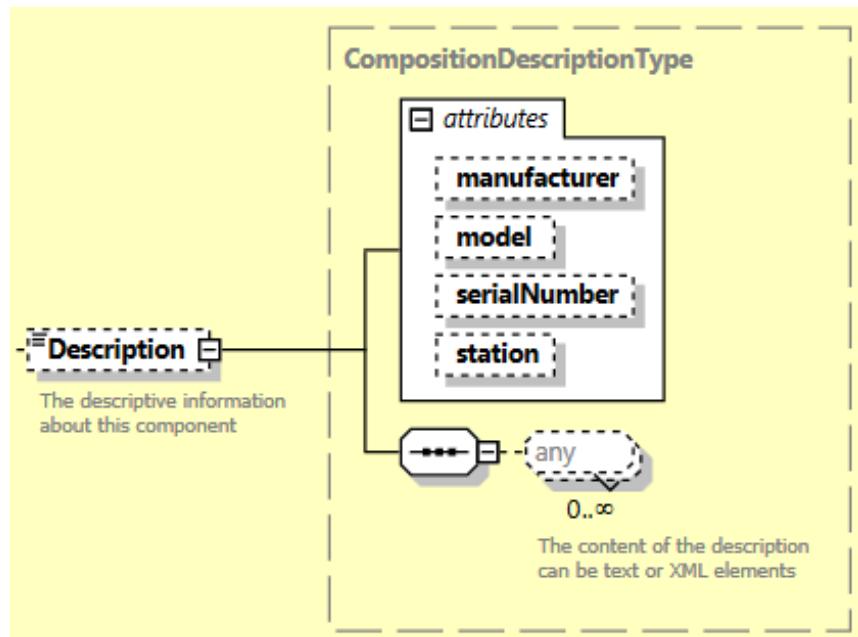


Figure 10: Description of Composition Diagram

631 *Table 17* lists the attributes defined for the Description XML element.

Table 17: Attributes for Description for Composition

| Attribute | Description | Occurrence |
|---------------------------|--|------------|
| <code>manufacturer</code> | The name of the manufacturer of the physical part of a piece of equipment represented by the Composition element. <code>manufacturer</code> is an optional attribute. | 0..1 |
| <code>model</code> | The model description of the physical part of a piece of equipment represented by the Composition element. <code>model</code> is an optional attribute. | 0..1 |

| Continuation of Table 17 | | |
|--------------------------|---|------------|
| Attribute | Description | Occurrence |
| serialNumber | The serial number associated with the physical part of a piece of equipment represented by the Composition element. serialNumber is an optional attribute. | 0..1 |
| station | The station where the physical part of a piece of equipment represented by the Composition element is located when it is part of a manufacturing unit or cell with multiple stations. station is an optional attribute. | 0..1 |

632 The content of **Description** **MAY** include any additional descriptive information the
 633 implementer chooses to include regarding the **Composition** element. This content
 634 **SHOULD** be limited to information not included elsewhere in the **MTConnectDevices**
 635 XML document.

Example 7: Example of Description

```
636 1 <Description manufacturer="Example Co"  

  637 2     serialNumber="A124FFF" station="2"> Spindle motor  

  638 3     associated with Path 2.  

  639 4 </Description>
```

640 4.7 References

641 **References** is an XML container that organizes pointers to information defined elsewhere within the XML document for a piece of equipment.

643 **References** may be modeled as part of a **Device**, **Component** or **Interface** type
 644 *Structural Element*.

645 **References** contains one or more **Reference** XML elements.

Table 18: MTConnect References Element

| Element | Description | Occurrence |
|------------|---|------------|
| References | An XML container consisting of one or more types of Reference XML elements. Only one References container MUST appear for a Device, Component, or Interface element. | 0..1 |

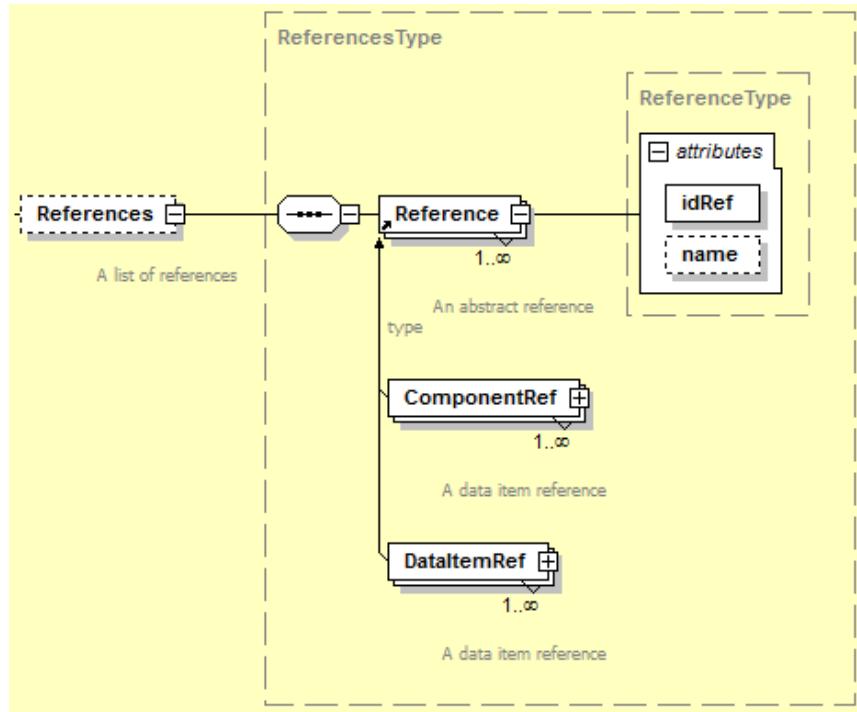
646 4.8 Reference

647 Reference is a pointer to information that is associated with another *Structural Element*
 648 defined elsewhere in the XML document for a piece of equipment. That information may
 649 be data from the other element or the entire structure of that element.

650 Reference is an efficient method to associate information with an element without du-
 651 plicating any of the data or structure. For example, a Bar Feeder System may make a re-
 652 quest for the BarFeederInterface and receive all the relevant data for the interface
 653 and the associated spindle (Rotary element) that is referenced as part of the BarFeed-
 654 erInterface.

655 Reference is an abstract type XML element and will never appear directly in the MT-
 656 Connect XML document. As an abstract type XML element, Reference will be re-
 657 placed in the XML document by a specific Reference type. The current supported
 658 types of Reference are DataItemRef and ComponentRef XML elements.

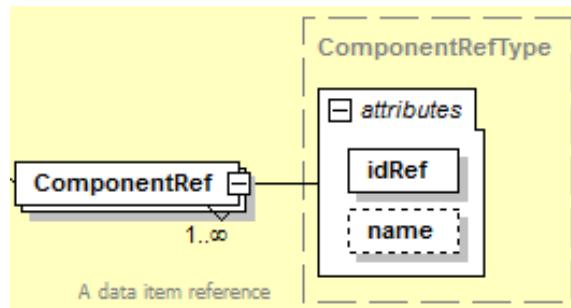
659 *Figure 11* represents the structure of the Reference XML element.

**Figure 11:** Reference Diagram

660 4.8.1 ComponentRef

661 ComponentRef XML element is a pointer to all of the information associated with another *Structural Element* defined elsewhere in the XML document for a piece of equipment.
 662 ComponentRef allows all of the information (*Lower Level Components* and all
 663 *Data Entities*) that is associated with the other *Structural Element* to be directly associated
 664 with this XML element.
 665

666 *Figure 12* represents the structure of a ComponentRef XML element showing the attributes defined for ComponentRef.
 667

**Figure 12:** ComponentRef Diagram

668 *Table 19* lists the attributes defined for the ComponentRef element.

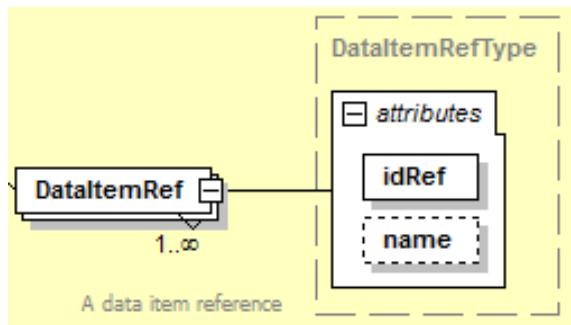
Table 19: Attributes for ComponentRef

| Attribute | Description | Occurrence |
|-----------|--|------------|
| idRef | A pointer to the <code>id</code> attribute of the Component that contains the information to be associated with this XML element. <code>idRef</code> is a required attribute. | 1 |
| name | The name of the ComponentRef element. <code>name</code> is an optional attribute. However, if there are multiple ComponentRef elements defined for a Component, the <code>name</code> attribute MUST be provided for all ComponentRef elements to differentiate between the similar elements. When provided, <code>name</code> MUST be unique for all ComponentRef elements associated with the <i>Parent Element</i> . An NMTOKEN XML type. | 0..1 |

669 4.8.2 DataItemRef

670 DataItemRef XML element is a pointer to a *Data Entity* associated with another *Structural Element* defined elsewhere in the XML document for a piece of equipment. DataItemRef allows the data associated with a data item defined in another *Structural Element* to be directly associated with this XML element.

674 *Figure 13* represents the structure of a DataItemRef XML element showing the attributes defined for DataItemRef.

**Figure 13:** DataItemRef Diagram

676 *Table 20* lists the attributes defined for the `DataItemRef` element.

Table 20: Attributes for DataItemRef

| Attribute | Description | Occurrence |
|--------------------|--|------------|
| <code>idRef</code> | A pointer to the <code>id</code> attribute of the <code>DataItem</code> that contains the information to be associated with this XML element. <code>idRef</code> is a required attribute. | 1 |
| <code>name</code> | The name of the <code>DataItemRef</code> element. <code>name</code> is an optional attribute. However, if there are multiple <code>DataItemRef</code> elements defined for a Component, the <code>name</code> attribute MUST be provided for all <code>DataItemRef</code> elements to differentiate between the similar elements. When provided, <code>name</code> MUST be unique for all <code>DataItemRef</code> elements associated with the <i>Parent Element</i> . An NMTOKEN XML type. | 0..1 |

677 4.9 Relationships

678 Relationships is an XML container that organizes information defining the association
 679 between pieces of equipment that function independently but together perform a manufacturing
 680 operation. Relationships may also define the association between components within a piece of equipment.

682 Relationships may be modeled as part of a Device or a Component *Structural Element*.

684 Relationships contains one or more Relationship XML elements.

Table 21: MTConnect Relationships Element

| Element | Description | Occurrence |
|---------------|---|------------|
| Relationships | <p>XML container consisting of one or more Relationship XML elements.</p> <p>Only one Relationships container MUST appear for a Device or a Component element.</p> | 0..1 |

685 4.10 Relationship

686 Relationship is an XML element that describes the association between two pieces
 687 of equipment that function independently but together perform a manufacturing operation.
 688 Relationship may also be used to define the association between two components
 689 within a piece of equipment.

690 Relationship is an abstract type XML element, Relationship will be replaced
 691 in the XML document by specific Relationship types. XML elements representing
 692 Relationship are described in *Section 4.10.1 - DeviceRelationship* and *Section 4.10.2
 693 - ComponentRelationship*.

694 A separate Relationship type element **MAY** be defined to describe each pair of associations
 695 with a piece of equipment or between Component elements within a piece of equipment.

697 Pieces of equipment may only be associated with other pieces of equipment and Component elements may only be associated with other Component elements within a specific
 698 piece of equipment.
 699

700 The XML schema diagram in *Figure 14* represents the structure of the Relationship XML element.
 701

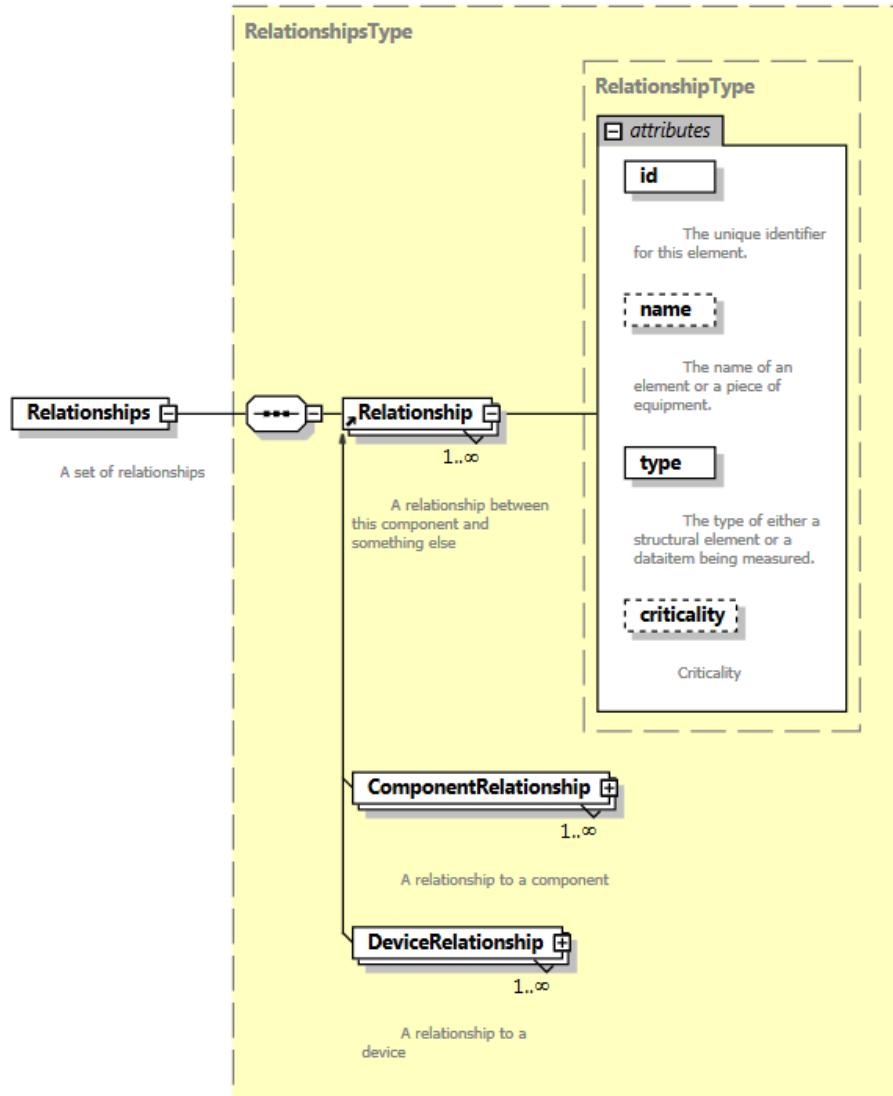


Figure 14: Relationship Diagram

702 **4.10.1 DeviceRelationship**

703 DeviceRelationship describes the association between two pieces of equipment that
704 function independently but together perform a manufacturing operation.

705 The XML schema diagram in *Figure 15* represents the structure of a DeviceRela-
706 tionship XML element showing the attributes defined for DeviceRelationship.

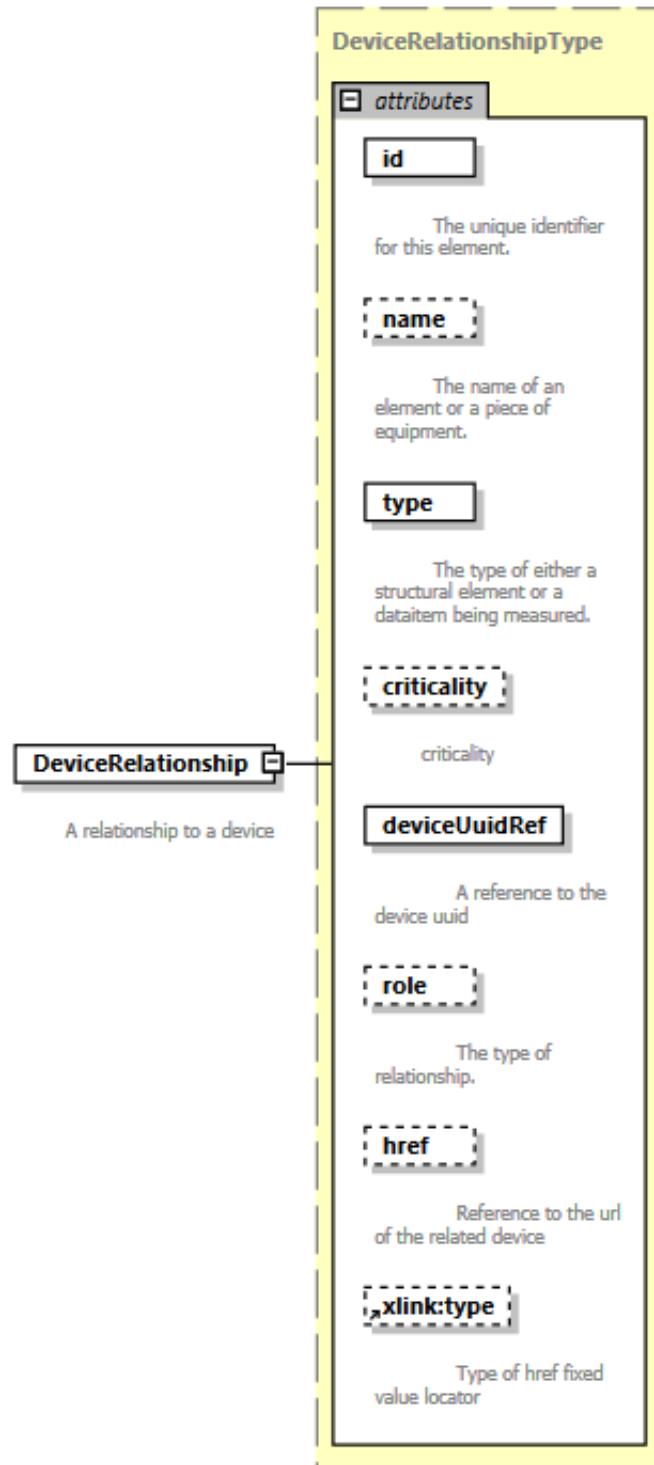


Figure 15: DeviceRelationship Diagram

707 The *Table 22* lists the attributes defined for the `DeviceRelationship` element.

Table 22: Attributes for `DeviceRelationship`

| Attribute | Description | Occurrence |
|-------------------|--|------------|
| <code>id</code> | <p>The unique identifier for this <code>DeviceRelationship</code>. <code>id</code> is a required attribute. The <code>id</code> attribute MUST be unique within the <code>MTConnectDevices</code> document. An XML ID-type.</p> | 1 |
| <code>name</code> | <p>The name associated with this <code>DeviceRelationship</code>. <code>name</code> is provided as an additional human readable identifier for this <code>DeviceRelationship</code>. <code>name</code> is an optional attribute. An NMTOKEN XML type.</p> | 0..1 |
| <code>type</code> | <p>Defines the authority that this piece of equipment has relative to the associated piece of equipment. <code>type</code> is a required attribute. The value provided for <code>type</code> MUST be one of the following values: PARENT: This piece of equipment functions as a parent in the relationship with the associated piece of equipment. CHILD: This piece of equipment functions as a child in the relationship with the associated piece of equipment. PEER: This piece of equipment functions as a peer which provides equal functionality and capabilities in the relationship with the associated piece of equipment.</p> | 1 |

| Continuation of Table 22 | | |
|--------------------------|---|------------|
| Attribute | Description | Occurrence |
| criticality | <p>Defines whether the services or functions provided by the associated piece of equipment is required for the operation of this piece of equipment.</p> <p>criticality is an optional attribute.</p> <p>The value provided for criticality MUST be one of the following values:</p> <ul style="list-style-type: none"> CRITICAL: The services or functions provided by the associated piece of equipment is required for the operation of this piece of equipment. NONCRITICAL: The services or functions provided by the associated piece of equipment is not required for the operation of this piece of equipment. | 0..1 |
| deviceUuidRef | <p>A reference to the associated piece of equipment.</p> <p>The value provided for deviceUuidRef MUST be the value provided for the uuid attribute of the Device element of the associated piece of equipment.</p> <p>deviceUuidRef is a required attribute.</p> <p>An NMTOKEN XML type.</p> | 1 |

| Continuation of Table 22 | | |
|--------------------------|--|------------|
| Attribute | Description | Occurrence |
| role | <p>Defines the services or capabilities that the referenced piece of equipment provides relative to this piece of equipment.</p> <p>role is an optional attribute.</p> <p>The value provided for role MUST be one of the following values:</p> <ul style="list-style-type: none"> SYSTEM: The associated piece of equipment performs the functions of a System for this piece of equipment. In MTConnect, System provides utility type services to support the operation of a piece of equipment and these services are required for the operation of a piece of equipment. AUXILIARY: The associated piece of equipment performs the functions as an Auxiliary for this piece of equipment. In MTConnect, Auxiliary extends the capabilities of a piece of equipment, but is not required for the equipment to function. | 0..1 |
| href | <p>A URI identifying the <i>Agent</i> that is publishing information for the associated piece of equipment. href MUST also include the UUID for that specific piece of equipment.</p> <p>href is of type <code>xlink:href</code> from the W3C XLink specification: (https://www.w3.org/TR/xlink11/).</p> <p>href is an optional attribute.</p> | 0..1 |
| xlink:type | <p>The XLink type attribute MUST have a fixed value of locator as defined in W3C XLink 1.1 https://www.w3.org/TR/xlink11/ section 5.4 <i>Locator Attribute (href)</i>.</p> <p>If the href attribute is provided, it MUST conform to the URI syntactic rules as defined in IETF RFC 3986 for Uniform Resource Identifiers. (https://www.ietf.org/rfc/rfc3986.txt)</p> | 0..1 |

708 4.10.2 ComponentRelationship

709 ComponentRelationship describes the association between two components within
 710 a piece of equipment that function independently but together perform a capability or
 711 service within a piece of equipment.

712 The XML schema in *Figure 16* represents the structure of a ComponentRelationship-
 713 ship XML element showing the attributes defined for ComponentRelationship.

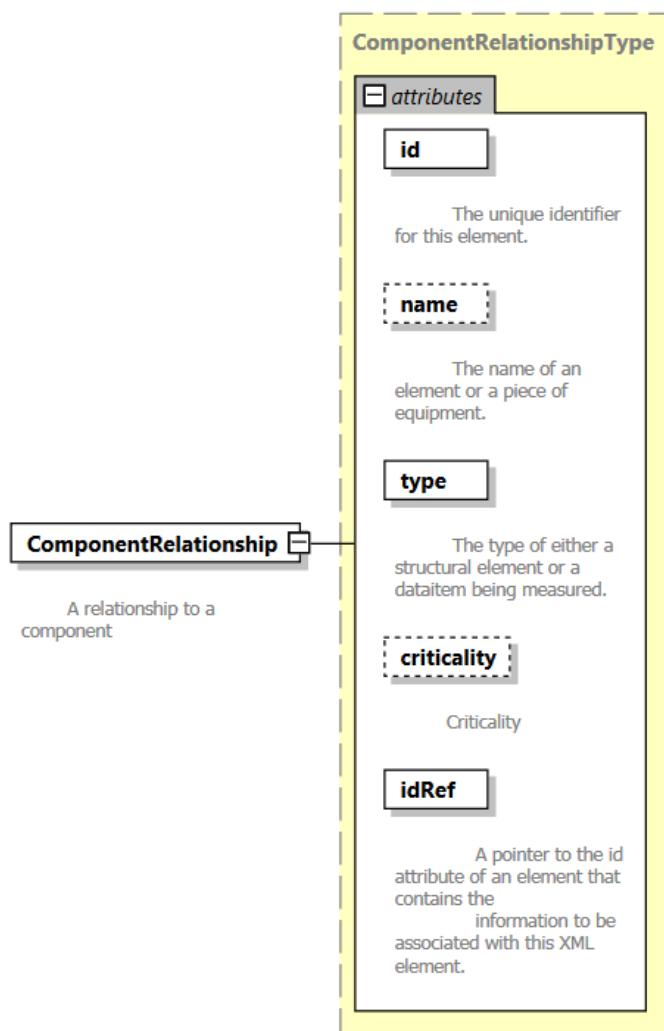


Figure 16: ComponentRelationship Diagram

714 The *Table 23* lists the attributes defined for the ComponentRelationship element.

Table 23: Attributes for ComponentRelationship

| Attribute | Description | Occurrence |
|-----------|--|------------|
| id | <p>The unique identifier for this ComponentRelationship.</p> <p>id is a required attribute.</p> <p>The id attribute MUST be unique within the MTConnectDevices document.</p> <p>An XML ID-type.</p> | 1 |
| name | <p>The name associated with this ComponentRelationship.</p> <p>name is provided as an additional human readable identifier for this ComponentRelationship.</p> <p>name is an optional attribute.</p> <p>An NMTOKEN XML type.</p> | 0..1 |
| type | <p>Defines the authority that this component element has relative to the associated component element.</p> <p>type is a required attribute.</p> <p>The value provided for type MUST be one of the following values:</p> <ul style="list-style-type: none"> PARENT: This component functions as a parent in the relationship with the associated component element. CHILD: This component functions as a child in the relationship with the associated component element. PEER: This component functions as a peer which provides equal functionality and capabilities in the relationship with the associated component element. | 1 |

| Continuation of Table 23 | | |
|--------------------------|--|------------|
| Attribute | Description | Occurrence |
| criticality | <p>Defines whether the services or functions provided by the associated component element is required for the operation of this piece of equipment.</p> <p>criticality is an optional attribute.</p> <p>The value provided for criticality MUST be one of the following values:</p> <ul style="list-style-type: none"> CRITICAL: The services or functions provided by the associated component element is required for the operation of this piece of equipment. NONCRITICAL: The services or functions provided by the associated component element is not required for the operation of this piece of equipment. | 0..1 |
| idRef | <p>A reference to the associated component element.</p> <p>The value provided for idRef MUST be the value provided for the id attribute of the associated Component element.</p> <p>idRef is a required attribute.</p> <p>An NMTOKEN XML type.</p> | 1 |

715 5 Component Structural Elements

716 Component *Structural Elements* are XML containers used to represent physical parts or
 717 logical functions of a piece of equipment.

718 Component *Structural Elements* are defined into two major categories:

- 719 • *Top Level* Component elements are used to group the *Structural Elements* repre-
 720 senting the most significant physical or logical functions of a piece of equipment.
 721 The *Top Level* Component elements provided in an MTConnectDevices docu-
 722 ment **SHOULD** be restricted to those defined in *Table 24*. However, these *Top Level*
 723 Component elements **MAY** also be used as *Lower Level* Component elements;
 724 as required.
- 725 • *Lower Level* Component elements are used to describe the sub-parts of the par-
 726 ent Component to provide more clarity and granularity to the physical or logical
 727 structure of the *Top Level* Component elements.

728 This section of the *Devices Information Model* provides guidance for the most common re-
 729 lationships between *Top Level* Component elements and *Lower Level* child components.
 730 However, all Component elements **MAY** be used in any configuration, as required, to
 731 fully describe a piece of equipment.

732 As described in *Section 4 - Structural Elements for MTConnectDevices*, Component is
 733 an abstract type *Structural Element* within the *Devices Information Model* and will never
 734 appear directly in the MTConnectDevices XML document. As abstract type XML
 735 elements, Component will be replaced in the XML document by a specific Component
 736 type.

737 *Table 24* defines the *Top Level* Component elements available to describe a piece of
 738 equipment.

Table 24: Top Level Component Elements

| Top Level Component Element †† | Description |
|--------------------------------|--|
| Axes | An XML container used to organize the <i>Structural Elements</i> of a piece of equipment that perform linear or rotational motion. |
| Controller | An XML container used to organize information about an intelligent or computational function within a piece of equipment. |

| Continuation of Table 24 | |
|--------------------------------|---|
| Top Level Component Element †† | Description |
| Systems | An XML container used to organize information for <i>Lower Level</i> elements representing the major sub-systems that are permanently integrated into a piece of equipment. |
| Auxiliaries | An XML container used to organize information for <i>Lower Level</i> elements representing functional sub-systems that provide supplementary or extended capabilities for a piece of equipment, but they are not required for the basic operation of the equipment. |
| Resources | An XML container used to organize information for <i>Lower Level</i> elements representing types of items, materials, and personnel that support the operation of a piece of equipment or work to be performed at a location. Resources also represents materials or other items consumed or transformed by a piece of equipment for production of parts or other types of goods. |
| Interfaces | An XML container that organizes information used to coordinate actions and activities between pieces of equipment that communicate information between each other. |

739 Note: ††The following components have been relocated or redefined since they are
 740 not classified as restricted *Top Level* components:

- 741 - Power was **DEPRECATED** in MTConnect Version 1.1 and was replaced
 742 by the *Data Entity* called AVAILABILITY.
 743 - Door has been redefined as a *Lower Level* component of a parent Compo-
 744 nent element or as a Composition element.
 745 - Actuator, due to its uniqueness, has been redefined as a piece of equip-
 746 ment with the ability to be represented as a *Lower Level* component of a parent
 747 Component element or as a Composition element.
 748 - Sensor, due to its uniqueness, has been redefined as a piece of equipment
 749 with the ability to be represented as a *Lower Level* component of a parent Com-
 750 ponent element (See *Section 9 - Sensor* for further detail).
 751 - Stock has been redefined as a *Lower Level* component of the Resources
 752 *Top Level* Component element.

753 The common relationship between the *Top Level* Component elements and the *Lower*
 754 *Level* child Component elements are described below. It should be noted that as the MT-
 755 Connect Standard evolves, more Component types will be added to organize information
 756 for new types of equipment and/or new physical or logical sub-parts of equipment.

757 5.1 Axes

758 Axes is a *Top Level* Component element. It is a container that organizes information
 759 representing the *Structural Elements* that perform linear or rotational motion for a piece
 760 of equipment.

761 Axes organizes information for the individual physical axes into Component types of
 762 Linear and Rotary based on the type of motion performed by each axis. Axes **MUST**
 763 contain at least one Linear or one Rotary type axis.

764 *Figure 17* defines the relationship between the Axes container and the individual axis
 765 type *Structural Elements*.

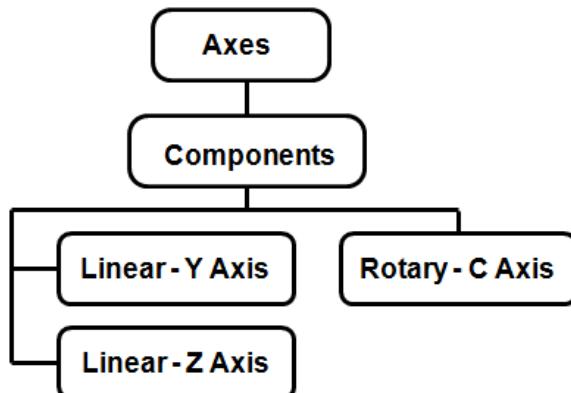


Figure 17: Axes Example with Two Linear Axes and One Rotary Axis

766 5.1.1 Linear

767 A Linear axis represents the movement of a physical piece of equipment, or a portion
 768 of the equipment, in a straight line.

769 Movement may be either in a positive or negative direction.

770 Linear type axes **MUST** be identified using a value for the name attribute as X, Y, or Z
 771 with numbers appended for additional axes in the same plane. Additional linear axes are

772 often referred to as U, V, and W. However, MTConnect defines the secondary axes to X,
773 Y, and Z as X2, Y2, and Z2.

774 If the piece of equipment is unable to provide information associated with the name at-
775 tribute, then the nativeName attribute **MUST** be included to identify the axis.

776 5.1.2 Rotary

777 A **Rotary** axis represents any non-linear or rotary movement of a physical piece of equip-
778 ment or a portion of the equipment.

779 **Rotary** type axes **MUST** be identified using a value for the name attribute as A, B, and
780 C for axes that rotate around the X, Y, and Z axes respectively. As with the **Linear** axes,
781 a number **MUST** be appended for additional axes in the same plane (C, C2, C3, C4, ...).

782 If the piece of equipment is unable to provide information associated with the name at-
783 tribute, then the nativeName attribute **MUST** be included to identify the axis.

784 An axis whose function is to provide rotary motion may function as a continuous rotation
785 (SPINDLE mode), continuous-path contour rotary motion (CONTOUR mode), or position-
786 ing (INDEX mode) to discrete rotary positions. As such, a **Rotary** type axis **SHOULD**
787 specify a ROTARY_MODE data item identifying the operating mode of the axis: SPINDLE,
788 INDEX, or CONTOUR.

789 5.1.2.1 Chuck

790 Chuck is an XML container that provides the information about a mechanism that holds a
791 part or stock material in place. It may also represent the information about any other type
792 mechanism that holds items in place within a piece of equipment.

793 The operation of a Chuck when represented as a Component element is defined by
794 CHUCK_STATE. The value of CHUCK_STATE **MUST** be OPEN, CLOSED, or UNLATCHED.

795 Chuck may be used in the MTConnectDevices document as either a *Lower Level*
796 component or as a Composition element of a parent Component element.

797 5.2 Controller

798 Controller is a *Top Level* container that organizes information for an intelligent part
799 of a piece of equipment that monitors and calculates information to alter the operating

800 conditions of the equipment. Typical types of controllers for a piece of equipment include
801 CNC (Computer Numerical Control), PAC (Programmable Automation Control), IPC (In-
802 dustrialized Computer), or IC (Imbedded Computer).

803 Controller is a component that organizes and provides information regarding the exe-
804 cution of a control program(s), the mode of operation of the piece of equipment, and fault
805 information regarding the operation of the equipment.

806 Note: MTConnect Version 1.1.0 and later implementations **SHOULD** use a *Lower*
807 *Level* Component element called Path to represent an individual tool path or
808 other independent function within a Controller element. When the Con-
809 troller element is capable of executing more than one simultaneous and in-
810 dependent programs, the implementation **MUST** specify a *Lower Level* Path
811 element representing each of the independent functions of the Controller.

812 5.2.1 Path

813 Path is an XML container that represents the information for an independent operation
814 or function within a Controller. For many types of equipment, Path represents a set
815 of Axes, one or more Program elements, and the data associated with the motion of a
816 control point as it moves through space. However, it **MAY** also represent any independent
817 function within a Controller that has unique data associated with that function.

818 Path **SHOULD** provide an EXECUTION data item to define the operational state of the
819 Controller component of the piece of equipment.

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

823 5.3 Systems

824 Systems is a *Top Level* XML container that provides structure for the information de-
825 scribing one or more *Lower Level* functional systems that perform as discrete operating
826 modules of the equipment or provide utility type services to support the operation of the
827 equipment. These systems are required for the piece of equipment to perform its intended
828 function and are permanently integrated into the piece of equipment.

829 Since these systems operate as separate functional units, they are represented in the MT-
830 ConnectDevices XML document as individual *Lower Level* Component elements

831 of Systems based on the function or service provided.

832 **5.3.1 Hydraulic System**

833 Hydraulic is an XML container that represents the information for a system comprised
834 of all the parts involved in moving and distributing pressurized liquid throughout the piece
835 of equipment.

836 **5.3.2 Pneumatic System**

837 Pneumatic is an XML container that represents the information for a system comprised
838 of all the parts involved in moving and distributing pressurized gas throughout the piece
839 of equipment.

840 **5.3.3 Coolant System**

841 Coolant is an XML container that represents the information for a system comprised
842 of all the parts involved in distribution and management of fluids that remove heat from a
843 piece of equipment.

844 **5.3.4 Lubrication System**

845 Lubrication is an XML container that represents the information for a system com-
846 prised of all the parts involved in distribution and management of fluids used to lubricate
847 portions of the piece of equipment.

848 **5.3.5 Electric System**

849 Electric is an XML container that represents the information for the main power sup-
850 ply for device piece of equipment and the distribution of that power throughout the equip-
851 ment. The electric system will provide all the data with regard to electric current, voltage,
852 frequency, etc. that applies to the piece of equipment as a functional unit. Data regarding
853 electric power that is specific to a Component will be reported as *Data Entities* for that
854 specific Component.

855 **5.3.6 Enclosure System**

856 Enclosure is an XML container that represents the information for a structure used to
857 contain or isolate a piece of equipment or area. The Enclosure system may provide
858 information regarding access to the internal components of a piece of equipment or the
859 conditions within the enclosure. For example, Door may be defined as a *Lower Level*
860 Component or Composition element of the Enclosure system.

861 **5.3.7 Protective System**

862 Protective is an XML container that represents the information for those functions
863 that detect or prevent harm or damage to equipment or personnel. Protective does not
864 include the information relating to the Enclosure system.

865 **5.3.8 ProcessPower System**

866 ProcessPower is an XML container that represents the information for a power source
867 associated with a piece of equipment that supplies energy to the manufacturing process
868 separate from the Electric system. For example, this could be the power source for an
869 EDM machining process, an electroplating line, or a welding system.

870 **5.3.9 Feeder System**

871 Feeder is an XML container that represents the information for a system that manages
872 the delivery of materials within a piece of equipment. For example, this could describe
873 the wire delivery system for an EDM or welding process; conveying system or pump and
874 valve system distributing material to a blending station; or a fuel delivery system feeding
875 a furnace.

876 **5.3.10 Dielectric System**

877 Dielectric is an XML container that represents the information for a system that man-
878 ages a chemical mixture used in a manufacturing process being performed at that piece of
879 equipment. For example, this could describe the dielectric system for an EDM process or
880 the chemical bath used in a plating process.

881 **5.3.11 EndEffector System**

882 `EndEffector` is an XML container that represents the information for those functions
883 that form the last link segment of a piece of equipment. It is the part of a piece of equipment
884 that interacts with the manufacturing process.

885 **5.4 Auxiliaries**

886 `Auxiliaries` is a *Top Level* XML container that provides structure for the information
887 describing one or more *Lower Level* functional systems that provide supplementary or
888 additional capabilities for the operation of a piece of equipment. These systems extend the
889 capabilities of a piece of equipment, but are not required for the equipment to function.

890 Since these systems operate as independent units or are only temporarily associated with a
891 piece of equipment, they are represented in the `MTConnectDevices` XML document as
892 individual *Lower Level* Component elements of `Auxiliaries` based on the function
893 or service provided to the equipment.

894 **5.4.1 Loader System**

895 `Loader` is an XML container that represents the information for a unit comprised of all
896 the parts involved in moving and distributing materials, parts, tooling, and other items to
897 or from a piece of equipment.

898 **5.4.2 WasteDisposal System**

899 `WasteDisposal` is an XML container that represents the information for a unit com-
900 prised of all the parts involved in removing manufacturing byproducts from a piece of
901 equipment.

902 **5.4.3 ToolingDelivery System**

903 `ToolingDelivery` is an XML container that represents the information for a unit in-
904 volved in managing, positioning, storing, and delivering tooling within a piece of equip-
905 ment.

906 5.4.4 BarFeeder System

907 BarFeeder is an XML container that represents the information for a unit involved in
908 delivering bar stock to a piece of equipment.

909 5.4.5 Environmental System

910 Environmental is an XML container that represents the information for a unit or func-
911 tion involved in monitoring, managing, or conditioning the environment around or within
912 a piece of equipment.

913 5.4.6 Sensor System

914 Sensor is a XML container that represents the information for a piece of equipment that
915 responds to a physical stimulus and transmits a resulting impulse or value from a sensing
916 unit. When modeled as a component of Auxiliaries, sensor **SHOULD** represent an
917 integrated *sensor unit* system that provides signal processing, conversion, and communi-
918 cations. A *sensor unit* may have multiple *sensing elements*; each representing the data for
919 a variety of measured values. See *Section 9.2 - Sensor Unit* for more details on *sensor*
920 *unit*.

921 Note: If modeling an individual sensor, then sensor should be associated with the
922 component that the measured value is most closely associated. See *Section 5.7.3*
923 - *Sensor*.

924 5.4.7 Deposition System

925 Deposition is an XML container that represents the information for a system that man-
926 ages the addition of material or state change of material being performed in an additive
927 manufacturing process. For example, this could describe the portion of a piece of equip-
928 ment that manages a material extrusion process or a vat polymerization process.

929 5.5 Resources

930 Resources is a *Top Level* XML container that groups items that support the operation
931 of a piece of equipment. Resources also represents materials or other items consumed,

932 transformed, or used for production of parts, materials, or other types of goods by a piece
933 of equipment.

934 **5.5.1 Materials**

935 **Materials** is an XML container that provides information about materials or other items
936 consumed or used by the piece of equipment for production of parts, materials, or other
937 types of goods. **Materials** also represents parts or part stock that are present at a piece
938 of equipment or location to which work is applied to transform the part or stock material
939 into a more finished state.

940 **5.5.1.1 Stock**

941 **Stock** is an XML container that represents the information for the material that is used in
942 a manufacturing process and to which work is applied in a machine or piece of equipment
943 to produce parts.

944 **Stock** may be either a continuous piece of material from which multiple parts may be
945 produced or it may be a discrete piece of material that will be made into a part or a set of
946 parts.

947 **5.6 Interfaces**

948 **Interfaces** is a *Top Level XML Structural Element* in the MTConnectDevices
949 XML document. **Interfaces** organizes the information provided by a piece of equip-
950 ment used to coordinate activities with other pieces of equipment. As such, **Interfaces**
951 represents the inter-device communication information between a piece of equipment and
952 other pieces of equipment.

953 See *MTConnect Standard: Part 5.0 - Interfaces* for detailed information on **Inter-**
954 **faces**.

955 **5.7 Other Components**

956 While most component elements **SHOULD** be modeled in a specific manner, there are
957 some types of component elements that are used ubiquitously in equipment and **MAY** be
958 associated with any number of different types of parent component elements.

959 These components **MAY** be modeled as *Lower Level* components of the Parent Element.

960 **5.7.1 Actuator**

961 **Actuator** is an XML container that represents the information for an apparatus for mov-
962 ing or controlling a mechanism or system. It takes energy usually provided by air, electric
963 current, or liquid and converts the energy into some kind of motion.

964 **5.7.2 Door**

965 **Door** is an XML container that represents the information for a mechanical mechanism or
966 closure that can cover, for example, a physical access portal into a piece of equipment. The
967 closure can be opened or closed to allow or restrict access to other parts of the equipment.

968 When **Door** is represented as a Component, it **MUST** have a data item called DOOR_-
969 STATE to indicate if the door is OPEN, CLOSED, or UNLATCHED. A Component **MAY**
970 contain multiple **Door** components.

971 **5.7.3 Sensor**

972 Sensor is a XML container that represents the information for a piece of equipment that
973 responds to a physical stimulus and transmits a resulting impulse or value. If modeling
974 individual sensors, then sensor should be associated with the component that the measured
975 value is most closely associated.

976 See *Section 9 - Sensor* for more details on the use of Sensor.

977 6 Composition Type Structural Elements

978 Composition *Structural Elements* are used to describe the lowest level physical building
 979 blocks of a piece of equipment contained within a Component. By referencing a specific
 980 Composition element, further clarification and meaning to data associated with a
 981 specific Component can be achieved.

982 Both Component and Composition elements are *Lower Level* child Component
 983 XML elements representing the sub-parts of the parent Component. However, there are
 984 distinct differences between Component and Composition type elements.

985 Component elements may be further defined with *Lower Level* Component elements
 986 and may have associated *Data Entities*.

987 Composition elements represent the lowest level physical part of a piece of equipment.
 988 They **MUST NOT** be further defined with *Lower Level* Component elements and they
 989 **MUST NOT** have *Data Entities* directly associated with them. They do provide additional
 990 information that can be used to enhance the specificity of *Data Entities* associated with the
 991 parent Component.

992 *Table 25* defines Composition type elements that are currently available to describe
 993 sub-parts of a Component element.

Table 25: Composition type Elements

| Element Type | Description |
|--------------|--|
| ACTUATOR | A mechanism for moving or controlling a mechanical part of a piece of equipment. It takes energy usually provided by air, electric current, or liquid and converts the energy into some kind of motion. |
| AMPLIFIER | An electronic component or circuit for amplifying power, electric current, or voltage. |
| BALLSCREW | A mechanical structure for transforming rotary motion into linear motion. |
| BELT | An endless flexible band used to transmit motion for a piece of equipment or to convey materials and objects. |

| Continuation of Table 25 | |
|--------------------------|---|
| Element Type | Description |
| BRAKE | A mechanism for slowing or stopping a moving object by the absorption or transfer of the energy of momentum, usually by means of friction, electrical force, or magnetic force. |
| CHAIN | An interconnected series of objects that band together and are used to transmit motion for a piece of equipment or to convey materials and objects. |
| CHOPPER | A mechanism used to break material into smaller pieces. |
| CHUCK | A mechanism that holds a part, stock material, or any other item in place. |
| CHUTE | An inclined channel for conveying material. |
| CIRCUIT_BREAKER | A mechanism for interrupting an electric circuit. |
| CLAMP | A mechanism used to strengthen, support, or fasten objects in place. |
| COMPRESSOR | A pump or other mechanism for reducing volume and increasing pressure of gases in order to condense the gases to drive pneumatically powered pieces of equipment. |
| DOOR | A mechanical mechanism or closure that can cover a physical access portal into a piece of equipment allowing or restricting access to other parts of the equipment. |
| DRAIN | A mechanism that allows material to flow for the purpose of drainage from, for example, a vessel or tank. |
| ENCODER | A mechanism used to measure rotary position. |
| EXPOSURE_UNIT | A mechanism for emitting a type of radiation |
| EXTRUSION_UNIT | A mechanism for dispensing liquid or powdered materials |
| FAN | Any mechanism for producing a current of air. |

| Continuation of Table 25 | |
|--------------------------|---|
| Element Type | Description |
| FILTER | Any substance or structure through which liquids or gases are passed to remove suspended impurities or to recover solids. |
| GALVANOMOTOR | An electromechanical actuator that produces deflection of a beam of light or energy in response to electric current through its coil in a magnetic field. |
| GRIPPER | A mechanism that holds a part, stock material, or any other item in place. |
| HOPPER | A chamber or bin in which materials are stored temporarily, typically being filled through the top and dispensed through the bottom. |
| LINEAR_POSITION_FEEDBACK | A mechanism that measures linear motion or position. |
| MOTOR | A mechanism that converts electrical, pneumatic, or hydraulic energy into mechanical energy. |
| OIL | A viscous liquid. |
| POWER_SUPPLY | A unit that provides power to electric mechanisms. |
| PULLEY | A mechanism or wheel that turns in a frame or block and serves to change the direction of or to transmit force. |
| PUMP | An apparatus raising, driving, exhausting, or compressing fluids or gases by means of a piston, plunger, or set of rotating vanes. |
| REEL | A rotary storage unit for material |
| SENSING_ELEMENT | A mechanism that provides a signal or measured value. |
| SPREADER | A mechanism for flattening or spreading materials |

| Continuation of Table 25 | |
|--------------------------|--|
| Element Type | Description |
| STORAGE_BATTERY | A component consisting of one or more cells, in which chemical energy is converted into electricity and used as a source of power. |
| SWITCH | A mechanism for turning on or off an electric current or for making or breaking a circuit. |
| TABLE | A surface for holding an object or material |
| TANK | A receptacle or container for holding material. |
| TENSIONER | A mechanism that provides or applies a stretch or strain to another mechanism. |
| TRANSFORMER | A mechanism that transforms electric energy from a source to a secondary circuit. |
| VALVE | Any mechanism for halting or controlling the flow of a liquid, gas, or other material through a passage, pipe, inlet, or outlet. |
| VAT | A container for liquid or powdered materials |
| WATER | A fluid. |
| WIRE | A string like piece or filament of relatively rigid or flexible material provided in a variety of diameters. |

994 Note: As the MTConnect Standard evolves, more Composition types will be
 995 added.

996 7 Data Entities for Device

- 997 In the MTConnectDevices XML document, *Data Entities* are XML elements that de-
 998 scribe data that can be reported by a piece of equipment and are associated with *Device*
 999 and *Component Structural Elements*. While the *Data Entities* describe the data that can
 1000 be reported by a piece of equipment in the MTConnectDevices document, the actual
 1001 data values are provided in the *Streams Information Model*. See *MTConnect Standard:*
 1002 *Part 3.0 - Streams Information Model* for detail on the reported values.
- 1003 Each *Data Entity* **SHOULD** be modeled in the MTConnectDevices document such
 1004 that it is associated with the *Structural Element* that the reported data directly applies.
- 1005 When *Data Entities* are associated with a *Structural Element*, they are organized in a
 1006 *DataItems* XML element. *DataItems* is a container type XML element. *DataItems*
 1007 provides the structure for organizing individual *DataItem* elements that represent each
 1008 *Data Entity*. The *DataItems* container is comprised of one or more *DataItem* type
 1009 XML element(s).
- 1010 *DataItem* describes specific types of *Data Entities* that represent a numeric value, a
 1011 functioning state, or a health status reported by a piece of equipment. *DataItem* provides
 1012 a detailed description for each *Data Entity* that is reported; it defines the type of data being
 1013 reported and an array of optional attributes that further describe that data. The different
 1014 types of *DataItem* elements are defined in *Section 8 - Listing of Data Items*.
- 1015 *Figure 18* demonstrates the relationship between *Data Entities* (*DataItem*) and the var-
 1016 ious *Structural Elements* in the MTConnectDevices XML document.

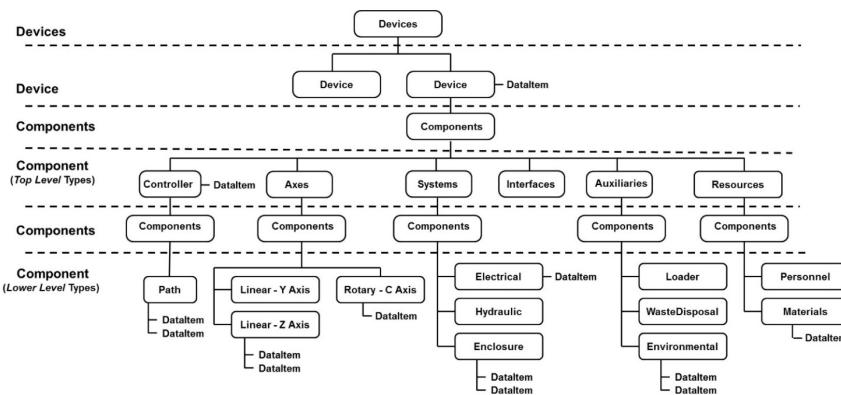


Figure 18: Example Data Entities for Device (DataItem)

1017 7.1 DataItems

1018 The DataItems XML element is the first, or highest, level container for the *Data Entities*
 1019 associated with a Device or Component XML element. DataItems **MUST** contain
 1020 only DataItem type elements. DataItems **MUST** contain at least one DataItem
 1021 type element, but **MAY** contain multiple DataItem type elements.

Table 26: MTConnect DataItems Element

| Element | Description | Occurrence |
|-----------|---|------------|
| DataItems | An XML container consisting of one or more types of DataItem XML elements. Only one DataItems container MUST appear for each <i>Structural Element</i> in the XML document. | 0..1 |

1022 7.2 DataItem

1023 A DataItem XML element represents each *Data Entity* that **MAY** be reported by a piece
 1024 of equipment through an *Agent*. DataItem provides a detailed description for each *Data*
 1025 *Entity* that is reported and defines the type of data being reported along with an array of
 1026 optional attributes that further define that data. XML elements representing DataItem
 1027 will include elements such as TEMPERATURE, PRESSURE, and VELOCITY.

Table 27: MTConnect DataItem Element

| Element | Description | Occurrence |
|----------|---|------------|
| DataItem | <i>Data Entity</i> describing a piece of information reported about a piece of equipment. | 1..* |

1028 7.2.1 XML Schema Structure for DataItem

1029 *Figure 19* represents the structure of a DataItem XML element showing the attributes
 1030 defined for DataItem and the elements that may be associated with DataItem type
 1031 XML elements.

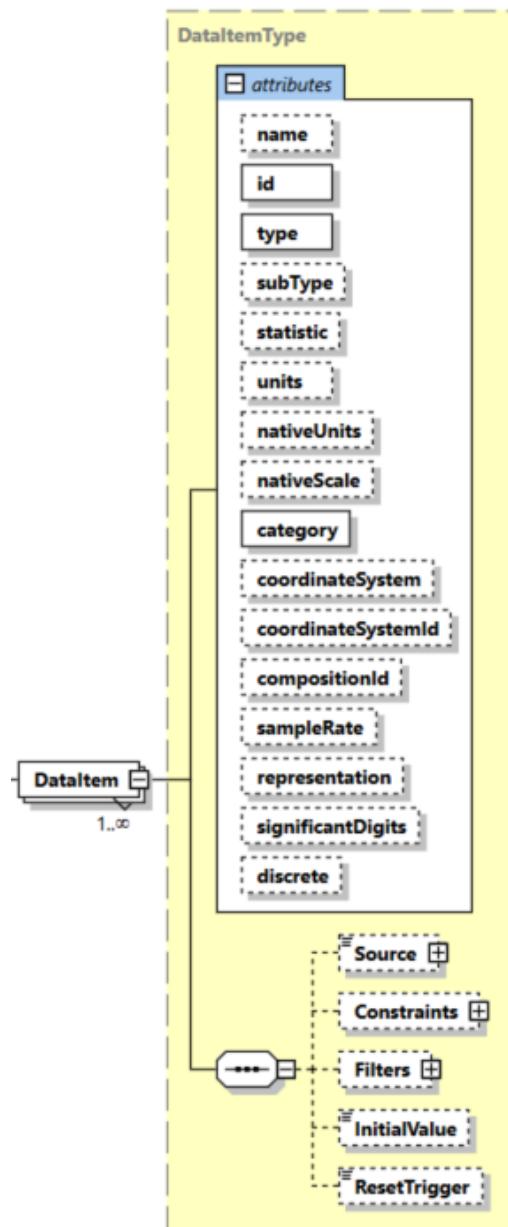


Figure 19: DataItem Diagram

1032 7.2.2 Attributes for DataItem

1033 *Table 28* lists the attributes defined to provide information for a DataItem type XML
 1034 element.

1035 DataItem **MUST** specify the type of data being reported, the id of the DataItem, and
 1036 the category of the DataItem.

Table 28: Attributes for DataItem

| Attribute | Description | Occurrence |
|-----------|---|------------|
| name | <p>The name of the data item.</p> <p>name is provided as an additional human readable identifier for this data item in addition to the id.</p> <p>name is an optional attribute and will be implementation dependent.</p> <p>An NMTOKEN XML type.</p> | 0..1 |
| id | <p>The unique identifier for this element.</p> <p>id is a required attribute.</p> <p>The id attribute MUST be unique within the MTConnectDevices document.</p> <p>An XML ID-type.</p> | 1 |
| type | <p>The type of data being measured.</p> <p>type is a required attribute.</p> <p>Examples of types are POSITION, VELOCITY, ANGLE, BLOCK, and ROTARY_VELOCITY.</p> | 1 |
| subType | <p>A sub-categorization of the data item type.</p> <p>subType is an optional attribute.</p> <p>For example, the subType of POSITION can be ACTUAL or COMMANDED.</p> <p>Not all type attributes have a subType.</p> | 0..1 |

| Continuation of Table 28 | | |
|--------------------------|--|------------|
| Attribute | Description | Occurrence |
| statistic | <p>Describes the type of statistical calculation performed on a series of data samples to provide the reported data value.</p> <p>statistic is an optional attribute.</p> <p>Examples of statistic are AVERAGE, MINIMUM, MAXIMUM, ROOT_MEAN_SQUARE, RANGE, MEDIAN, MODE, and STANDARD_DEVIATION.</p> | 0..1 |
| units | <p>The unit of measurement for the reported value of the data item.</p> <p>units is an optional attribute.</p> <p>Data items in the Sample category MUST report the standard units for the measured values.</p> <p>See <i>Section 7.2.2.5 - units Attribute for DataItem</i> for a list of available standard units identified in the MTConnect Standard.</p> | 0..1 |
| nativeUnits | <p>The native units of measurement for the reported value of the data item.</p> <p>nativeUnits is an optional attribute.</p> <p>See <i>Section 7.2.2.6 - nativeUnits Attribute for DataItem</i> for a list of available native units identified in the MTConnect Standard.</p> | 0..1 |

| Continuation of Table 28 | | |
|--------------------------|---|------------|
| Attribute | Description | Occurrence |
| nativeScale | <p>The nativeUnits may not be scaled to directly represent the original measured value. nativeScale MAY be used to convert the reported value to represent the original measured value.</p> <p>nativeScale is an optional attribute.</p> <p>As an example, the nativeUnits may be reported as GALLON/MINUTE. The measured value may actually be in 1000 GALLON/MINUTE. The value of the reported data MAY be divided by the nativeScale to convert the reported value to its original measured value and units.</p> <p>If provided, the value MUST be numeric.</p> | 0..1 |
| category | <p>Specifies the kind of information provided by a data item.</p> <p>category is a required attribute.</p> <p>The available options are Sample, Event, or Condition.</p> | 1 |
| coordinateSystem | <p>For measured values relative to a coordinate system like POSITION, the coordinate system being used may be reported.</p> <p>coordinateSystem is an optional attribute.</p> <p>The available values for coordinateSystem are WORK and MACHINE.</p> | 0..1 |
| compositionId | <p>The identifier attribute of the Composition element that the reported data is most closely associated.</p> <p>compositionId is an optional attribute.</p> | 0..1 |

| Continuation of Table 28 | | |
|--------------------------|--|------------|
| Attribute | Description | Occurrence |
| sampleRate | <p>The rate at which successive samples of a data item are recorded by a piece of equipment.</p> <p>sampleRate is an optional attribute.</p> <p>sampleRate is expressed in terms of samples per second.</p> <p>If the sampleRate is smaller than one, the number can be represented as a floating point number.</p> <p>For example, a rate 1 per 10 seconds would be 0.1</p> | 0..1 |
| representation | <p>Description of a means to interpret data consisting of multiple data points or as a single value.</p> <p>representation is an optional attribute.</p> <p>representation defines the unique format for each set of data.</p> <p>representation for TIME_SERIES, DISCRETE (DEPRECATED in <i>Version 1.5</i>), DATA_SET, and VALUE are defined in <i>Section 7.2.2.12 - representation Attribute for DataItem</i>.</p> <p>If representation is not specified, it MUST be determined to be VALUE.</p> | 0..1 |
| significantDigits | <p>The number of significant digits in the reported value.</p> <p>significantDigits is an optional attribute.</p> <p>This SHOULD be specified for all numeric values.</p> | 0..1 |

| Continuation of Table 28 | | |
|--------------------------|--|------------|
| Attribute | Description | Occurrence |
| discrete | <p>An indication signifying whether each value reported for the <i>Data Entity</i> is significant and whether duplicate values are to be suppressed.</p> <p>The value defined MUST be either <code>true</code> or <code>false</code> - an XML boolean type.</p> <p><code>true</code> indicates that each update to the <i>Data Entity</i>'s value is significant and duplicate values MUST NOT be suppressed.</p> <p><code>false</code> indicates that duplicated values MUST be suppressed.</p> <p>If a value is not defined for <code>discrete</code>, the default value MUST be <code>false</code>.</p> | 0..1 |

1037 **7.2.2.1 name Attribute for DataItem**

1038 The attribute `name` is provided as an additional human readable identifier for a data item.
 1039 It is not required and is implementation dependent.

1040 **7.2.2.2 id Attribute for DataItem**

1041 Each `DataItem` element **MUST** be identified with an `id`. The `id` attribute **MUST** be
 1042 unique across the entire `MTConnectDevices` document for a piece of equipment, in-
 1043 cluding the identifiers for all *Structural Elements*. This unique `id` provides the information
 1044 required by a client software application to uniquely identify each *Data Entity*.

1045 For example, an XML document may provide three different *Data Entities* representing
 1046 the position of the axes on a machine (x axis position, y axis position, and z axis position).
 1047 All three may be modeled in the XML document as `POSITION` type data items for the
 1048 `Axes` components. The unique `id` allows the client software application to distinguish
 1049 the data for each of the axes.

1050 **7.2.2.3 type and subType Attributes for DataItem**

1051 The attribute `type` specifies the kind of data that is represented by the data item.

1052 The attribute `type` **MUST** be specified for every data item.

1053 A data item **MAY** further qualify the data being reported by specifying a `subType`.
 1054 `subType` is required for certain data item types. For example, `POSITION` has the
 1055 `subType` of `ACTUAL` and `PROGRAMMED`. Both data values can be represented in the
 1056 document as two separate and different `DataItem` XML elements – `POSITION` with
 1057 `subType ACTUAL` and `POSITION` with `subType PROGRAMMED`.

1058 The `type` and `subType` **SHOULD** be used to further identify the meaning of the `DataItem`
 1059 associated with a `Component` element when a `subType` is applicable. There **SHOULD**
 1060 **NOT** be more than one `DataItem` with the same `type`, `subType`, and `composi-`
 1061 `tionId` within a `Component` element.

1062 *Section 8 - Listing of Data Items* provides a detailed listing of the data item `type` and
 1063 `subType` elements defined for each category of data item available for a piece of
 1064 equipment: `SAMPLE`, `EVENT`, and `CONDITION`.

1065 **7.2.2.4 statistic Attribute for DataItem**

1066 A piece of equipment may further process some data types using a statistical calculation
 1067 like average, mean, or square root. In this case, the `statistic` attribute **MAY** be used
 1068 to indicate how the data was processed.

1069 `statistic` may be defined for any `SAMPLE` type `DataItem`. All statistic data is re-
 1070 ported in the standard units of the `DataItem`.

1071 `statistic` data is always the result of a calculation using data that has been measured
 1072 over a specified period of time.

1073 The value of `statistic` may be periodically reset. When a piece of equipment reports
 1074 a `DataItem` with a value that is a `statistic`, the information provided in the XML
 1075 document for that *Data Entity* **MUST** include an additional attribute called `duration`.
 1076 The attribute `duration` defines the period of time over which the `statistic` has been
 1077 calculated. See *MTConnect Standard: Part 3.0 - Streams Information Model* for more
 1078 information about `duration`.

1079 *Table 29* shows the `statistic` calculations that can be defined for a `DataItem`.

Table 29: DataItem attribute statistic type

| Statistic | Description |
|--------------------|--|
| AVERAGE | Mathematical Average value calculated for the data item 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 data item during the calculation period. |
| MEDIAN | The middle number of a series of numbers. |
| MINIMUM | Minimum value recorded for the data item during the calculation period. |
| MODE | The number in a series of numbers that occurs most often. |
| RANGE | Difference between the maximum and minimum value of a data item during the calculation period. Also represents Peak-to-Peak measurement in a waveform. |
| ROOT_MEAN_SQUARE | Mathematical Root Mean Square (RMS) value calculated for the data item during the calculation period. |
| STANDARD_DEVIATION | Statistical Standard Deviation value calculated for the data item during the calculation period. |

1080 **7.2.2.5 units Attribute for DataItem**

1081 *Table 30* lists the units that are defined as the standard unit of measure for each type of
 1082 DataItem. All SAMPLE type data items **MUST** report data values in standard units.

Table 30: DataItem attribute units type

| Units | Description |
|--------------------------------------|--|
| AMPERE | Amps |
| CELSIUS | Degrees Celsius |
| COUNT | A count of something. |
| CUBIC_MILLIMETER | Geometric volume in millimeters |
| CUBIC_MILLIMETER/SECOND | Change of geometric volume per second |
| CUBIC_MILLIMETER/SECOND ² | Change in geometric volume per second squared |
| 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 | Measurement of volume of a fluid |
| LITER/SECOND | Liters per second |
| MICRO_RADIAN | Measurement of Tilt |
| MILLIGRAM | Milligram |
| MILLIGRAM/CUBIC_MILLIMETER | Milligram per cubic millimeter |
| MILLILITER | Milliliter |
| MILLIMETER | Millimeters |
| MILLIMETER/REVOLUTION | Millimeters per revolution. |
| MILLIMETER/SECOND | Millimeters per second |

| Continuation of Table 30 | |
|--------------------------------|---|
| Units | Description |
| 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 |
| VOLT_AMPERE | Volt-Ampere (VA) |
| VOLT_AMPERE_REACTIVE | Volt-Ampere Reactive (VAR) |
| WATT | Watts |
| WATT_SECOND | Measurement of electrical energy, equal to one Joule |

1083 **7.2.2.6 nativeUnits Attribute for DataItem**

1084 The `nativeUnits` attribute provides additional information about the original measured
 1085 value for a *Data Entity* reported by a piece of equipment. `nativeUnits` **MAY** be spec-
 1086 ified to provide additional information about the data if the units of the measured value
 1087 supplied by the piece of equipment differ from the value provided for that data when con-
 1088 verted to standard units.

1089 Table 31 defines the nativeUnits currently supported by the MTConnectDevices
 1090 XML document:

Table 31: DataItem attribute nativeunits type

| 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. |
| HOUR | A measurement of time in hours |
| 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. |
| LITER | Measurement of volume of a fluid |
| LITER/MINUTE | Measurement of rate of flow of a fluid |
| MILLIMETER/MINUTE | Velocity in millimeters per minute |

| Continuation of Table 31 | |
|----------------------------|--|
| Native Units | Description |
| MINUTE | A measurement of time in minutes |
| OTHER | Unsupported units |
| POUND | US pounds |
| POUND / INCH ² | Pressure in pounds per square inch (PSI). |
| RADIAN | Angle in radians |
| RADIAN/MINUTE | Velocity in radians per minute. |
| RADIAN/SECOND | Rotational acceleration in radian per second squared |
| RADIAN/SECOND ² | Rotational acceleration in radian per second squared |
| REVOLUTION/SECOND | Rotational velocity in revolution per second |

1091 **7.2.2.7 nativeScale Attribute for DataItem**

1092 The units of measure for some measured values may be different from the nativeUnits
 1093 defined in *Section 7.2.2.8 - category Attribute for DataItem*. In the cases where the units
 1094 of measure use a different weighting or range than is provided by nativeUnits, the
 1095 nativeScale attribute can be used to define the original units of measure.

1096 As an example, a velocity measured in units of 100 ft/min can be represented as native-
 1097 Units="FEET/MINUTE" and nativeScale="100".

1098 **7.2.2.8 category Attribute for DataItem**

1099 Many DataItem types provide two forms of data, a value (reported as either a SAMPLE
 1100 or EVENT category) and a health status (reported as a CONDITION category). Therefore,
 1101 each occurrence of a DataItem in the XML document **MUST** report a category at-
 1102 tribute. This category attribute provides the information required by a client software
 1103 application to determine the specific meaning of the data provided.

1104 Each *Data Entity* provided by a piece of equipment **MUST** be identified with one of the
1105 following: SAMPLE, EVENT, CONDITION.

1106 A SAMPLE is the reading of the value of a continuously variable or analog data value. A
1107 continuous value can be measured at any point-in-time and will always produce a result.
1108 An example of a continuous data value is the position of a linear axis called X.

1109 The data provided for a SAMPLE category data item is always a floating point number
1110 or integers that have an infinite number of possible values. This is different from a state
1111 or discrete type data item that has a limited number of possible values. A data item of
1112 category SAMPLE **MUST** also provide the units attribute.

1113 An EVENT is a data item representing a discrete piece of information from the piece of
1114 equipment. EVENT does not have intermediate values that vary over time, as does SAM-
1115 PLE. An EVENT is information that, when provided at any specific point in time, repre-
1116 sents the current state of the piece of equipment.

1117 There are two types of EVENT: those representing state, with two or more discrete values,
1118 and those representing messages that contain plain text data.

1119 An example of a state type EVENT is the value of the data item DOOR_STATE, which
1120 can be OPEN, CLOSED, or UNLATCHED. (Note: No other values are valid to represent the
1121 value of DOOR_STATE.)

1122 An example of a message type EVENT is the value for a data item PROGRAM. The value
1123 representing PROGRAM can be any valid string of characters.

1124 A CONDITION is a data item that communicates information about the health of a piece
1125 of equipment and its ability to function. A valid value for a data item in the category
1126 CONDITION can be one of Normal, Warning, or Fault.

1127 A data item of category CONDITION **MAY** report multiple values (CONDITION) at one
1128 time whereas a data item of category SAMPLE or EVENT can only have a single value at
1129 any one point in time.

1130 **7.2.2.9 coordinateSystem Attribute for DataItem**

1131 The values reported by a piece of equipment for some types of data will be associated
 1132 to a specific positioning measurement system used by the equipment. The coordinateSystem
 1133 attribute **MAY** be used to specify the coordinate system used for the measured value.

1135 The coordinateSystem attribute is used by a client software application to interpret
 1136 the spatial relationship between values reported by a piece of equipment.

1137 If coordinateSystem is not provided, all values representing positional data for Axes
 1138 **MUST** be interpreted using the MACHINE coordinate system and all values representing
 1139 positional data for Path **MUST** be interpreted using the WORK coordinate system.

1140 *Table 32* defines the types of coordinateSystem currently supported by the MTConnectDevices XML document:

Table 32: DataItem attribute coordinateSystem type

| 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 piece of equipment, the MACHINE coordinates will be used. |

1142 **7.2.2.10 compositionId Attribute for DataItem**

1143 compositionId attribute identifies the id of the Composition element where the
 1144 reported data is most closely associated.

1145 An example would be a TEMPERATURE associated with a Linear type axis may be
 1146 further clarified by referencing the MOTOR or AMPLIFIER type Composition element
 1147 associated with that axis, which differentiates the temperature of the motor from the tem-
 1148 perature of the amplifier.

1149 The `compositionId` attribute provides the information required by a client software
 1150 application to interpret the data with a greater specificity and to disambiguate between
 1151 multiple *Data Entities* of the same data type associated with a `Component` element.

1152 **7.2.2.11 sampleRate Attribute for DataItem**

1153 The value for some data types provided by a piece of equipment may be reported as a
 1154 single set of data containing a series of values that have been recorded at a fixed sample
 1155 rate. When such data is reported, the `sampleRate` defines the rate at which successive
 1156 samples of data were recorded.

1157 The `sampleRate` attribute provides the information required by a client software applica-
 1158 tion to interpret the data and the sampling time relationship between successive values
 1159 contained in the set of data.

1160 `sampleRate` is expressed in terms of samples per second. If the sample rate is smaller
 1161 than one, the number can be represented as a floating point number. For example, a rate 1
 1162 per 10 seconds would be 0.1

1163 **7.2.2.12 representation Attribute for DataItem**

1164 Some data types provide data that may consist of a series of values or a file of data, not a
 1165 single value. Other data types provide a series of data values that may require additional
 1166 information so that the data may be correctly understood by a client software application.

1167 When such data is provided, the `representation` attribute **MUST** be used to define
 1168 the format for the data provided.

1169 The types of representation defined are provided in *Table 33*.

1170 Note: See *MTConnect Standard: Part 3.0 - Streams Information Model* for more
 1171 information on the structure and format of each representation.

Table 33: DataItem attribute representation type

| Representation | Description |
|----------------|---|
| DATA_SET | The reported value(s) are represented as a set of <i>key-value pairs</i> . Each reported value in the <i>Data Set</i> MUST have a unique key. |

| Continuation of Table 33 | |
|--|---|
| Representation | Description |
| DISCRETE (DEPRECATED in Version 1.5) | <p>DEPRECATED as a representation in MTConnect Version. 1.5. Replaced by the discrete attribute for a <i>Data Entity</i> – <i>Section 7.2.2.14 - discrete Attribute for DataItem</i>.</p> <p>A <i>Data Entity</i> 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.</p> <p>In this case, duplicate occurrences of the same data value SHOULD NOT be suppressed.</p> <p>An example of a DISCRETE data type would be a parts counter that reports the completion of each part versus the accumulation of parts. Another example would be a Message that does not typically have a reset state and may re-occur each time a specific message is triggered.</p> |
| TIME_SERIES | <p>A series of sampled data.</p> <p>The data is reported for a specified number of samples and each sample is reported with a fixed period.</p> |
| VALUE | <p>The measured value of the sample data.</p> <p>If no representation is specified for a data item, the representation MUST be determined to be VALUE.</p> |

1172 **7.2.2.13 significantDigits Attribute for DataItem**

1173 `significantDigits` is used to specify the level of precision (number of significant
1174 digits) for the value provided for a data item.

1175 `significantDigits` attribute is not required for a data item, but it is recommended
1176 and **SHOULD** be used for any data item reporting a numeric value.

1177 **7.2.2.14 discrete Attribute for DataItem**

1178 An indication signifying whether each value reported for the *Data Entity* is significant and
1179 whether duplicate values are to be suppressed.

1180 The value defined **MUST** be either `true` or `false` - an XML boolean type.

1181 `true` indicates that each update to the *Data Entity*'s value is significant and duplicate
1182 values **MUST NOT** be suppressed.

1183 `false` indicates that duplicated values **MUST** be suppressed.

1184 If a value is not defined for `discrete`, the default value **MUST** be `false`.

1185 **7.2.3 Elements for DataItem**

1186 *Table 34* lists the elements defined to provide additional information for a `DataItem`
1187 type XML element.

Table 34: Elements for DataItem

| Element | Description | Occurrence |
|---------|--|------------|
| Source | <p>Source is an optional XML element that identifies the Component, DataItem, or Composition representing the area of the piece of equipment from which a measured value originates.</p> <p>Additionally, Source MAY provide information relating to the identity of a measured value. This information is reported as CDATA for Source. (example, a PLC tag)</p> | 0..1 |

| Continuation of Table 34 | | |
|--------------------------|---|------------|
| Element | Description | Occurrence |
| Constraints | Constraints is an optional container that provides a set of expected values that can be reported for this DataItem. Constraints are used by a software application to evaluate the validity of the reported data. | 0..1 |
| Filters | An optional container for the Filter elements associated with this DataItem element. | 0..1 |
| InitialValue | <p>InitialValue is an optional XML element that defines the starting value for a data item as well as the value to be set for the data item after a reset event.</p> <p>Only one InitialValue element may be defined for a data item. The value will be constant and cannot change.</p> <p>If no InitialValue element is defined for a data item that is periodically reset, then the starting value for the data item MUST be a value of 0.</p> | 0..1 |
| ResetTrigger | ResetTrigger is an optional XML element that identifies the type of event that may cause a reset to occur. It is additional information regarding the meaning of the data that establishes an understanding of the time frame that the data represents so that the data may be correctly understood by a client software application. | 0..1 |

1188 7.2.3.1 Source Element for DataItem

1189 Source is an optional XML element that may be used to identify the physical part of a
 1190 piece of equipment where the data represented by DataItem originated and/or it may be
 1191 used to identify a complex name or an alternate name used to identify the data where it
 1192 originated (e.g. a PLC tag name).

1193 As an example, data related to a servo motor on an Axes component may actually origi-
 1194 nate from a measurement made in the Controller element.

1195 In the case where the real name associated with a DataItem element is either complex

1196 or does not meet the format requirements of a NMTOKEN XML type, the real name of
 1197 the element may not be able to be expressed in the name attribute. Additionally, a second
 1198 or alternate name may be required to describe a piece of data. An example of this case
 1199 would be the identity of the bit address in a PLC that represents this piece of data (PLC
 1200 address I0015.4). When these cases occur, the alternate name can be provided as the value
 1201 for the CDATA for Source.

1202 The XML schema in *Figure 20* represents the structure of the Source XML element
 1203 showing the attributes defined for Source.

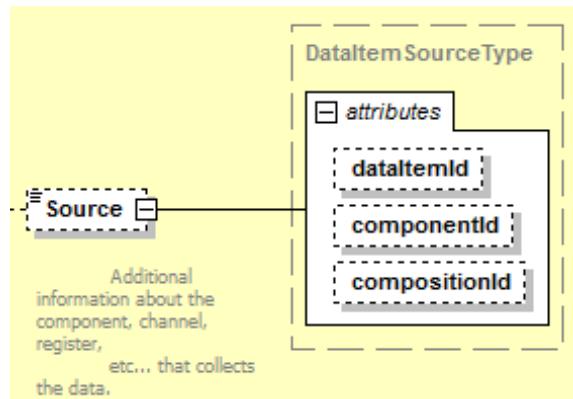


Figure 20: Source Diagram

1204 7.2.3.1.1 Attributes for Source

1205 *Table 35* identifies the attributes available to identify Source for a measured value:

Table 35: Attributes for Source

| Attribute | Description | Occurrence |
|-------------|---|------------|
| componentId | <p>The identifier attribute of the Component element that represents the physical part of a piece of equipment where the data represented by the DataItem element originated.</p> <p>A <i>Valid Data Value</i> reported for componentId MUST be the value of the id attribute for the Component element identified.</p> <p>componentId is an optional attribute.</p> | 0..1 |

| Continuation of Table 35 | | |
|--------------------------|---|------------|
| Attribute | Description | Occurrence |
| dataItemId | <p>The identifier attribute of the DataItem that represents the originally measured value of the data referenced by this data item.</p> <p>A <i>Valid Data Value</i> reported for dataItemId MUST be the value of the id attribute for the DataItem element identified.</p> <p>dataItemId is an optional attribute.</p> | 0..1 |
| compositionId | <p>The identifier attribute of the Composition element that represents the physical part of a piece of equipment where the data represented by the DataItem element originated.</p> <p>A <i>Valid Data Value</i> reported for compositionId MUST be the value of the id attribute for the Composition element identified.</p> <p>compositionId is an optional attribute.</p> | 0..1 |

1206 Note: [†]One of componentID, compositionId , or dataItemId **MUST** be provided.

1207 **7.2.3.2 Constraints Element for DataItem**

1208 For some types of DataItem elements, the expected value(s) for the data reported for the
 1209 DataItem **MAY** be restricted to specific values or a range of values.

1210 Constraints is an optional XML element that provides a way to define the expected
 1211 value(s) or the upper and lower limits for the range of values that are expected to be
 1212 reported in response to a *Current Request* or *Sample Request*.

1213 Constraints are used by a software application to evaluate the validity of the data
 1214 reported.

1215 The value associated with each Constraint element is reported in the CDATA for that
 1216 element.

1217 **7.2.3.2.1 Schema for Constraints**

1218 The XML schema in *Figure 21* represents the structure of the Constraints XML
 1219 element and the elements defined for Constraints.

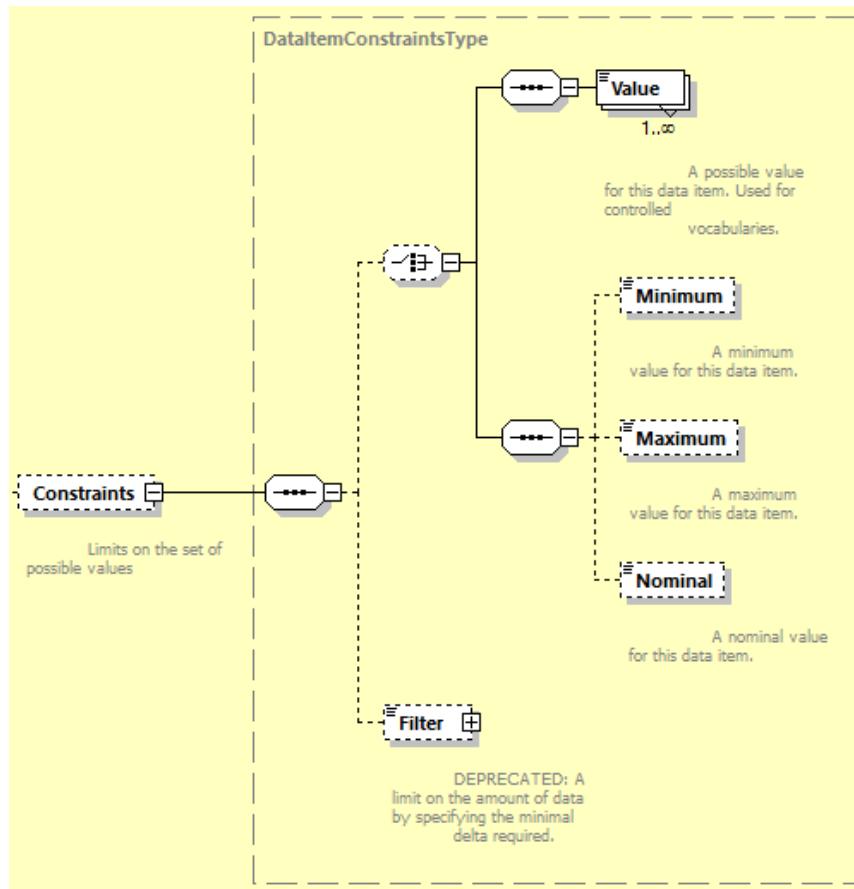


Figure 21: Constraints Diagram

1220 *Table 36* identifies the elements available to identify Constraints for a measured value:

Table 36: Elements for Constraints

| Element | Description | Occurrence |
|---------|--|------------|
| Value | <p>Value represents a single data value that is expected to be reported for a DataItem element.</p> <p>The data value is provided in the CDATA for this element and may be any numeric or text content.</p> <p>When there are multiple data values that may be expected to be reported for a DataItem element, multiple Value elements may be defined.</p> <p>In the case where only one Value element is defined, the data returned in response to a <i>Current Request</i> or <i>Sample Request</i> request MUST be the data value defined for Value element.</p> <p>Value MUST NOT be used in conjunction with any other Constraint elements.</p> | 0..* |
| Maximum | <p>If the data reported for a data item is a range of numeric values, the expected value reported MAY be described with an upper limit defined by this constraint.</p> <p>The data value is provided in the CDATA for this element and MUST be a value using the same units as the reported data.</p> | 0..1 |
| Minimum | <p>If the data reported for a data item is a range of numeric values, the expected value reported MAY be described with a lower limit defined by this constraint.</p> <p>The data value is provided in the CDATA for this element and MUST be a value using the same units as the reported data.</p> | 0..1 |
| Nominal | <p>The target or expected value for this data item.</p> <p>The data value is provided in the CDATA for this element and MUST be a value using the same units as the reported data.</p> | 0..1 |

| Continuation of Table 36 | | |
|--------------------------|--|-------------------|
| Element | Description | Occurrence |
| Filter | <p>DEPRECATED in Version 1.4 – Moved to the Filters element of a DataItem.</p> <p>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.</p> | 0..1 [†] |

1221 Note: [†]Remains in schema for backwards compatibility.

1222 7.2.3.3 Filters Element for DataItem

1223 Filters is an optional XML container that organizes the Filter elements for DataItem.

1224 Filters contains one or more Filter XML elements.

Table 37: MTConnect Filters Element

| Element | Description | Occurrence |
|---------|---|------------|
| Filters | An XML container consisting of one or more types of Filter XML elements. Only one Filters container MAY appear for a DataItem element. | 0..1 |

1225 **7.2.3.3.1 Filter**

1226 Filter provides a means to control when an *Agent* records updated information for a
 1227 data item. Currently, there are two types of Filter elements defined in the MTConnect
 1228 Standard - MINIMUM_DELTA and PERIOD. More Filter types may be added in the
 1229 future.

1230 The value associated with each Filter element is reported in the CDATA for that ele-
 1231 ment.

1232 *Figure 22* represents the structure for Filter XML element.

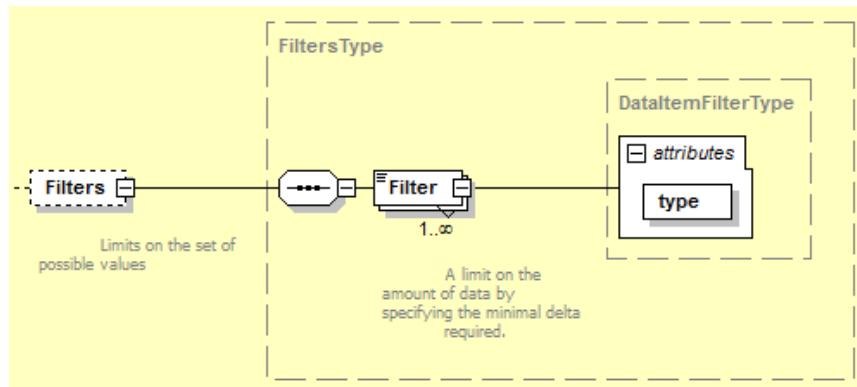


Figure 22: Filter Diagram

1233 *Table 38* describes the types of Filter defined for a DataItem element and the ex-
 1234 pected behavior of an *Agent* when a Filter is applied to DataItem element.

Table 38: DataItem Element Filter type

| type | Description | Occurrence |
|---------------|---|-------------------|
| MINIMUM_DELTA | <p>For a MINIMUM_DELTA type Filter, a new value MUST NOT be reported for a data item unless the measured value has changed from the last reported value by at least the delta given as the CDATA of this element.</p> <p>The CDATA MUST be an absolute value using the same units as the reported data.</p> | 0..1 [†] |

| Continuation of Table 38 | | |
|--------------------------|--|-------------------|
| type | Description | Occurrence |
| PERIOD | <p>For a PERIOD type Filter, the data reported for a data item is provided on a periodic basis. The PERIOD for reporting data is defined in the CDATA for the Filter.</p> <p>The CDATA MUST be an absolute value reported in seconds representing the time between reported samples of the value of the data item.</p> <p>If the PERIOD is smaller than one second, the number can be represented as a floating point number. For example, a PERIOD of 100 milliseconds would be 0.1.</p> | 0..1 [†] |

1235 [†]Note: Either MINIMUM_DELTA or PERIOD can be defined, not both.

1236 **7.2.3.4 InitialValue Element for DataItem**

1237 InitialValue is an XML element that defines the value to be set for the data item after
 1238 a reset event.

1239 The value associated with the InitialValue element is reported in the CDATA for this
 1240 element and **MUST** be an absolute value using the same units as the reported data.

1241 **7.2.3.5 ResetTrigger Element for DataItem**

1242 The value of some data types is periodically reset to the value of the InitialValue element.
 1243 These reset events may be based upon a specific elapsed time or may be triggered by
 1244 a physical or logical reset action that causes the reset to occur. ResetTrigger provides
 1245 additional information regarding the meaning of the data – establishing an understanding
 1246 of the time frame that the data represents so that the data may be correctly understood by
 1247 a client software application.

Table 39: MTConnect ResetTrigger Element

| Element | Description | Occurrence |
|--------------|---|------------|
| ResetTrigger | <p>ResetTrigger is an XML element that describes the reset action that causes a reset to occur.</p> <p>It is additional information regarding the meaning of the data that establishes an understanding of the time frame that the data represents so that the data may be correctly understood by a client software application.</p> | 0..1 |

1248 The reset action that **MAY** cause a reset to occur is provided in the CDATA for this element.
 1249

1250 The reset actions that may cause a reset to occur are described in *Table 40*.

Table 40: DataItem Element ResetTrigger type

| Reset Actions | Description |
|-----------------|--|
| ACTION_COMPLETE | The value of the <i>Data Entity</i> that is measuring an action or operation is to be reset upon completion of that action or operation. |
| ANNUAL | The value of the <i>Data Entity</i> is to be reset at the end of a 12-month period. |
| DAY | The value of the <i>Data Entity</i> is to be reset at the end of a 24-hour period. |
| LIFE | The value of the <i>Data Entity</i> is not reset and accumulates for the entire life of the piece of equipment. |
| MAINTENANCE | The value of the <i>Data Entity</i> is to be reset upon completion of a maintenance event. |
| MONTH | The value of the <i>Data Entity</i> is to be reset at the end of a monthly period. |
| POWER_ON | The value of the <i>Data Entity</i> is to be reset when power was applied to the piece of equipment after a planned or unplanned interruption of power has occurred. |

| Continuation of Table 40 | |
|--------------------------|--|
| Reset Actions | Description |
| SHIFT | The value of the <i>Data Entity</i> is to be reset at the end of a work shift. |
| WEEK | The value of the <i>Data Entity</i> is to be reset at the end of a 7-day period. |

1251 8 Listing of Data Items

1252 In the MTConnect Standard, DataItem elements are defined and organized based upon
1253 the category and type attributes. The category attribute provides a high level
1254 grouping for DataItem elements based on the kind of information that is reported by
1255 the data item.

1256 These categories are:

- 1257 • SAMPLE

1258 A SAMPLE reports a continuously variable or analog data value.

- 1259 • EVENT

1260 An EVENT reports information representing a functional state, with two or more
1261 discrete values, associated with a component or it contains a message. The data
1262 provided may be a numeric value or text.

- 1263 • CONDITION

1264 A CONDITION reports information about the health of a piece of equipment and its
1265 ability to function.

1266 The type attribute specifies the specific kind of data that is reported. For some types of
1267 data items, a subType attribute may also be used to differentiate between multiple data
1268 items of the same type where the information reported by the data item has a different,
1269 but related, meaning.

1270 Many types of data items provide two forms of data: a value (reported as either a SAMPLE
1271 or EVENT) and a health status (reported as a CONDITION). These DataItem types **MAY**
1272 be defined in more than one category based on the data that they report.

1273 8.1 Data Items in category SAMPLE

1274 The types of DataItem elements in the SAMPLE category report data representing a
 1275 continuously changing or analog data value. This data can be measured at any point-in-
 1276 time and will always produce a result. The data provided may be a scalar floating point
 1277 number or integers that have an infinite number of possible values. The units attribute
 1278 **MUST** be defined and reported for each DataItem in this category.

1279 *Table 41* defines the types and subtypes of DataItem elements defined for the SAMPLE
 1280 category. The subtypes are indented below their associated types.

Table 41: DataItem type subType for category SAMPLE

| DataItem 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. DEPRECATION WARNING : May be deprecated in the future. Recommend using PROCESS_TIMER and EQUIPMENT_TIMER. | SECOND |
| AMPERAGE | The measurement of electrical current. | AMPERE |
| ACTUAL | The measured amperage being delivered from a power source. | AMPERE |
| ALTERNATING | The measurement of alternating current. If not specified further in statistic, defaults to RMS voltage. | AMPERE |
| DIRECT | The measurement of DC current. | AMPERE |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|--|----------------------------|
| DataItem type/subType | Description | Units |
| TARGET | The desired or preset amperage to be delivered from a power source. | 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 |
| ANGULAR_ACCELERATION | Rate of change of angular velocity. | DEGREE/SECOND ² |
| ANGULAR_VELOCITY | Rate of change of angular position. | DEGREE/SECOND |
| AXIS_FEEDRATE | The feedrate of a linear axis. | MILLIMETER/SECOND |
| ACTUAL | The measured value of the feedrate of a linear axis. | MILLIMETER/SECOND |
| COMMANDED | <p>The feedrate of a linear axis as specified by the Controller type component.</p> <p>The COMMANDED feedrate is a calculated value that includes adjustments and overrides.</p> | MILLIMETER/SECOND |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|---|-----------------------------|
| DataItem type/subType | Description | Units |
| 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 |
| OVERRIDE | The operator's overridden value. Percent of commanded. DEPRECATED in Version 1.3. See EVENT category data items. | PERCENT |
| PROGRAMMED | The feedrate specified by a logic or motion program or set by a switch for a linear axis. | 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 a linear axis when operating in a rapid positioning mode. | MILLIMETER/SECOND |
| CAPACITY_FLUID | The fluid capacity of an object or container. | MILLILITER |
| CAPACITY_SPATIAL | The geometric capacity of an object or container. | CUBIC_MILLIMETER |
| 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 . fffff |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|---|---|
| DataItem type/subType | Description | Units |
| CONCENTRATION | Percentage of one component within a mixture of components. | PERCENT |
| CONDUCTIVITY | The ability of a material to conduct electricity. | SIEMENS/METER |
| CUTTING_SPEED | The speed difference (relative velocity) between the cutting mechanism and the surface of the workpiece it is operating on. | MILLIMETER/SECOND |
| ACTUAL | The measured value between the cutting mechanism and the surface of the workpiece it is operating on. | MILLIMETER/SECOND |
| COMMANDED | The commanded value between the cutting mechanism and the surface of the workpiece it is operating on. | MILLIMETER/SECOND |
| PROGRAMMED | The programmed value between the cutting mechanism and the surface of the workpiece it is operating on. | MILLIMETER/SECOND |
| DENSITY | The volumetric mass of a material per unit volume of that material. | MILLIGRAM/CUBIC_- MILLIMETER |
| DEPOSITION_- ACCELERATION_- VOLUMETRIC | The rate of change in spatial volume of material deposited in an additive manufacturing process. | CUBIC_- MILLIMETER/SECOND ² |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|--|---|
| DataItem type/subType | Description | Units |
| ACTUAL | The measured rate of change in spatial volume of material deposited in an additive manufacturing process. | CUBIC_- MILLIMETER/SECOND ² |
| COMMANDER | The commanded rate of change in spatial volume of material to be deposited in an additive manufacturing process. | CUBIC_- MILLIMETER/SECOND ² |
| DEPOSITION_DENSITY | The density of the material deposited in an additive manufacturing process per unit of volume. | MILLIGRAM/CUBIC_- MILLIMETER |
| ACTUAL | The measured density of the material deposited in an additive manufacturing process. | MILLIGRAM/CUBIC_- MILLIMETER |
| COMMANDER | The commanded density of material to be deposited in an additive manufacturing process. | MILLIGRAM/CUBIC_- MILLIMETER |
| DEPOSITION_MASS | The mass of the material deposited in an additive manufacturing process. | MILLIGRAM |
| ACTUAL | The measured mass of the material deposited in an additive manufacturing process. | MILLIGRAM |
| COMMANDER | The commanded mass of the material to be deposited in an additive manufacturing process. | MILLIGRAM |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|--|--------------------------|
| DataItem type/subType | Description | Units |
| DEPOSITION_RATE--VOLUMETRIC | The rate at which a spatial volume of material is deposited in an additive manufacturing process. | CUBIC_-MILLIMETER/SECOND |
| ACTUAL | The measured rate at which a spatial volume of material is deposited in an additive manufacturing process. | CUBIC_-MILLIMETER/SECOND |
| COMMANDDED | The programmed rate at which a spatial volume of material is to be deposited in an additive manufacturing process. | CUBIC_-MILLIMETER/SECOND |
| DEPOSITION_VOLUME | The spatial volume of material to be deposited in an additive manufacturing process. | CUBIC_MILLIMETER |
| ACTUAL | The measured spatial volume of material deposited. | CUBIC_MILLIMETER |
| COMMANDDED | The target spatial volume of material to be deposited. | CUBIC_MILLIMETER |
| DISPLACEMENT | The change in position of an object. | MILLIMETER |
| ELECTRICAL_ENERGY | The measurement of electrical energy consumption by a component. | WATT_SECOND |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|---|--------|
| DataItem type/subType | Description | Units |
| EQUIPMENT_TIMER | <p>The measurement of the amount of time a piece of equipment or a sub-part of a piece of equipment has performed specific activities. Often used to determine when maintenance may be required for the equipment.</p> <p>Multiple subTypes of EQUIPMENT_TIMER MAY be defined.</p> <p>A subType MUST always be specified.</p> | SECOND |
| DELAY | Measurement of the time that a piece of equipment is waiting for an event or an action to occur. | SECOND |
| LOADED | <p>Measurement of the time that the sub-parts of a piece of equipment are under load.</p> <p>Example: For traditional machine tools, this is a measurement of the time that the cutting tool is assumed to be engaged with the part.</p> | SECOND |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|---|--------|
| DataItem type/subType | Description | Units |
| OPERATING | <p>Measurement of the time that the major sub-parts of a piece of equipment are powered or performing any activity whether producing a part or product or not.</p> <p>Example: For traditional machine tools, this includes WORKING, plus idle time.</p> | SECOND |
| POWERED | <p>The measurement of time that primary power is applied to the piece of equipment and, as a minimum, the controller or logic portion of the piece of equipment is powered and functioning or components that are required to remain on are powered.</p> <p>Example: Heaters for an extrusion machine that are required to be powered even when the equipment is turned off</p> | SECOND |
| WORKING | <p>Measurement of the time that a piece of equipment is performing any activity the equipment is active and performing a function under load or not.</p> <p>Example: For traditional machine tools, this includes LOADED, plus rapid moves, tool changes, etc.</p> | SECOND |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|--|--------------|
| DataItem type/subType | Description | Units |
| 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 Version 1.1 | None |
| LENGTH | The length of an object. | MILLIMETER |
| REMAINING | The remaining total length of an object. | MILLIMETER |
| STANDARD | The standard or original length of an object. | MILLIMETER |
| USEABLE | The remaining useable length of an object. | MILLIMETER |
| LEVEL | DEPRECATED in Version 1.2. See FILL_LEVEL | None |
| LINEAR_FORCE | The measurement 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 piece of equipment. | PERCENT |
| MASS | The measurement of the mass of an object(s) or an amount of material. | KILOGRAM |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|--|-------------------|
| DataItem type/subType | Description | Units |
| PATH_FEEDRATE | The feedrate for the axes, or a single axis, associated with a Path component— a vector. | MILLIMETER/SECOND |
| ACTUAL | The measured value of the feedrate of the axes, or a single axis, associated with a path component. | MILLIMETER/SECOND |
| COMMANDED | <p>The feedrate as specified by the Controller type component for the axes, or a single axis, associated with a Path component.</p> <p>The COMMANDED feedrate is a calculated value that includes adjustments and overrides.</p> | 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, or a single axis, associated with a Path when operating in a manual state or method (jogging). | MILLIMETER/SECOND |
| OVERRIDE | <p>The operator's overridden value. Percent of commanded.</p> <p>DEPRECATED in Version 1.3. See EVENT category data items.</p> | PERCENT |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|--|----------------------------|
| DataItem 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, or a single axis, 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, or a single axis, associated with a Path when operating in a rapid positioning mode. | MILLIMETER/SECOND |
| PATH_FEEDRATE_- PER_REVOLUTION | The feedrate for the axes, or a single axis. | MILLIMETER/REVO- LUTION |
| ACTUAL | The measured value of the feedrate of the axes, or a single axis. | MILLIMETER/REVO- LUTION |
| COMMANDED | The feedrate as specified by the Controller for the axes, or a single axis. The COMMANDED feedrate is a calculated value that includes adjustments and overrides. | MILLIMETER/REVO- LUTION |
| PROGRAMMED | The feedrate specified by a logic or motion program or set by a switch as the feedrate for the axes, or a single axis. | MILLIMETER/REVO- LUTION |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|---|---------------|
| DataItem type/subType | Description | Units |
| PATH_POSITION | <p>A measured or calculated position of a control point associated with a piece of equipment. The control point MUST be reported as a set of space-delimited floating-point numbers representing a point in 3-D space. The position of the control point MUST be reported in units of MILLIMETER and listed in order of X, Y, and Z referenced to the coordinate system of the piece of equipment. Any control point representing a position in 1-D or 2-D space MAY be represented in terms of 3-D space by setting any undefined coordinate to zero (0).</p> <p>PATH_POSITION SHOULD be further defined with a coordinateSystem attribute. If a coordinateSystem attribute is not specified, the position of the control point MUST be reported in WORK coordinates.</p> | MILLIMETER_3D |
| ACTUAL | The measured position of the current program control point as reported by the piece of equipment. | MILLIMETER_3D |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|---|---------------|
| DataItem type/subType | Description | Units |
| PROGRAMMED | The position of the control point specified by a logic or motion program. | MILLIMETER_3D |
| COMMANDED | The position computed by the Controller type component. | MILLIMETER_3D |
| PROBE | The position provided by a measurement probe. | 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 |
| PH | The measurement of the acidity or alkalinity. | PH |
| POSITION | A measured or calculated position of a Component element as reported by a piece of equipment. POSITION SHOULD be further defined with a coordinateSystem attribute. If a coordinateSystem attribute is not specified, the position of the control point MUST be reported in MACHINE coordinates. | MILLIMETER |
| ACTUAL | The physical measured position of the control point for a Component. | MILLIMETER |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|---|------------|
| DataItem type/subType | Description | Units |
| COMMANDED | A position calculated by the Controller type component for a discrete movement. | MILLIMETER |
| PROGRAMMED | The position of the control point for a Component specified by a logic or motion program. | MILLIMETER |
| TARGET | The desired end position of the control point for 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 |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|--|--------|
| DataItem type/subType | Description | Units |
| PROCESS_TIMER | <p>The measurement of the amount of time a piece of equipment has performed different types of activities associated with the process being performed at that piece of equipment.</p> <p>Multiple subtypes of PROCESS_TIMER may be defined.</p> <p>Typically, PROCESS_TIMER SHOULD be modeled as a data item for the Device element, but MAY be modeled for either a Controller or Path <i>Structural Element</i> in the XML document.</p> <p>A subType MUST always be specified.</p> | SECOND |
| DELAY | Measurement of the time that a process is waiting and unable to perform its intended function. | SECOND |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|---|-------------------|
| DataItem type/subType | Description | Units |
| PROCESS | The measurement of the time from the beginning of production of a part or product on a piece of equipment until the time that production is complete for that part or product on that piece of equipment. This includes the time that the piece of equipment is running, producing parts or products, or in the process of producing parts. | SECOND |
| RESISTANCE | The degree to which a substance opposes the passage of an electric current. | 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 |
| 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 |
| OVERRIDE | The operator's overridden value. Percent of commanded. DEPRECATED in Version 1.3. See EVENT category data items. | PERCENT |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|--|-------------------|
| DataItem type/subType | Description | Units |
| PROGRAMMED | The rotational velocity specified by a logic or motion program or set by a switch. | REVOLUTION/MINUTE |
| SOUND_LEVEL | The measurement of a sound level or sound pressure level relative to atmospheric pressure. | 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 |
| NO_SCALE | No weighting factor on the frequency scale | DECIBEL |
| SPINDLE_SPEED | DEPRECATED in Version 1.2. Replaced by ROTARY_VELOCITY | REVOLUTION/MINUTE |
| 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 |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|---|-------------------|
| DataItem type/subType | Description | Units |
| TEMPERATURE | The measurement of temperature. | CELSIUS |
| TENSION | The measurement of a force that stretches or elongates an object. | NEWTON |
| TILT | The measurement of angular displacement. | MICRO_RADIAN |
| TORQUE | The turning force exerted on an object or by an object. | NEWTON_METER |
| VELOCITY | The rate of change of position. | MILLIMETER/SECOND |
| VISCOSITY | The measurement of a fluids resistance to flow. | PASCAL_SECOND |
| VOLTAGE | The measurement of electrical potential between two points. | VOLT |
| ACTUAL | The measured voltage being delivered from a power source. | 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 |
| TARGET | The desired or preset voltage to be delivered from a power source. | VOLT |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|--|-----------------------|
| DataItem type/subType | Description | Units |
| VOLT_AMPERE | The measurement 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 |
| VOLUME_FLUID | The fluid volume of an object or container. | MILLILITER |
| ACTUAL | The amount of fluid currently present in an object or container. | MILLILITER |
| CONSUMED | The amount of fluid material consumed from an object or container during a manufacturing process. | MILLILITER |
| VOLUME_SPATIAL | The geometric volume of an object or container. | CUBIC_MILLIMETER |
| ACTUAL | The amount of bulk material currently present in an object or container. | CUBIC_MILLIMETER |
| CONSUMED | The amount of bulk material consumed from an object or container during a manufacturing process. | CUBIC_MILLIMETER |

| Continuation of Table 41: DataItem type subType for category SAMPLE | | |
|---|--|-------|
| DataItem type/subType | Description | Units |
| WATTAGE | The measurement of power flowing through or dissipated by an electrical circuit or piece of equipment. | WATT |
| ACTUAL | The measured wattage being delivered from a power source. | WATT |
| TARGET | The desired or preset wattage to be delivered from a power source. | WATT |

1281 8.2 Data Items in category EVENT

1282 DataItem types in the EVENT category represent a discrete piece of information from a
 1283 piece of equipment. EVENT does not have intermediate values that vary over time.

1284 An EVENT is information that, when provided at any specific point in time, represents the
 1285 current state of the piece of equipment.

1286 There are two types of EVENT: those representing state, with two or more discrete values,
 1287 and those representing messages that contain plain text data.

1288 *Table 42* defines the DataItem types and subtypes defined for the EVENT category. The
 1289 subtypes are indented below their associated types.

Table 42: DataItem type subType for category EVENT

| DataItem type subType | Description |
|-----------------------|---|
| ACTIVE_AXES | <p>The set of axes currently associated with a Path or Controller <i>Structural Element</i>. If this DataItem is not provided, it will be assumed that all axes are currently associated with the Controller <i>Structural Element</i> and with an individual Path.</p> <p>The <i>Valid Data Value</i> for ACTIVE_AXES SHOULD be a space-delimited set of axes reported as the value of the name attribute for each axis. If name is not available, the piece of equipment MUST report the value of the nativeName attribute for each axis.</p> |
| ACTUATOR_STATE | <p>Represents the operational state of an apparatus for moving or controlling a mechanism or system.</p> <p>The <i>Valid Data Value</i> MUST be ACTIVE or INACTIVE.</p> |
| ALARM | DEPRECATED in Version 1.1. Replaced with CONDITION category. |
| AVAILABILITY | <p>Represents the Agent's ability to communicate with the data source.</p> <p>This MUST be provided for a Device Element and MAY be provided for any other <i>Structural Element</i>. The <i>Valid Data Value</i> MUST be AVAILABLE or UNAVAILABLE.</p> |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|---|
| DataItem type subType | Description |
| AXIS_COUPLING | <p>Describes the way the axes will be associated to each other.</p> <p>This is used in conjunction with COUPLED_AXES to indicate the way they are interacting.</p> <p>The <i>Valid Data Value</i> MUST be TANDEM, SYNCHRONOUS, MASTER, and SLAVE.</p> <p>The coupling MUST be viewed from the perspective of a specific axis. Therefore, a MASTER coupling indicates that this axis is the master for the COUPLED_AXES.</p> |
| AXIS_FEEDRATE_OVERRIDE | <p>The value of a signal or calculation issued to adjust the feedrate of an individual linear type axis.</p> <p>The value provided for AXIS_FEEDRATE_OVERRIDE is expressed as a percentage of the designated feedrate for the axis.</p> <p>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.</p> <p>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 subtypes of operation of an axis are currently defined as PROGRAMMED, JOG, and RAPID.</p> |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|--|
| DataItem type subType | Description |
| JOG | <p>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).</p> <p>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.</p> |
| PROGRAMMED | <p>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.</p> <p>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.</p> |
| RAPID | <p>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.</p> <p>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.</p> |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|---|
| DataItem type subType | Description |
| 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 <i>Valid Data Value</i> MUST be ACTIVE or INACTIVE. |
| AXIS_STATE | An indicator of the controlled state of a Linear or Rotary component representing an axis. The <i>Valid Data Value</i> MUST be HOME, TRAVEL, PARKED, or STOPPED. |
| BLOCK | The line of code or command being executed by a Controller <i>Structural Element</i> . The value reported for Block MUST include the entire expression for a line of program code, including all parameters. |
| BLOCK_COUNT | The total count of the number of blocks of program code that have been executed since execution started. BLOCK_COUNT counts blocks of program code executed regardless of program structure (e.g., looping or branching within the program). The starting value for BLOCK_COUNT MAY be established by an initial value provided in the Constraint element defined for the data item. |
| 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 <i>Valid Data Value</i> MUST be ACTIVE or INACTIVE. |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|--|
| DataItem type subType | Description |
| MANUAL_UNCLAMP | <p>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.</p> <p>The <i>Valid Data Value</i> MUST be ACTIVE or INACTIVE.</p> <p>When MANUAL_UNCLAMP is ACTIVE, it is expected that a chuck cannot be unclamped until MANUAL_UNCLAMP is set to INACTIVE.</p> |
| CHUCK_STATE | <p>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 piece of equipment.</p> <p>The <i>Valid Data Value</i> MUST be OPEN, CLOSED, or UNLATCHED.</p> |
| CODE | DEPRECATED in Version 1.1. |
| COMPOSITION_STATE | <p>An indication of the operating condition of a mechanism represented by a Composition type element.</p> <p>A subType MUST always be specified.</p> <p>A compositionId MUST always be specified.</p> |
| ACTION | <p>An indication of the operating state of a mechanism represented by a Composition type component.</p> <p>The operating state indicates whether the Composition element is activated or disabled.</p> <p>The <i>Valid Data Value</i> MUST be ACTIVE or INACTIVE.</p> |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|--|
| DataItem type subType | Description |
| LATERAL | <p>An indication of the position of a mechanism that may move in a lateral direction. The mechanism is represented by a Composition type component.</p> <p>The position information indicates whether the Composition element is positioned to the right, to the left, or is in transition.</p> <p>The <i>Valid Data Value</i> MUST be RIGHT, LEFT, or TRANSITIONING.</p> |
| MOTION | <p>An indication of the open or closed state of a mechanism. The mechanism is represented by a Composition type component.</p> <p>The operating state indicates whether the state of the Composition element is open, closed, or unlatched.</p> <p>The <i>Valid Data Value</i> MUST be OPEN, UNLATCHED, or CLOSED.</p> |
| SWITCHED | <p>An indication of the activation state of a mechanism represented by a Composition type component.</p> <p>The activation state indicates whether the Composition element is activated or not.</p> <p>The <i>Valid Data Value</i> MUST be ON or OFF.</p> |
| VERTICAL | <p>An indication of the position of a mechanism that may move in a vertical direction. The mechanism is represented by a Composition type component.</p> <p>The position information indicates whether the Composition element is positioned to the top, to the bottom, or is in transition.</p> <p>The <i>Valid Data Value</i> MUST be UP, DOWN, or TRANSITIONING.</p> |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|---|
| DataItem type subType | Description |
| CONTROLLER_MODE | The current mode of the Controller component. The <i>Valid Data Value</i> MUST be AUTOMATIC, MANUAL, MANUAL_DATA_INPUT, SEMI_AUTOMATIC, or EDIT. |
| CONTROLLER_MODE_OVERRIDE | A setting or operator selection that changes the behavior of a piece of equipment. A subType MUST always be specified. |
| DRY_RUN | A setting or operator selection used to execute a test mode to confirm the execution of machine functions. The <i>Valid Data Value</i> MUST be ON or OFF. When DRY_RUN is ON, the equipment performs all of its normal functions, except no part or product is produced. If the equipment has a spindle, spindle operation is suspended. |
| MACHINE_AXIS_LOCK | A setting or operator selection that changes the behavior of the controller on a piece of equipment. The <i>Valid Data Value</i> MUST be ON or OFF. When MACHINE_AXIS_LOCK is ON, program execution continues normally, but no equipment motion occurs |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|--|
| DataItem type subType | Description |
| OPTIONAL_STOP | <p>A setting or operator selection that changes the behavior of the controller on a piece of equipment.</p> <p>The <i>Valid Data Value</i> MUST be ON or OFF.</p> <p>The program execution is stopped after a specific program block is executed when OPTIONAL_STOP is ON.</p> <p>In the case of a G-Code program, a program BLOCK containing a M01 code designates the command for an OPTIONAL_STOP.</p> <p>EXECUTION MUST change to OPTIONAL_STOP after a program block specifying an optional stop is executed and the OPTIONAL_STOP selection is ON.</p> |
| SINGLE_BLOCK | <p>A setting or operator selection that changes the behavior of the controller on a piece of equipment.</p> <p>The <i>Valid Data Value</i> MUST be ON or OFF.</p> <p>Program execution is paused after each BLOCK of code is executed when SINGLE_BLOCK is ON.</p> <p>When SINGLE_BLOCK is ON, EXECUTION MUST change to INTERRUPTED after completion of each BLOCK of code.</p> |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|--|
| DataItem type subType | Description |
| TOOL_CHANGE_STOP | <p>A setting or operator selection that changes the behavior of the controller on a piece of equipment.</p> <p>The <i>Valid Data Value</i> MUST be ON or OFF.</p> <p>Program execution is paused when a command is executed requesting a cutting tool to be changed.</p> <p>EXECUTION MUST change to INTERRUPTED after completion of the command requesting a cutting tool to be changed and TOOL_CHANGE_STOP is ON.</p> |
| COUPLED_AXES | <p>Refers to the set of associated axes.</p> <p>The <i>Valid Data Value</i> for COUPLED_AXES SHOULD be a space-delimited set of axes reported as the value of the name attribute for each axis. If name is not available, the piece of equipment MUST report the value of the nativeName attribute for each axis.</p> |
| DATE_CODE | <p>The time and date code associated with a material or other physical item.</p> <p>DATE_CODE MUST be reported in ISO 8601 format.</p> |
| MANUFACTURE | The time and date code relating to the production of a material or other physical item. |
| EXPIRATION | The time and date code relating to the expiration or end of useful life for a material or other physical item. |
| FIRST_USE | The time and date code relating the first use of a material or other physical item. |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|--|
| DataItem type subType | Description |
| DEVICE_UUID | <p>The identifier of another piece of equipment that is temporarily associated with a component of this piece of equipment to perform a particular function.</p> <p>The <i>Valid Data Value</i> MUST be a NMTOKEN XML type.</p> |
| DIRECTION | <p>The direction of motion. A subType MUST always be specified.</p> |
| LINEAR | <p>The direction of motion of a linear motion.</p> <p>The <i>Valid Data Value</i> MUST be POSITIVE or NEGATIVE.</p> |
| ROTARY | <p>The rotational direction of a rotary motion using the right hand rule convention.</p> <p>The <i>Valid Data Value</i> MUST be CLOCKWISE or COUNTER_CLOCKWISE.</p> |
| DOOR_STATE | <p>The operational state of a DOOR type component or composition element.</p> <p>The <i>Valid Data Value</i> MUST be OPEN, UNLATCHED, or CLOSED.</p> |
| EMERGENCY_STOP | <p>The current state of the emergency stop signal for a piece of equipment, controller path, or any other component or subsystem of a piece of equipment.</p> <p>The <i>Valid Data Value</i> 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).</p> |
| END_OF_BAR | <p>An indication of whether the end of a piece of bar stock being feed by a bar feeder has been reached.</p> <p>The <i>Valid Data Value</i> MUST be expressed as a Boolean expression of YES or NO.</p> |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|---|
| DataItem type subType | Description |
| 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. |
| 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 subType is specified, PRIMARY MUST be the default END_OF_BAR indication. |
| EQUIPMENT_MODE | An indication that a piece of equipment, or a sub-part of a piece of equipment, is performing specific types of activities. EQUIPMENT_MODE MAY have more than one subtype defined. A subType MUST always be specified. |
| DELAY | An indication that a piece of equipment is waiting for an event or an action to occur. |
| LOADED | An indication that the sub-parts of a piece of equipment are under load. Example: For traditional machine tools, this is an indication that the cutting tool is assumed to be engaged with the part. The <i>Valid Data Value</i> MUST be ON or OFF. |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|---|
| DataItem type subType | Description |
| OPERATING | <p>An indication that the major sub-parts of a piece of equipment are powered or performing any activity whether producing a part or product or not.</p> <p>Example: For traditional machine tools, this includes when the piece of equipment is WORKING or it is idle.</p> <p>The <i>Valid Data Value</i> MUST be ON or OFF.</p> |
| POWERED | <p>An indication that primary power is applied to the piece of equipment and, as a minimum, the controller or logic portion of the piece of equipment is powered and functioning or components that are required to remain on are powered.</p> <p>Example: Heaters for an extrusion machine that required to be powered even when the equipment is turned off.</p> <p>The <i>Valid Data Value</i> MUST be ON or OFF.</p> |
| WORKING | <p>An indication that a piece of equipment is performing any activity the equipment is active and performing a function under load or not.</p> <p>Example: For traditional machine tools, this includes when the piece of equipment is LOADED, making rapid moves, executing a tool change, etc.</p> <p>The <i>Valid Data Value</i> MUST be ON or OFF.</p> |
| EXECUTION | <p>The execution status of the Controller.</p> <p>The <i>Valid Data Value</i> MUST be READY, ACTIVE, INTERRUPTED, WAIT, FEED_HOLD, STOPPED, OPTIONAL_STOP, PROGRAM_STOPPED, or PROGRAM_COMPLETED.</p> |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|--|
| DataItem type subType | Description |
| FUNCTIONAL_MODE | <p>The current intended production status of the device or component.</p> <p>Typically, the FUNCTIONAL_MODE SHOULD be modeled as a data item for the Device element, but MAY be modeled for any <i>Structural Element</i> in the XML document.</p> <p>The <i>Valid Data Value</i> MUST be PRODUCTION, SETUP, TEARDOWN, MAINTENANCE, or PROCESS_DEVELOPMENT.</p> |
| HARDNESS | <p>The measurement of the hardness of a material.</p> <p>The measurement does not provide a unit.</p> <p>A subType MUST always be specified to designate the hardness scale associated with the measurement.</p> |
| BRINELL | A scale to measure the resistance to deformation of a surface. |
| LEEB | A scale to measure the elasticity of a surface. |
| MOHS | A scale to measure the resistance to scratching of a surface. |
| ROCKWELL | A scale to measure the resistance to deformation of a surface. |
| SHORE | A scale to measure the resistance to deformation of a surface. |
| VICKERS | A scale to measure the resistance to deformation of a surface. |
| INTERFACE_STATE | <p>The current functional or operational state of an Interface type element indicating whether the interface is active or is not currently functioning.</p> <p>The <i>Valid Data Value</i> MUST be ENABLED or DISABLED.</p> |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|---|
| DataItem type subType | Description |
| 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. DEPRECATED in Version 1.4.0. |
| MAXIMUM | The maximum line number of the code being executed. |
| MINIMUM | The minimum line number of the code being executed. |
| LINE_LABEL | An optional identifier for a BLOCK of code in a PROGRAM. |
| LINE_NUMBER | A reference to the position of a block of program code within a control program. The line number MAY represent either an absolute position starting with the first line of the program or an incremental position relative to the occurrence of the last LINE_LABEL. LINE_NUMBER does not change subject to any looping or branching in a control program. A subType MUST be defined. |
| ABSOLUTE | The position of a block of program code relative to the beginning of the control program. |
| INCREMENTAL | The position of a block of program code relative to the occurrence of the last LINE_LABEL encountered in the control program. |
| MATERIAL | The identifier of a material used or consumed in the manufacturing process. The <i>Valid Data Value</i> MUST be a text string. |
| MATERIAL_LAYER | Identifies the layers of material applied to a part or product as part of an additive manufacturing process. The <i>Valid Data Value</i> MUST be an integer. |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|--|
| DataItem type subType | Description |
| ACTUAL | The current number of layers of material applied to a part or product during an additive manufacturing process. |
| TARGET | The target or planned number layers of material applied to a part or product during an additive manufacturing process. |
| MESSAGE | Any text string of information to be transferred from a piece of equipment to a client software application. |
| OPERATOR_ID | <p>The identifier of the person currently responsible for operating the piece of equipment.</p> <p>DEPRECATION WARNING : May be deprecated in the future. See USER below.</p> |
| PALLET_ID | <p>The identifier for a pallet.</p> <p>The <i>Valid Data Value MUST</i> be a text string.</p> |
| PART_COUNT | <p>The current count of parts produced as represented by the Controller component.</p> <p>The <i>Valid Data Value MUST</i> be an integer value.</p> |
| ALL | The count of all the parts produced. If the subtype is not given, this is the default. |
| BAD | Indicates the count of incorrect parts produced. |
| GOOD | Indicates the count of correct parts made. |
| REMAINING | The number of parts remaining in stock or to be produced. |
| TARGET | Indicates the number of parts that are projected or planned to be produced. |
| PART_DETECT | <p>An indication designating whether a part or work piece has been detected or is present.</p> <p>The <i>Valid Data Value MUST</i> be PRESENT or NOT_PRESENT.</p> |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|---|
| DataItem type subType | Description |
| PART_ID | An identifier of a part in a manufacturing operation. The <i>Valid Data Value</i> MUST be a text string. |
| PART_NUMBER | An identifier of a part or product moving through the manufacturing process. The <i>Valid Data Value</i> MUST be a text string. DEPRECATION WARNING : May be deprecated in the future. |
| PATH_FEEDRATE_OVERRIDE | The value of a signal or calculation issued to adjust the feedrate for the axes associated with a Path component that 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. |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|--|
| DataItem type subType | Description |
| JOG | <p>The value of a signal or calculation issued to adjust the feedrate of the axes associated with a Path component when the axes, or a single axis, are being operated in a manual mode or method (jogging).</p> <p>When the JOG subtype of PATH_FEEDRATE_OVERRIDE is applied, the resulting commanded feedrate for the axes, or a single 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.</p> |
| PROGRAMMED | <p>The value of a signal or calculation issued to adjust the feedrate of the axes associated with a Path component when the axes, or a single axis, are operating as specified by a logic or motion program or set by a switch.</p> <p>When the PROGRAMMED subtype of PATH_FEEDRATE_OVERRIDE is applied, the resulting commanded feedrate for the axes, or a single 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.</p> |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|---|
| DataItem type subType | Description |
| RAPID | <p>The value of a signal or calculation issued to adjust the feedrate of the axes associated with a Path component when the axes, or a single axis, are being operated in a rapid positioning mode or method (rapid).</p> <p>When the RAPID subtype of PATH_FEEDRATE_OVERRIDE is applied, the resulting commanded feedrate for the axes, or a single 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.</p> |
| PATH_MODE | <p>Describes the operational relationship between a Path <i>Structural Element</i> and another Path <i>Structural Element</i> for pieces of equipment comprised of multiple logical groupings of controlled axes or other logical operations.</p> <p>The <i>Valid Data Value</i> MUST be INDEPENDENT, MASTER, SYNCHRONOUS, or MIRROR.</p> <p>The default value MUST be INDEPENDENT if PATH_MODE is not specified.</p> |
| POWER_STATE | <p>The indication of the status of the source of energy for a <i>Structural Element</i> to allow it to perform its intended function or the state of an enabling signal providing permission for the <i>Structural Element</i> to perform its functions.</p> <p>The <i>Valid Data Value</i> MUST be ON or OFF.</p> <p>DEPRECATION WARNING : May be deprecated in the future.</p> |
| CONTROL | The state of the enabling signal or control logic that enables or disables the function or operation of the <i>Structural Element</i> . |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|--|
| DataItem type subType | Description |
| LINE | The state of the power source for the <i>Structural Element</i> . |
| POWER_STATUS | DEPRECATED in Version 1.1.0. |
| PROCESS_TIME | The time and date associated with an activity or event. PROCESS_TIME MUST be reported in ISO 8601 format. |
| START | The time and date associated with the beginning of an activity or event. |
| COMPLETE | The time and date associated with the completion of an activity or event. |
| TARGET_COMPLETION | The projected time and date associated with the end or completion of an activity or event. |
| PROGRAM | The identity of the logic or motion program being executed by the piece of equipment. The <i>Valid Data Value</i> MUST be a text string. |
| SCHEDULE | The identity of a control program that is used to specify the order of execution of other programs. |
| MAIN | The identity of the primary logic or motion program currently being executed. It is the starting nest level in a call structure and may contain calls to sub programs. |
| ACTIVE | The identity of the logic or motion program currently executing. |
| PROGRAM_COMMENT | A comment or non-executable statement in the control program. The <i>Valid Data Value</i> MUST be a text string. |
| SCHEDULE | The identity of a control program that is used to specify the order of execution of other programs. |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|--|
| DataItem type subType | Description |
| MAIN | The identity of the primary logic or motion program currently being executed. It is the starting nest level in a call structure and may contain calls to sub programs. |
| ACTIVE | The identity of the logic or motion program currently executing. |
| PROGRAM_EDIT | <p>An indication of the status of the Controller components program editing mode.</p> <p>On many controls, a program can be edited while another program is currently being executed.</p> <p>The <i>Valid Data Value</i> MUST be:</p> <p>ACTIVE: The controller is in the program edit mode.</p> <p>READY: The controller is capable of entering the program edit mode and no function is inhibiting a change of mode.</p> <p>NOT_READY: A function is inhibiting the controller from entering the program edit mode.</p> |
| PROGRAM_EDIT_NAME | <p>The name of the program being edited.</p> <p>This is used in conjunction with PROGRAM_EDIT when in ACTIVE state.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p> |
| PROGRAM_HEADER | <p>The non-executable header section of the control program.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p> |
| PROGRAM_LOCATION | The Uniform Resource Identifier (URI) for the source file associated with PROGRAM. |
| SCHEDULE | An identity of a control program that is used to specify the order of execution of other programs. |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|--|
| DataItem type subType | Description |
| MAIN | The identity of the primary logic or motion program currently being executed. It is the starting nest level in a call structure and may contain calls to sub programs. |
| ACTIVE | The identity of the logic or motion program currently executing. |
| PROGRAM_LOCATION_TYPE | Defines whether the logic or motion program defined by PROGRAM is being executed from the local memory of the controller or from an outside source. The <i>Valid Data Value</i> MUST be LOCAL or EXTERNAL. |
| SCHEDULE | An identity of a control program that is used to specify the order of execution of other programs. |
| MAIN | The identity of the primary logic or motion program currently being executed. It is the starting nest level in a call structure and may contain calls to sub programs. |
| ACTIVE | The identity of the logic or motion program currently executing. |
| PROGRAM_NEST_LEVEL | An indication of the nesting level within a control program that is associated with the code or instructions that is currently being executed. If an initial value is not defined, the nesting level associated with the highest or initial nesting level of the program MUST default to zero (0). The value reported for PROGRAM_NEST_LEVEL MUST be an integer. |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|---|
| DataItem type subType | Description |
| ROTARY_MODE | <p>The current operating mode for a Rotary type axis.</p> <p>The <i>Valid Data Value MUST</i> be SPINDLE, INDEX, or CONTOUR.</p> |
| ROTARY_VELOCITY_OVERRIDE | <p>The value of a command issued to adjust the programmed velocity for a Rotary type axis.</p> <p>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.</p> <p>ROTARY_VELOCITY_OVERRIDE is expressed as a percentage of the programmed ROTARY_VELOCITY.</p> |
| SERIAL_NUMBER | <p>The serial number associated with a Component, Asset, or Device. The <i>Valid Data Value MUST</i> be a text string.</p> |
| SPINDLE_INTERLOCK | <p>An indication of the status of the spindle for a piece of equipment when power has been removed and it is free to rotate.</p> <p>The <i>Valid Data Value MUST</i> be:</p> <ul style="list-style-type: none"> ACTIVE if power has been removed and the spindle cannot be operated. INACTIVE if power to the spindle has not been deactivated. |
| TOOL_ASSET_ID | <p>The identifier of an individual tool asset. The <i>Valid Data Value MUST</i> be a text string.</p> |
| TOOL_GROUP | <p>An identifier for the tool group associated with a specific tool. Commonly used to designate spare tools.</p> |
| TOOL_ID | <p>DEPRECATED in Version 1.2.0. See TOOL_ASSET_ID. The identifier of the tool currently in use for a given Path.</p> |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|--|
| DataItem type subType | Description |
| TOOL_NUMBER | <p>The identifier assigned by the Controller component to a cutting tool when in use by a piece of equipment.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p> |
| TOOL_OFFSET | <p>A reference to the tool offset variables applied to the active cutting tool.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p> <p>The reported value returned for TOOL_OFFSET identifies the location in a table or list where the actual tool offset values are stored.</p> <p>DEPRECATED in V1.5 A subType MUST always be specified.</p> |
| LENGTH | A reference to a length type tool offset. |
| RADIAL | A reference to a radial type tool offset. |
| USER | <p>The identifier of the person currently responsible for operating the piece of equipment.</p> <p>A subType MUST always be specified.</p> |
| MAINTENANCE | The identifier of the person currently responsible for performing maintenance on the piece of equipment. |
| OPERATOR | The identifier of the person currently responsible for operating the piece of equipment. |
| SET_UP | The identifier of the person currently responsible for preparing a piece of equipment for production or restoring the piece of equipment to a neutral state after production. |
| VARIABLE | A data value whose meaning may change over time due to changes in the operation of a piece of equipment or the process being executed on that piece of equipment. |

| Continuation of Table 42: DataItem type subType for category EVENT | |
|--|--|
| DataItem type subType | Description |
| WAIT_STATE | An indication of the reason that EXECUTION is reporting a value of WAIT. The <i>Valid Data Value</i> MUST be POWERING_UP, POWERING_DOWN, PART_LOAD, PART_UNLOAD, TOOL_LOAD, TOOL_UNLOAD, MATERIAL_LOAD, MATERIAL_UNLOAD, SECONDARY_PROCESS, PAUSING, or RESUMING. |
| WIRE | The identifier for the type of wire used as the cutting mechanism in Electrical Discharge Machining or similar processes. The <i>Valid Data Value</i> MUST be a text string. |
| WORKHOLDING_ID | The identifier for the current workholding or part clamp in use by a piece of equipment. The <i>Valid Data Value</i> MUST be a text string. |
| WORK_OFFSET | A reference to the offset variables for a work piece or part associated with a Path in a Controller type component. The <i>Valid Data Value</i> MUST be a text string. The reported value returned for WORK_OFFSET identifies the location in a table or list where the actual tool offset values are stored. |

1290 8.3 Data Items in category CONDITION

1291 CONDITION category data items report data representing a *Structural Element*'s status
 1292 regarding its ability to operate or it provides an indication whether the data reported for
 1293 the *Structural Element* is within an expected range.

1294 CONDITION is reported differently than SAMPLE or EVENT. CONDITION **MUST** be
 1295 reported as Normal, Warning, or Fault.

1296 All DataItem types in the SAMPLE category **MAY** have associated CONDITION states.
 1297 CONDITION states indicate whether the value for the data is within an expected range and
 1298 **MUST** be reported as Normal, or the value is unexpected or out of tolerance for the data
 1299 and a Warning or Fault **MUST** be provided.

1300 Some DataItem types in the EVENT category **MAY** have associated CONDITION states.

1301 Additional CONDITION types are provided to represent the health and fault status of
 1302 *Structural Elements*. Table 43 defines these additional DataItem types.

1303 CONDITION type data items are unlike other data item types since they **MAY** have mul-
 1304 tiple concurrently active values at any point in time.

Table 43: DataItem type for category CONDITION

| DataItem type | Description |
|-----------------|---|
| ACTUATOR | An indication of a fault associated with an actuator. |
| CHUCK_INTERLOCK | An indication of the operational condition of the interlock function for an electronically controller chuck. |
| COMMUNICATIONS | An indication that the piece of equipment has experienced a communications failure. |
| DATA_RANGE | An indication that the value of the data associated with a measured value or a calculation is outside of an expected range. |
| DIRECTION | An indication of a fault associated with the direction of motion of a <i>Structural Element</i> . |
| END_OF_BAR | An indication that the end of a piece of bar stock has been reached. |
| HARDWARE | An indication of a fault associated with the hardware subsystem of the <i>Structural Element</i> . |

| Continuation of Table 43 | |
|--------------------------|--|
| DataItem type | Description |
| INTERFACE_STATE | An indication of the operation condition of an Interface component. |
| LOGIC_PROGRAM | An indication that an error occurred in the logic program or programmable logic controller (PLC) associated with a piece of equipment. |
| MOTION_PROGRAM | An indication that an error occurred in the motion program associated with a piece of equipment. |
| SYSTEM | An indication of a fault associated with a piece of equipment or component that cannot be classified as a specific type. |

1305 9 Sensor

1306 *Sensor* is a unique type of a piece of equipment. A *Sensor* is typically comprised of
 1307 two major components: a *sensor unit* that provides signal processing, conversion, and
 1308 communications and the *sensing elements* that provides a signal or measured value.

1309 The *sensor unit* is modeled as a *Lower Level Component* called *Sensor*. The *sensing*
 1310 *element* may be modeled as a *Composition element* of a *Sensor* element and the mea-
 1311 sured value would be modeled as a *DataItem* (See *Section 8 - Listing of Data Items* for
 1312 more information on *DataItem* elements). Each *sensor unit* may have multiple *sensing*
 1313 *elements*; each representing the data for a variety of measured values.

1314 Example: A pressure transducer could be modeled as a *Sensor* (*Component*) with a
 1315 name = *Pressure Transducer B* and its measured value could be modeled as a *PRESSURE*
 1316 type *DataItem*.

1317 While a *Sensor* may be modeled in the XML document in different ways, it will always be
 1318 modeled to associate the information measured by each *sensor element* with the *Structural*
 1319 *Element* to which the measured value is most closely associated.

1320 9.1 Sensor Data

1321 The most basic implementation of a sensor occurs when the *sensing element* itself is not
 1322 identified in the data model, but the data that is measured by the *sensing element* is pro-
 1323 vided as a data item associated with a *Component*. An example would be the measured
 1324 value of the temperature of a spindle motor. This would be represented as a *DataItem*
 1325 called *TEMPERATURE* that is associated with the *Rotary* type axis element called "C"
 1326 as shown in *Example 8*:

Example 8: Example of Sensing Element provided as data item associated with a Com-
 ponent

```
1327 1 <Components>
1328 2   <Axes
1329 3     <Components>
1330 4       <Rotary id="c" name="C">
1331 5         <DataItems>
1332 6           <DataItem type="TEMPERATURE"
1333 7             id="ctemp" category="SAMPLE"
1334 8               name="Stemp" units="DEGREE"/>
1335 9         </DataItems>
1336 10       </Rotary>
1337 11     </Components>
1338 12   </Axes>
```

1339 13 </Components>

1340 A sensor may measure values associated with any Component or Device element.
 1341 Some examples of how sensor data may be modeled are represented in *Figure 23* :

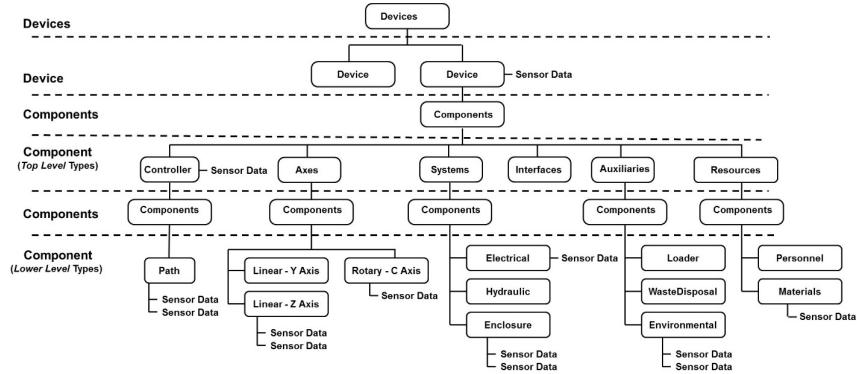


Figure 23: Sensor Data Associations

1342 9.2 Sensor Unit

1343 A *sensor unit* is an intelligent piece of equipment that manages the functions of one or
 1344 more *sensing elements*.

1345 Typical functions of the *sensor unit* include:

- 1346 • convert low level signals from the *sensing elements* into data that can be used by
 1347 other pieces of equipment. (Example: Convert a non-linear millivolt signal from a
 1348 temperature sensor into a scaled temperature value that can be transmitted to another
 1349 piece of equipment.)
- 1350 • process *sensing element* data into calculated values. (Example: temperature sensor
 1351 data is converted into calculated values of average temperature, maximum tempera-
 1352 ture, minimum temperature, etc.)
- 1353 • provide calibration and configuration information associated with each *sensing ele-
 1354 ment*
- 1355 • monitor the health and integrity of the *sensing elements* and the *sensor unit*. (Exam-
 1356 ple: The *sensor unit* may provide diagnostics on each *sensing element* (e.g., open
 1357 wire detection) and itself (e.g., measure internal temperature of the *sensor unit*).

1358 Depending on how the *sensor unit* is used, it may be considered as either an independent
 1359 piece of equipment and modeled in the XML document as a Device, or it may be mod-
 1360 eled as a *Top Level* Component called Sensor if it is integral to a piece of equipment.

1361 A Sensor **MAY** have its own uuid so it can be tracked throughout its lifetime.

1362 The following examples demonstrate how a *Sensor* may be modeled in the XML document
 1363 differently based on how the *Sensor* functions within the overall piece of equipment

1364 Example#1: If the Sensor provides vibration measurement data for the spindle on a
 1365 piece of equipment, it could be modeled as a Sensor for rotary axis named C.

Example 9: Example of Sensor for rotary axis

```

1366 1  <Components>
1367 2    <Axes
1368 3      <Components>
1369 4        <Rotary id="c" name="C">
1370 5          <Components>
1371 6            <Sensor id="spdlm" name="Spindlemonitor">
1372 7              <DataItems>
1373 8                <DataItem type="DISPLACEMENT" id="cvib"
1374 9                  category="SAMPLE" name="Svib"
1375 10                 units="MILLIMETER"/>
1376 11           </DataItems>
1377 12         </Sensor >
1378 13       <Components>
1379 14         </Rotary>
1380 15       </Components>
1381 16     </Axes>
1382 17   </Components>
```

1383 Example#2: If a Sensor provides measurement data for multiple Component elements
 1384 within a piece of equipment and is not associated with any particular Component ele-
 1385 ment, it **MAY** be modeled in the XML document as an independent *Lower Level* Com-
 1386 ponent and the data associated with measurements are associated with their associated
 1387 Component elements.

1388 This example represents a *sensor unit* with two *sensing elements*, one measures spindle
 1389 vibration and the other measures the temperature for the X axis. The *sensor unit* also has
 1390 a *sensing element* measuring the internal temperature of the *sensor unit*.

Example 10: Example of Sensor Unit with Sensing Element

```

1391 1  <Device id="d1" uuid="H1" name="HMC_3Axis">
1392 2    <Description>3 Axis Mill</Description>
1393 3    <Components>
1394 4      <Axes
1395 5        <Components>
```

```

1396 6      <Sensor id="sens1" name="Sensorunit">
1397 7          <DataItems>
1398 8              <DataItem type="TEMPERATURE" id="sentemp"
1399 9                  category="SAMPLE" name="Sensortemp"
1400 10                 units="DEGREE"/>
1401 11          </DataItems>
1402 12      </Sensor >
1403 13      <Rotary id="c" name="C">
1404 14          <DataItems>
1405 15              <DataItem type="DISPLACEMENT" id="cvib"
1406 16                  %category="SAMPLE" name="Svib"
1407 17                  units="MILLIMETER">
1408 18                      <Source componentId="sens1"/>
1409 19          <DataItem/>
1410 20          </DataItems>
1411 21      </Rotary>
1412 22      <Linear id="x" name="X">
1413 23          <DataItems>
1414 24              <DataItem type="TEMPERATURE" id="xt"
1415 25                  category="SAMPLE" name="Xtemp"
1416 26                  units="DEGREE">
1417 27                      <Source componentId="sens1"/>
1418 28          <DataItem/>
1419 29          </DataItems>
1420 30      </Linear>
1421 31      <Components>
1422 32          </Axes>
1423 33      </Components>
1424 34  </Device>

```

1425 9.3 Sensor Configuration

1426 When a Sensor unit is modeled in the XML document as a Component or as a separate
 1427 piece of equipment, it may provide additional configuration information for the *sensor*
 1428 *elements* and the *sensor unit* itself.

1429 Configuration data provides information required for maintenance and support of the
 1430 sensor.

1431 Configuration data is only available when the Sensor unit is modeled as a Com-
 1432 ponent or a separate piece of equipment. For details on the modeling of configuration
 1433 data in the XML document, see *Section 4.4.3.2 - Configuration for Component*.

1434 When Sensor represents the *sensor unit* for multiple *sensing element(s)*, each sensing
 1435 element is represented by a Channel. The *sensor unit* itself and each Channel repre-
 1436 senting one *sensing element* **MAY** have its own configuration data.

1437 SensorConfiguration can contain any descriptive content for a *sensor unit*. This
 1438 element is defined to contain mixed content and additional XML elements (indicated by
 1439 the any element in *Figure 24*) **MAY** be added to extend the schema for SensorCon-
 1440 figuration.

1441 *Figure 24* represents the structure of the SensorConfiguration XML element show-
 1442 ing the attributes defined for SensorConfiguration.

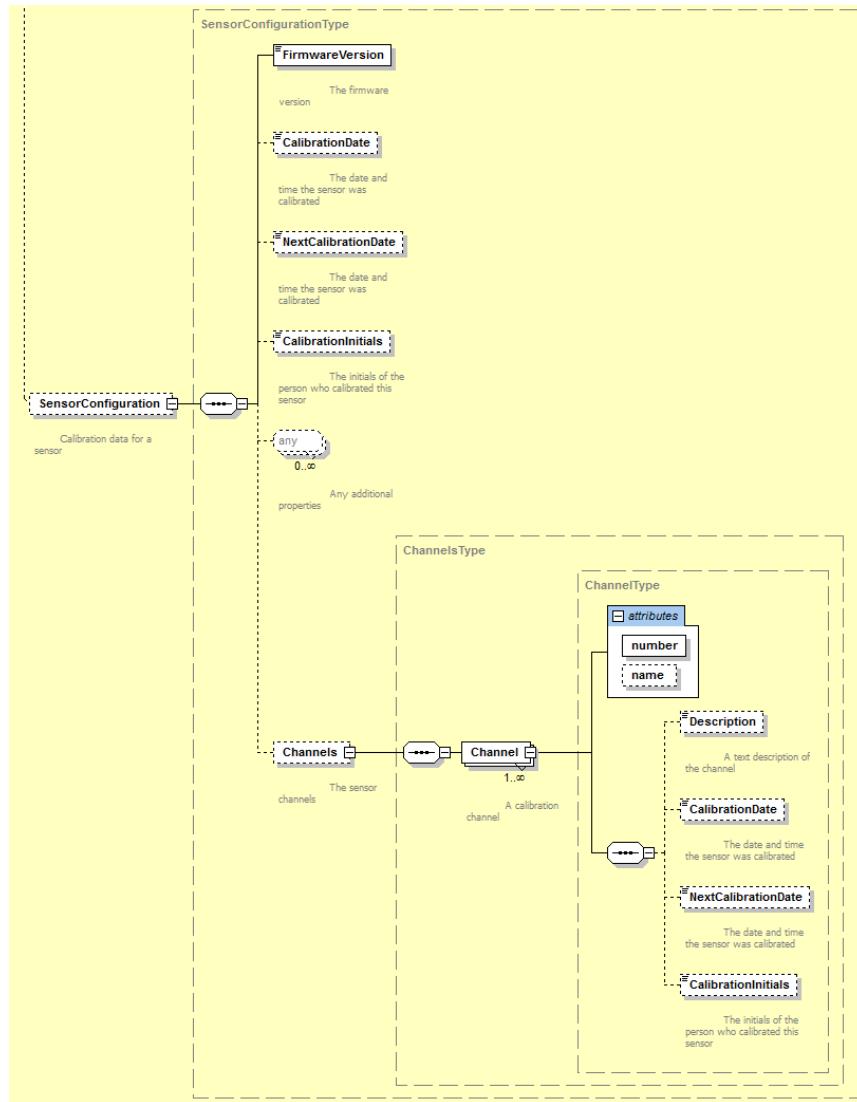


Figure 24: SensorConfiguration Diagram

Table 44: MTConnect SensorConfiguration Element

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

1443 9.3.1 Elements for SensorConfiguration

1444 *Table 45* defines the configuration elements available for SensorConfiguration:

Table 45: Elements for SensorConfiguration

| Element | Description | Occurrence |
|-----------------|---|------------|
| FirmwareVersion | <p>Version number for the sensor unit as specified by the manufacturer.</p> <p>FirmwareVersion is a required element if SensorConfiguration is used.</p> <p>The data value for FirmwareVersion is provided in the CDATA for this element and MAY be any numeric or text content.</p> | 1 |

| Continuation of Table 45 | | |
|--------------------------|--|------------|
| Element | Description | Occurrence |
| CalibrationDate | Date upon which the <i>sensor unit</i> was last calibrated. The data value for CalibrationDate is provided in the CDATA for this element and MUST be represented in the W3C ISO 8601 format. | 0..1 |
| NextCalibrationDate | Date upon which the <i>sensor unit</i> is next scheduled to be calibrated. The data value for NextCalibrationDate is provided in the CDATA for this element and MUST be represented in the W3C ISO 8601 format. | 0..1 |
| CalibrationInitials | The initials of the person verifying the validity of the calibration data. The data value for CalibrationInitials is provided in the CDATA for this element and MAY be any numeric or text content. | 0..1 |
| Channels | When Sensor represents multiple <i>sensing elements</i> , each <i>sensing element</i> is represented by a Channel for the Sensor. Channels is an XML container used to organize information for the <i>sensing elements</i> . | 0..1 |

1445 **9.3.1.1 Attributes for Channel**1446 Channel represents each *sensing element* connected to a *sensor unit*. Table 46 defines
1447 the attributes for Channel:

Table 46: Attributes for Channel

| Attribute | Description | Occurrence |
|-----------|---|------------|
| number | <p>A unique identifier that will only refer to a specific <i>sensing element</i>.</p> <p>number is a required attribute.</p> <p>For example, this can be the manufacturer code and the serial number.</p> <p>number SHOULD be alphanumeric and not exceeding 255 characters.</p> <p>An NMTOKEN XML type.</p> | 1 |
| name | <p>The name of the <i>sensing element</i>.</p> <p>name is an optional attribute.</p> <p>name SHOULD be unique within the <i>sensor unit</i> to allow for easier data integration.</p> <p>An NMTOKEN XML type.</p> | 0..1 |

1448 9.3.1.2 Elements for Channel

1449 Table 47 describes the elements provided for Channel.

Table 47: Elements for Channel

| Element | Description | Occurrence |
|-------------|---|------------|
| Description | <p>An XML element that can contain any descriptive content.</p> <p>The CDATA of Description MAY include any additional descriptive information the implementer chooses to include regarding a <i>sensor element</i>.</p> | 0..1 |

| Continuation of Table 47 | | |
|--------------------------|---|------------|
| Element | Description | Occurrence |
| CalibrationDate | Date upon which the <i>sensor unit</i> was last calibrated to the <i>sensor element</i> . The data value for CalibrationDate is provided in the CDATA for this element and MUST be represented in the W3C ISO 8601 format. | 0..1 |
| NextCalibrationDate | Date upon which the <i>sensor element</i> is next scheduled to be calibrated with the <i>sensor unit</i> . The data value for NextCalibrationDate is provided in the CDATA for this element and MUST be represented in the W3C ISO 8601 format. | 0..1 |
| CalibrationInitials | The initials of the person verifying the validity of the calibration data. The data value for CalibrationInitials is provided in the CDATA for this element and MAY be any numeric or text content. | 0..1 |

1450 Example 11 is an example of the configuration data for Sensor that is modeled as a Component.
 1451 It has Configuration data for the *sensor unit*, one Channel named A/D:1,
 1452 and two DataItems – Voltage (as a SAMPLE) and Voltage (as a CONDITION or
 1453 alarm).

Example 11: Example of configuration data for Sensor

```

1454 1 <Sensor id="sensor" name="sensor">
1455 2   <Configuration>
1456 3     <SensorConfiguration>
1457 4       <FirmwareVersion>2.02</FirmwareVersion>
1458 5       <CalibrationDate>2010-05-16</CalibrationDate>
1459 6       <NextCalibrationDate>2010-05-16</NextCalibrationDate>
1460 7       <CalibrationInitials>WS</CalibrationInitials>
1461 8     <Channels>
1462 9       <Channel number="1" name="A/D:1">
1463 10         <Description>A/D With Thermister</Description>
1464 11       </Channel>
```

```
1465 12      </Channels>
1466 13      </SensorConfiguration>
1467 14      </Configuration>
1468 15      <DataItems>
1469 16          <DataItem category="CONDITION" id="senvc"
1470 17              type="VOLTAGE" />
1471 18          <DataItem category="SAMPLE" id="senv"
1472 19              type="VOLTAGE" units="VOLT" subType="DIRECT" />
1473 20      </DataItems>
1474 21  </Sensor>
```

1475 Appendices

1476 A Bibliography

- 1477 Engineering Industries Association. *EIA Standard - EIA-274-D*, Interchangeable Variable,
1478 Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically
1479 Controlled Machines. Washington, D.C. 1979.
- 1480 ISO TC 184/SC4/WG3 N1089. *ISO/DIS 10303-238*: Industrial automation systems and
1481 integration Product data representation and exchange Part 238: Application Protocols: Ap-
1482 plication interpreted model for computerized numerical controllers. Geneva, Switzerland,
1483 2004.
- 1484 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
1485 tems and integration – Physical device control – Data model for computerized numerical
1486 controllers – Part 10: General process data. Geneva, Switzerland, 2004.
- 1487 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
1488 tems and integration – Physical device control – Data model for computerized numerical
1489 controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.
- 1490 International Organization for Standardization. *ISO 6983/1* – Numerical Control of ma-
1491 chines – Program format and definition of address words – Part 1: Data format for posi-
1492 tioning, line and contouring control systems. Geneva, Switzerland, 1982.
- 1493 Electronic Industries Association. *ANSI/EIA-494-B-1992*, 32 Bit Binary CL (BCL) and
1494 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines.
1495 Washington, D.C. 1992.
- 1496 National Aerospace Standard. *Uniform Cutting Tests* - NAS Series: Metal Cutting Equip-
1497 ment Specifications. Washington, D.C. 1969.
- 1498 International Organization for Standardization. *ISO 10303-11*: 1994, Industrial automa-
1499 tion systems and integration Product data representation and exchange Part 11: Descrip-
1500 tion methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.
- 1501 International Organization for Standardization. *ISO 10303-21*: 1996, Industrial automa-
1502 tion systems and integration – Product data representation and exchange – Part 21: Imple-
1503 mentation methods: Clear text encoding of the exchange structure. Geneva, Switzerland,
1504 1996.
- 1505 H.L. Horton, F.D. Jones, and E. Oberg. *Machinery's Handbook*. Industrial Press, Inc.

- 1506 New York, 1984.
- 1507 International Organization for Standardization. *ISO 841-2001: Industrial automation sys-*
1508 *tems and integration - Numerical control of machines - Coordinate systems and motion*
1509 *nomenclature.* Geneva, Switzerland, 2001.
- 1510 ASME B5.57: *Methods for Performance Evaluation of Computer Numerically Controlled*
1511 *Lathes and Turning Centers,* 1998.
- 1512 ASME/ANSI B5.54: *Methods for Performance Evaluation of Computer Numerically Con-*
1513 *trolled Machining Centers.* 2005.
- 1514 OPC Foundation. *OPC Unified Architecture Specification, Part 1: Concepts Version 1.00.*
1515 July 28, 2006.
- 1516 IEEE STD 1451.0-2007, *Standard for a Smart Transducer Interface for Sensors and Ac-*
1517 *tuators – Common Functions, Communication Protocols, and Transducer Electronic Data*
1518 *Sheet (TEDS) Formats,* IEEE Instrumentation and Measurement Society, TC-9, The In-
1519 *stitute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH99684,*
1520 *October 5, 2007.*
- 1521 IEEE STD 1451.4-1994, Standard for a Smart Transducer Interface for Sensors and Ac-
1522 *tuators – Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet*
1523 *(TEDS) Formats,* IEEE Instrumentation and Measurement Society, TC-9, The Institute of
1524 *Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH95225, December*
1525 *15, 2004.*



MTConnect® Standard

Part 3.0 – Streams Information Model

Version 1.5.0

Prepared for: MTConnect Institute
Prepared on: December 2, 2019

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/>.

MTConnect Specification and Materials

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 | 9 |
| 2.3 MTConnect References | 10 |
| 3 Streams Information Model | 11 |
| 4 Structural Elements for MTConnectStreams | 14 |
| 4.1 Streams | 16 |
| 4.2 DeviceStream | 18 |
| 4.2.1 XML Schema for DeviceStream | 18 |
| 4.2.2 Attributes for DeviceStream | 19 |
| 4.2.3 Elements for DeviceStream | 20 |
| 4.3 ComponentStream | 21 |
| 4.3.1 XML Schema for ComponentStream | 21 |
| 4.3.2 Attributes for ComponentStream | 22 |
| 4.3.3 Elements for ComponentStream | 26 |
| 5 Data Entities | 27 |
| 5.1 Element Names for Data Entities | 29 |
| 5.1.1 Element Names when MTConnectDevices category is SAMPLE or EVENT | 29 |
| 5.1.2 Changes to Element Names when representation attribute is used | 30 |
| 5.1.3 Element Names when MTConnectDevices category is CONDITION | 30 |
| 5.2 Samples Container | 31 |
| 5.3 Sample Data Entities | 31 |
| 5.3.1 XML Schema Structure for Sample | 32 |
| 5.3.2 Attributes for Sample | 33 |
| 5.3.2.1 duration Attribute for Sample | 37 |
| 5.3.2.2 resetTriggered Attribute for Sample | 37 |
| 5.3.3 Response for SAMPLE category DataItem Elements with a representation Attribute of TIME_SERIES | 38 |
| 5.3.3.1 XML Schema Structure for Sample when reporting Time Series Data | 39 |
| 5.3.3.2 Attributes for a Sample when reporting Time Series Data | 40 |
| 5.3.4 Response for SAMPLE category DataItem Elements with a representation attribute of DATA_SET | 40 |
| 5.3.4.1 XML Schema Structure for Sample when reporting Data Set data | 41 |

| | | |
|-------------------|--|------------|
| 5.3.4.2 | Attributes for Sample when reporting Data Set data | 41 |
| 5.3.4.3 | Elements for Sample when reporting Data Set data | 42 |
| 5.3.4.3.1 | XML Schema Structure for Entry Element for a Data Entity | 42 |
| 5.3.4.3.2 | Attributes for Entry Element for a Data Entity | 43 |
| 5.3.5 | Valid Data Values for Sample | 44 |
| 5.3.6 | Unavailability of Valid Data Values for Sample | 46 |
| 5.4 | Events Container | 46 |
| 5.5 | Event Data Entities | 47 |
| 5.5.1 | XML Schema Structure for Event | 48 |
| 5.5.2 | Attributes for Event | 49 |
| 5.5.3 | Response for EVENT category DataItem Elements with a repre- sentation attribute of DATA_SET | 50 |
| 5.5.4 | Valid Data Values for Event | 51 |
| 5.5.5 | Unavailability of Valid Data Value for Event | 52 |
| 5.6 | Condition Container | 52 |
| 5.7 | Condition Data Entity | 53 |
| 5.7.1 | Element Names for Condition | 54 |
| 5.7.2 | XML Schema Structure for Condition | 55 |
| 5.7.3 | Attributes for Condition | 56 |
| 5.7.3.1 | qualifier Attribute for Condition | 59 |
| 5.7.4 | Valid Data Value for Condition | 60 |
| 5.8 | Unavailability of Fault State for Condition | 60 |
| 6 | Listing of Data Entities | 62 |
| 6.1 | Sample Element Names | 62 |
| 6.2 | Event Element Names | 83 |
| 6.3 | Types of Condition Elements | 123 |
| Appendices | | 125 |
| A | Bibliography | 125 |

Table of Figures

| | |
|---|----|
| Figure 1: Streams Data Structure | 15 |
| Figure 2: Streams Schema Diagram | 17 |
| Figure 3: DeviceStream Schema Diagram | 19 |
| Figure 4: ComponentStream Schema Diagram | 22 |
| Figure 5: ComponentStream XML Tree Diagram | 27 |
| Figure 6: Sample Schema Diagram | 33 |
| Figure 7: AbsTimeSeries Schema Diagram | 39 |
| Figure 8: Sample Data Set Schema Diagram | 41 |
| Figure 9: Entry Element Schema Diagram | 43 |
| Figure 10: Event Schema Diagram | 48 |
| Figure 11: Condition Schema Diagram | 55 |

List of Tables

| | |
|--|-----|
| Table 1: MTConnect Streams Element | 17 |
| Table 2: MTConnect DeviceStream Element | 18 |
| Table 3: Attributes for DeviceStream | 19 |
| Table 4: Elements for DeviceStream | 20 |
| Table 5: Attributes for ComponentStream | 23 |
| Table 6: Elements for ComponentStream | 26 |
| Table 7: MTConnect Samples Element | 31 |
| Table 8: MTConnect Sample Element | 32 |
| Table 9: Attributes for Sample | 33 |
| Table 10: Values for resetTriggered | 38 |
| Table 11: MTConnect sampleCount Attribute | 40 |
| Table 12: Attributes for DataSet | 42 |
| Table 13: Elements for DataSet | 42 |
| Table 14: Attributes for Entry | 44 |
| Table 15: MTConnect Event Element | 47 |
| Table 16: MTConnect Event Element | 48 |
| Table 17: Attributes for Event | 49 |
| Table 18: MTConnect Condition Element Container | 53 |
| Table 19: MTConnect Condition Element | 54 |
| Table 20: Attributes for Condition | 56 |
| Table 21: Element Names for Sample | 62 |
| Table 22: Element Names for Event | 84 |
| Table 23: Element Names for Condition | 124 |

1 Purpose of This Document

2 This document, *MTConnect Standard: Part 3.0 - Streams Information Model* of the MT-
3 Connect Standard, establishes the rules and terminology that describes the information
4 returned by an MTConnect Agent from a piece of equipment. The *Streams Information*
5 *Model* also defines, in *Section 3 - Streams Information Model*, the structure for the XML
6 documents that are returned from an Agent in response to a *Sample Request* or *Current*
7 *Request*.

8 *MTConnect Standard: Part 3.0 - Streams Information Model* is not a stand-alone docu-
9 ment. This document is used in conjunction with *MTConnect Standard Part 1.0 - Overview*
10 and *Fundamentals* which defines the fundamentals of the operation of the MTConnect
11 Standard and *MTConnect Standard: Part 2.0 - Devices Information Model* that defines
12 the semantic model representing the information that may be returned from a piece of
13 equipment.

14 Note: *MTConnect Standard: Part 5.0 - Interfaces* provides details on extensions to
15 the *Streams Information Model* required to describe the interactions between pieces of
16 equipment.

17 In the MTConnect Standard, equipment represents any tangible property that is used in the
18 operation of a manufacturing facility. Examples of equipment are machine tools, ovens,
19 sensor units, workstations, software applications, and bar feeders.

20 2 Terminology and Conventions

21 Refer to *Section 3 of MTConnect Standard Part 1.0 - Overview and Fundamentals* for a
 22 dictionary of terms, reserved language, and document conventions used in the MTConnect
 23 Standard.

24 2.1 Glossary

25 CDATA

26 General meaning:

27 An abbreviation for Character Data.

28 CDATA is used to describe a value (text or data) published as part of an XML ele-
 29 ment.

30 For example, "This is some text" is the CDATA in the XML element:

31 <Message ...>This is some text</Message>

32 Appears in the documents in the following form: CDATA

33 HTTP

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

36 Note: HTTP is an IETF standard and is defined in RFC 7230.

37 See <https://tools.ietf.org/html/rfc7230> for more information.

38 NMTOKEN

39 The data type for XML identifiers.

40 Note: The identifier must start with a letter, an underscore "_" or a colon. The next
 41 character must be a letter, a number, or one of the following ".", "-", "_", ":". The
 42 identifier must not have any spaces or special characters.

43 Appears in the documents in the following form: NMTOKEN.

44 XML

45 Stands for eXtensible Markup Language.

46 XML defines a set of rules for encoding documents that both a human-readable and
 47 machine-readable.

48 XML is the language used for all code examples in the MTConnect Standard.

49 Refer to <http://www.w3.org/XML> for more information about XML.

50 **Agent**

51 Refers to an MTConnect Agent.

52 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
53 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.
54
55

56 Appears in the documents in the following form: *Agent*.

57 **Asset Document**

58 An electronic document published by an *Agent* in response to a *Request* for information from a client software application relating to Assets.
59

60 **Child Element**

61 A portion of a data modeling structure that illustrates the relationship between an
62 element and the higher-level *Parent Element* within which it is contained.

63 Appears in the documents in the following form: *Child Element*.

64 **Component**

65 General meaning:

66 A *Structural Element* that represents a physical or logical part or subpart of a piece
67 of equipment.

68 Appears in the documents in the following form: *Component*.

69 Used in Information Models:

70 A data modeling element used to organize the data being retrieved from a piece of
71 equipment.

- 72 • When used as an XML container to organize *Lower Level Component* elements.
73

74 Appears in the documents in the following form: Components.

- 75 • When used as an abstract XML element. *Component* is replaced in a data
76 model by a type of *Component* element. *Component* is also an XML con-
77 tainer used to organize *Lower Level Component* elements, *Data Entities*, or
78 both.

79 Appears in the documents in the following form: Component.

80 **Condition**

81 General meaning:

82 An indicator of the health of a piece of equipment or a *Component* and its ability to
83 function.

84 Used as a modeling element:

85 A data modeling element used to organize and communicate information relative to
86 the health of a piece of equipment or *Component*.

87 Appears in the documents in the following form: *Condition*.

88 Used in *Information Models*:

89 An XML element used to represent *Condition* elements.

- 90 ● When used as an XML container to organize *Lower Level* Condition ele-
91 ments.

92 Appears in the documents in the following form: *Condition*.

- 93 ● When used as a *Lower Level* element, the form *Condition* is an abstract
94 type XML element. This *Lower Level* element is a *Data Entity*. *Condition*
95 is replaced in a data model by type of *Condition* element.

96 Appears in the documents in the following form: *Condition*.

97 Note: The form *Condition* is used to represent both above uses.

98 ***Controlled Vocabulary***

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

101 Appears in the documents in the following form: *Controlled Vocabulary*.

102 ***Current Request***

103 An HTTP request to the *Agent* for returning latest known values for the *DataItem*
104 as an *MTConnectStreams* XML document

105 ***Data Entity***

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

109 Appears in the documents in the following form: *Data Entity*.

110 ***Data Set***

111 A set of *key-value pairs* where each entry is uniquely identified by the *key*.

112 ***Devices Information Model***

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

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

116 ***Document***

117 General meaning:

118 A piece of written, printed, or electronic matter that provides information.

119 Used to represent an *MTConnect Document*:

120 Refers to printed or electronic document(s) that represent a *Part(s)* of the MTCon-
121 nect Standard.

122 Appears in the documents in the following form: *MTConnect Document*.

123 Used to represent a specific representation of an *MTConnect Document*:

124 Refers to electronic document(s) associated with an *Agent* that are encoded using
125 XML; *Response Documents* or *Asset Documents*.

126 Appears in the documents in the following form: *MTConnect XML Document*.

127 Used to describe types of information stored in an *Agent*:

128 In an implementation, the electronic documents that are published from a data source
129 and stored by an *Agent*.

130 Appears in the documents in the following form: *Asset Document*.

131 Used to describe information published by an *Agent*:

132 A document published by an *Agent* based upon one of the *semantic data models*
133 defined in the MTConnect Standard in response to a request from a client.

134 Appears in the documents in the following form: *Response Document*.

135 ***Element Name***

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

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

139 Used to describe the name for a specific XML element:

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

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

143 ***Equipment Metadata***

144 See *Metadata*

145 **Fault State**

146 In the MTConnect Standard, a term that indicates the reported status of a *Condition*
147 category *Data Entity*.

148 Appears in the documents in the following form: *Fault State*.

149 **Information Model**

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

152 For example, an information model is used to define the structure for each *MTCon-*
153 *nnect Response Document*; the definition of each piece of information within those
154 documents and the relationship between pieces of information.

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

156 **Interaction Model**

157 The definition of information exchanged to support the interactions between pieces
158 of equipment collaborating to complete a task.

159 Appears in the documents in the following form: *Interaction Model*.

160 **Interface**

161 General meaning:

162 The exchange of information between pieces of equipment and/or software systems.

163 Appears in the documents in the following form: interface.

164 Used as an Interaction Model:

165 An *Interaction Model* that describes a method for inter-operations between pieces
166 of equipment.

167 Appears in the documents in the following form: *Interface*.

168 Used as an XML container or element:

169 - When used as an XML container that consists of one or more types of Interface
170 XML elements.

171 Appears in the documents in the following form: Interfaces.

172 - When used as an abstract XML element. It is replaced in the XML document
173 by types of Interface elements.

174 Appears in the documents in the following form: Interface

175 **key**

176 A unique identifier in a *key-value pair* association.

177 **key-value pair**

178 An association between an identifier referred to as the *key* and a value which taken
179 together create a *key-value pair*. When used in a set of *key-value pairs* each *key* is
180 unique and will only have one value associated with it at any point in time.

181 **Lower Level**

182 A nested element that is below a higher level element.

183 **Metadata**

184 Data that provides information about other data.

185 For example, *Equipment Metadata* defines both the *Structural Elements* that rep-
186 resent the physical and logical parts and sub-parts of each piece of equipment, the
187 relationships between those parts and sub-parts, and the definitions of the *Data En-*
188 *tities* associated with that piece of equipment.

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

190 **MTConnect Document**

191 See *Document*.

192 **MTConnect XML Document**

193 See *Document*.

194 **Parent Element**

195 An XML element used to organize *Lower Level* child elements that share a common
196 relationship to the *Parent Element*.

197 Appears in the documents in the following form: *Parent Element*.

198 **Request**

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

201 Appears in the documents in the following form: *Request*.

202 **Response Document**

203 See *Document*.

204 **Sample Request**

205 A request from the *Agent* for a stream of time series data.

206 ***semantic data model***

207 A methodology for defining the structure and meaning for data in a specific logical
208 way.

209 It provides the rules for encoding electronic information such that it can be inter-
210 preted by a software system.

211 Appears in the documents in the following form: *semantic data model*.

212 ***sequence number***

213 The primary key identifier used to manage and locate a specific piece of *Streaming*
214 *Data* in an *Agent*.

215 *sequence number* is a monotonically increasing number within an instance of an
216 *Agent*.

217 Appears in the documents in the following form: *sequence number*.

218 ***Streaming Data***

219 The values published by a piece of equipment for the *Data Entities* defined by the
220 *Equipment Metadata*.

221 Appears in the documents in the following form: *Streaming Data*.

222 ***Streams Information Model***

223 The rules and terminology (*semantic data model*) that describes the *Streaming Data*
224 returned by an *Agent* from a piece of equipment in response to a *Sample Request* or
225 a *Current Request*.

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

227 ***Structural Element***

228 General meaning:

229 An XML element that organizes information that represents the physical and logical
230 parts and sub-parts of a piece of equipment.

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

232 Used to indicate hierarchy of Components:

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

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

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

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

239 **Top Level**

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

242 **Valid Data Value**

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

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

246 2.2 Acronyms

247 **AMT**

248 The Association for Manufacturing Technology

249 2.3 MTConnect References

250 [MTConnect Part 1.0] *MTConnect Standard Part 1.0 - Overview and Fundamentals*. Ver-
251 sion 1.5.0.

252 [MTConnect Part 2.0] *MTConnect Standard: Part 2.0 - Devices Information Model*. Ver-
253 sion 1.5.0.

254 [MTConnect Part 3.0] *MTConnect Standard: Part 3.0 - Streams Information Model*. Ver-
255 sion 1.5.0.

256 [MTConnect Part 5.0] *MTConnect Standard: Part 5.0 - Interfaces*. Version 1.5.0.

²⁵⁷ 3 Streams Information Model

258 The *Streams Information Model* provides a representation of the data reported by a piece
 259 of equipment used for a manufacturing process, or used for any other purpose. Additional
 260 descriptive information associated with the reported data is defined in the `MTConnect-`
 261 `Devices` document, which is described in *MTConnect Standard: Part 2.0 - Devices*
 262 *Information Model*.

263 Information defined in the *Streams Information Model* allows a software application to (1)
 264 determine the value for *Data Entities* returned from a piece of equipment and (2) interpret
 265 the data associated with those *Data Entities* with the same meaning, value, and context
 266 that it had at its original source. To do this, the software application issues one of two
 267 HTTP requests to an *Agent* associated with a piece of equipment. They are:

- 268 • `sample`: Returns a designated number of time stamped *Data Entities* from an *Agent*
 269 associated with a piece of equipment; subject to any HTTP filtering associated with
 270 the request. See *Section 8.3.3 of MTConnect Standard Part 1.0 - Overview and Fun-*
 271 *damentals* of the MTConnect Standard for details on the `sample` HTTP request.
- 272 • `current`: Returns a snapshot of either the most recent values or the values at a
 273 given sequence number for all *Data Entities* associated with a piece of equipment
 274 from an *Agent*; subject to any HTTP filtering associated with the request. See *Sec-*
 275 *tion 8.3.2 of MTConnect Standard Part 1.0 - Overview and Fundamentals* of the
 276 MTConnect Standard for details on the `current` HTTP request.

277 An *Agent* responds to either the `sample` or `current` HTTP request with an
 278 `MTConnectStreams` XML document. This document contains information describing
 279 *Data Entities* reported by an *Agent* associated with a piece of equipment. A client software
 280 application may correlate the information provided in the `MTConnectStreams` XML
 281 document with the physical and logical structure for that piece of equipment defined in the
 282 `MTConnectDevices` document to form a clear and unambiguous understanding of the
 283 information provided. (See details on the structure for a piece of equipment described in
 284 *MTConnect Standard: Part 2.0 - Devices Information Model*).

285 The `MTConnectStreams` XML document is comprised of two sections: `Header` and
 286 `Streams`.

287 The `Header` section contains protocol related information as defined in *Section 6.5 of*
 288 *MTConnect Standard Part 1.0 - Overview and Fundamentals* of the MTConnect Standard.

289 The `Streams` section of the `MTConnectStreams` document contains a
 290 `DeviceStream` XML container for each piece of equipment represented in the docu-

291 ment. Each `DeviceStream` container is comprised of two primary types of XML ele-
 292 ments – *Structural Elements* and *Data Entities*. The contents of the `DeviceStream` con-
 293 tainer are described in detail in this document, *MTConnect Standard: Part 3.0 - Streams*
 294 *Information Model* of the MTConnect Standard.

295 *Structural Elements* are defined for both the `MTConnectDevices` and the `MTCon-
 296 nnectStreams XML documents. These Structural Elements are used to provide a logi-
 297 cal organization of the information provided in each document. While used for a similar
 298 purpose, the Structural Elements in the MTConnectStreams document are specifically
 299 designed to be distinctly different from those in the MTConnectDevices document:`

300 • `MTConnectDevices` document: *Structural Elements* organize information that
 301 represents the physical and logical parts and sub-parts of a piece of equipment. (See
 302 *MTConnect Standard: Part 2.0 - Devices Information Model*, Section 4 of the MT-
 303 Connect Standard for more details on *Structural Elements* used in the `MTConnect-
 304 Devices document).`

305 • `MTConnectStreams` document: *Structural Elements* provide the structure to or-
 306 ganize the data returned from a piece of equipment and establishes the proper context
 307 for that data. The *Structural Elements* specifically defined for use in the `MTCon-
 308 nnectStreams document are DeviceStream (see Section 4.2 - DeviceStream)
 309 and ComponentStream (see Section 4.3 - ComponentStream).`

310 `DeviceStream` and `ComponentStream` elements have a direct correlation to
 311 each of the *Structural Elements* defined in the `MTConnectDevices` document.

312 *Data Entities* that describe data reported by a piece of equipment are also defined for both
 313 the `MTConnectDevices` and the `MTConnectStreams` XML documents. The *Data*
 314 *Entities* provided in both documents directly relate to each other. However, *Data Entities*
 315 are used for different purposes in each document:

316 • `MTConnectDevices` document: *Data Entity* elements define the data that may
 317 be returned from a piece of equipment. *MTConnect Standard: Part 2.0 - Devices*
 318 *Information Model*, *Sections 7 and 8* lists the possible *Data Entity* XML elements
 319 that can be returned in a `MTConnectDevices` document.

320 • `MTConnectStreams` document: *Data Entity* elements provide the data reported
 321 by a piece of equipment. This data is organized in separate `ComponentStream`
 322 XML containers for each of the *Structural Elements* defined in the `MTConnectDe-
 323 vices document associated with the data that is reported by a piece of equipment.`

324 Within each ComponentStream XML container in the MTConnectStreams docu-
325 ment, *Data Entities* are organized into three types of XML container elements - Samples,
326 Events, and Conditions. (See *Section 5 - Data Entities* and *Section 6 - Listing of*
327 *Data Entities* for more information on these elements.)

328 4 Structural Elements for MTConnectStreams

329 *Structural Elements* are XML elements that form the logical structure for the MTCon-
330 nectionStreams XML document. These elements are used to organize the information
331 and data that is reported by an *Agent* for a piece of equipment. See *Figure 1* for an
332 overview of the *Structural Elements* used in an MTConnectStreams document.

333 The first, or highest level, *Structural Element* in an MTConnectStreams XML docu-
334 ment is Streams. Streams is a container type XML element used to group the data
335 reported from one or more pieces of equipment into a single XML document. Streams
336 **MUST** always appear in the MTConnectStreams document.

337 DeviceStream is the next *Structural Element* in the MTConnectStreams document.
338 DeviceStream is also a XML container type element. A separate DeviceStream
339 container is used to organize the information and data reported by each piece of equip-
340 ment represented in the MTConnectStreams document. There **MUST** be at least one
341 DeviceStream element in the Streams container.

342 A DeviceStream element provides the data reported by a piece of equipment. Each
343 DeviceStream element **MUST** contain the attributes name and uid to correlate the
344 DeviceStream with a specific Device defined in the MTConnectDevices docu-
345 ment. Once the DeviceStream element is associated with a specific piece of equipment
346 based on this identity, all data reported by that piece of equipment is directly associated
347 with that unique identity and that association does not need to be repeated for every piece
348 of data reported. A client software application may then directly relate the information
349 provided in the MTConnectDevices document with the data provided in the MTCon-
350 nectionStreams document based on this identity.

351 ComponentStream is the next level XML element in the MTConnectStreams docu-
352 ment. ComponentStream is also a container type XML element. There **MUST** be
353 a separate ComponentStream XML element for each of the *Structural Elements* (De-
354 vice elements, *Top Level* Component elements, or *Lower Level* Component elements)
355 defined for that piece of equipment in the associated MTConnectDevices XML docu-
356 ment. A ComponentStream representing a *Structural Element* will only appear if there
357 is data reported for that *Structural Element*. (Note: See *MTConnect Standard: Part 2.0 -*
358 *Devices Information Model* of the MTConnect Standard for a description of the *Structural*
359 *Elements* for a piece of equipment).

360 There are three (3) *Structural Elements* – Samples, Events, and Condition at the
361 next level of the MTConnectStreams document. Each one of these *Structural Elements*
362 is a container type XML element. These *Structural Elements* group the data reported for
363 each component of a piece of equipment according to the *Data Entity* categories defined

364 in *MTConnect Standard: Part 2.0 - Devices Information Model*, Sections 7 and 8.

- 365 • Samples contains SAMPLE category *Data Entities* defined in the MTConnect–
366 Devices XML document (See *MTConnect Standard: Part 2.0 - Devices Infor-*
367 *mation Model*, Section 8.1)
- 368 • Events contains EVENT category *Data Entities* defined in the MTConnectDe-
369 vices XML document (See *MTConnect Standard: Part 2.0 - Devices Infor-*
370 *mation Model*, Section 8.2)
- 371 • Condition contains CONDITION category *Data Entities* defined in the MTCon-
372 nectDevices XML document (See *MTConnect Standard: Part 2.0 - Devices*
373 *Information Model*, Section 8.3)

374 There **MUST** be at least one of Samples, Events, or Condition elements in each
375 ComponentStream container.

376 *Figure 1* XML tree structure illustrates the various *Structural Elements* used to organize
377 the data reported by a piece of equipment and the relationship between these elements.

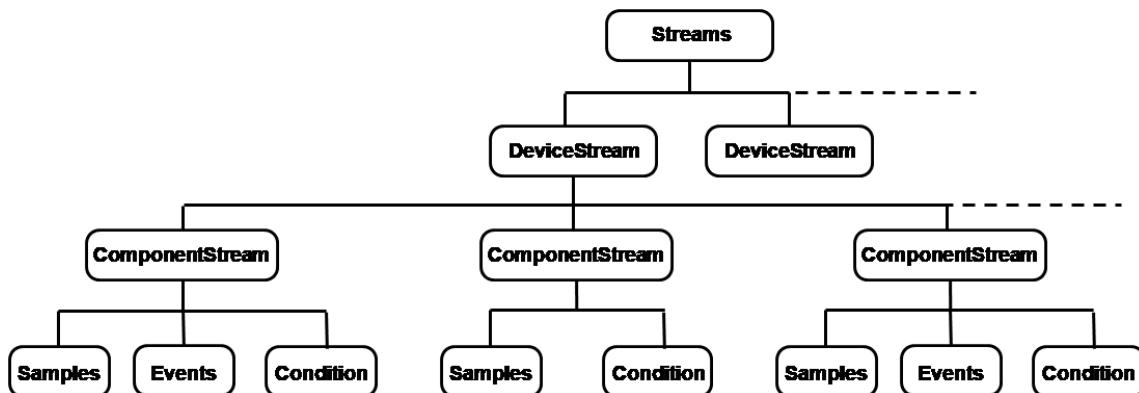


Figure 1: Streams Data Structure

378 *Example 1* is a sample from an MTConnectStreams XML document that contains the
379 response from an Agent representing two pieces of equipment, *mill-1* and *mill-2*. The data
380 from each piece of equipment is reported in a separate DeviceStream container.

Example 1: Example of DeviceStream

```

381 1 <MTConnectStreams ...>
382 2   <Header ... />
383 3   <Streams>
384 4     <DeviceStream name="mill-1" uuid="1">
385 5       <ComponentStream component="Device" name="mill-1"
  
```

```

386 6      componentId="d1">
387 7      <Events>
388 8          <Availability dataItemId="avail1" name="avail"
389 9              sequence="5"
390 10             timestamp="2010-04-06T06:19:35.153141">
391 11             AVAILABLE</Availability>
392 12         </Events>
393 13     </ComponentStream>
394 14 </DeviceStream>
395 15 <DeviceStream name="mill-2" uuid="2">
396 16     <ComponentStream component="Device" name="mill-2"
397 17         componentId="d2">
398 18         <Events>
399 19             <Availability dataItemId="avail2" name="avail"
400 20                 sequence="15"
401 21                 timestamp="2010-04-06T06:19:35.153141">
402 22                 AVAILABLE</Availability>
403 23             </Events>
404 24         </ComponentStream>
405 25     </DeviceStream>
406 26 </Streams>
407 27 </MTConnectStreams>

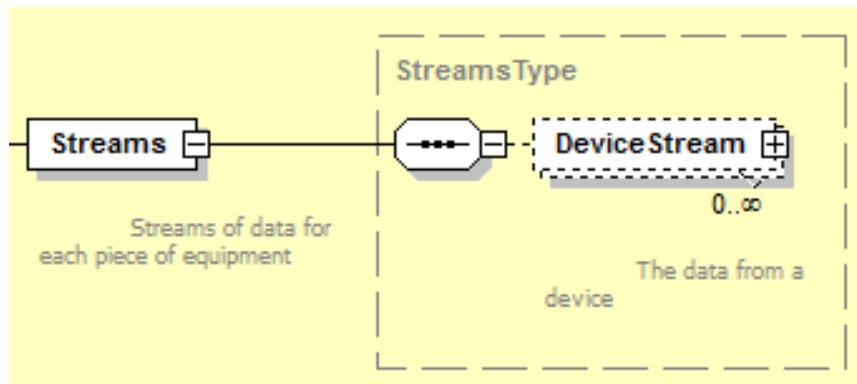
```

408 In *Example 1*, it should be noted that the *sequence numbers* are unique across the two
 409 pieces of equipment. Client software applications **MUST NOT** assume that the *Events*
 410 and *Samples* sequence numbers are strictly in sequence. All sequence numbers **MAY**
 411 **NOT** be included. For instance, such a case would occur when HTTP filtering is applied to
 412 the request and the SAMPLE, EVENT, and CONDITION data types for other components
 413 are not returned. Another case would occur when an *Agent* is supporting more than one
 414 piece of equipment and data from only one piece of equipment is requested. Refer to MT-
 415 Connect Standard *MTConnect Standard Part 1.0 - Overview and Fundamentals, Section 5*
 416 for more information on *sequence numbers*.

417 4.1 Streams

418 Streams is a container type XML element that **MUST** contain only DeviceStream
 419 elements. Streams **MAY** contain any number of DeviceStream elements. If there is
 420 no data to be reported for a request for data, an MTConnectStreams document **MUST**
 421 be returned with an empty Streams container. *Data Entities* **MAY NOT** be directly
 422 associated with the Streams container.

423 The XML schema in *Figure 2* represents the structure of the Streams XML element.

**Figure 2:** Streams Schema Diagram**Table 1:** MTConnect Streams Element

| Element | Description | Occurrence |
|---------|--|------------|
| Streams | <p>The first, or highest, level XML container element in an MTConnectStreams <i>Response</i> Document provided by an <i>Agent</i> in response to a sample or current HTTP <i>Request</i>.</p> <p>There MAY be only one Streams element in an MTConnectStreams <i>Response</i> Document for each piece of equipment represented in the document.</p> <p>An empty Streams container MAY be provided to indicate that no data is available for the given <i>Request</i>.</p> <p>The Streams element MAY contain any number of DeviceStream elements, one for each piece of equipment represented in the MTConnectStreams document.</p> | 1 |

424 4.2 DeviceStream

425 DeviceStream is a XML container that organizes data reported from a single piece of
 426 equipment. A DeviceStream element **MUST** be provided for each piece of equipment
 427 reporting data in an MTConnectStreams document.

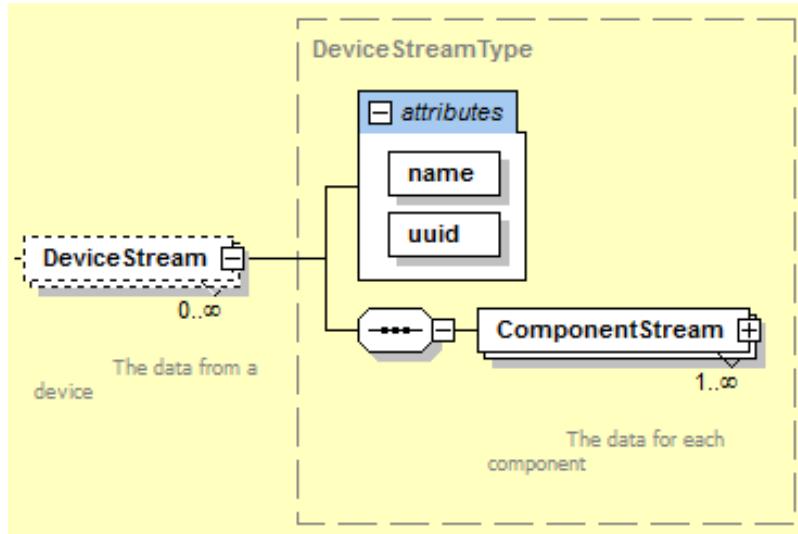
428 A DeviceStream **MAY** contain any number of ComponentStream elements; lim-
 429 ited to one for each component element represented in the MTConnectDevices doc-
 430 ument. If the response to the request for data from an *Agent* does not contain any data
 431 for a specific piece of equipment, an empty DeviceStream element **MAY** be created to
 432 indicate that the piece of equipment exists, but there was no data available. In this case,
 433 there will be no ComponentStream elements provided.

Table 2: MTConnect DeviceStream Element

| Element | Description | Occurrence |
|--------------|---|------------|
| DeviceStream | An XML container element provided in the Streams container in the MTConnectStreams document. There MAY be one or more DeviceStream elements in a Streams container; one for each piece of equipment represented in the MTConnectStreams document. | 0..* |

434 4.2.1 XML Schema for DeviceStream

435 The XML schema in *Figure 3* represents the structure of the DeviceStream XML
 436 element showing the attributes defined for DeviceStream and the elements that **MAY**
 437 be associated with DeviceStream.

**Figure 3:** DeviceStream Schema Diagram

438 4.2.2 Attributes for DeviceStream

439 *Table 3* defines the attributes that **MUST** be provided to uniquely identify each specific
 440 piece of equipment associated with the information provided in each DeviceStream.

Table 3: Attributes for DeviceStream

| Attribute | Description | Occurrence |
|-----------|--|------------|
| name | <p>The name of an element or a piece of equipment. The name associated with the piece of equipment reporting the data contained in this DeviceStream container.</p> <p>name is a required attribute.</p> <p>The value reported for name MUST be the same as the value defined for the name attribute of the same piece of equipment in the MTConnectDevices document</p> <p>An NMTOKEN XML type.</p> <p>WARNING: name may become an optional attribute in future versions of the MTConnect Standard.</p> | 1 |

| Continuation of Table 3 | | |
|-------------------------|--|------------|
| Attribute | Description | Occurrence |
| uuid | <p>The <code>uuid</code> associated with the piece of equipment reporting the data contained in this <code>DeviceStream</code> container.</p> <p><code>uuid</code> is a required attribute.</p> <p>The value reported for <code>uuid</code> MUST be the same as the value defined for the <code>uuid</code> attribute of the same piece of equipment in the <code>MTConnectDevices</code> document.</p> | 1 |

441 4.2.3 Elements for DeviceStream

442 *Table 4* lists the XML element(s) that **MAY** be provided in the `DeviceStream` XML
 443 element.

Table 4: Elements for DeviceStream

| Element | Description | Occurrence |
|-----------------|---|------------|
| ComponentStream | <p>An XML container type element that organizes data returned from an <i>Agent</i> in response to a current or sample HTTP request.</p> <p>Any number of <code>ComponentStream</code> elements MAY be provided in a <code>DeviceStream</code> container.</p> <p>There MUST be a separate <code>ComponentStream</code> XML element for each of the <i>Structural Elements</i> (<code>Device</code> elements, <i>Top Level Component</i> elements, or <i>Lower Level Component</i> elements) defined for that piece of equipment in the associated <code>MTConnectDevices</code> XML document. A <code>ComponentStream</code> representing a <i>Structural Element</i> will only appear if there is data reported for that <i>Structural Element</i>.</p> | 0..* |

444 4.3 ComponentStream

445 ComponentStream is a XML container that organizes the data associated with each
446 *Structural Element* (Device element, *Top Level Component*, or *Lower Level Com-*
447 *ponent element*) defined for that piece of equipment in the associated MTConnectDe-
448 vices XML document. The data reported in each ComponentStream element **MUST**
449 be grouped into individual XML containers based on the value of the category attribute
450 (SAMPLE, EVENT, or CONDITION) defined for each *Data Entity* in the MTConnect-
451 Devices XML document. These containers are Samples, Events, and Condition.

452 4.3.1 XML Schema for ComponentStream

453 The XML schema in *Figure 4* represents the structure of a ComponentStream XML
454 element showing the attributes defined for ComponentStream and the elements that
455 **MAY** be associated with ComponentStream.

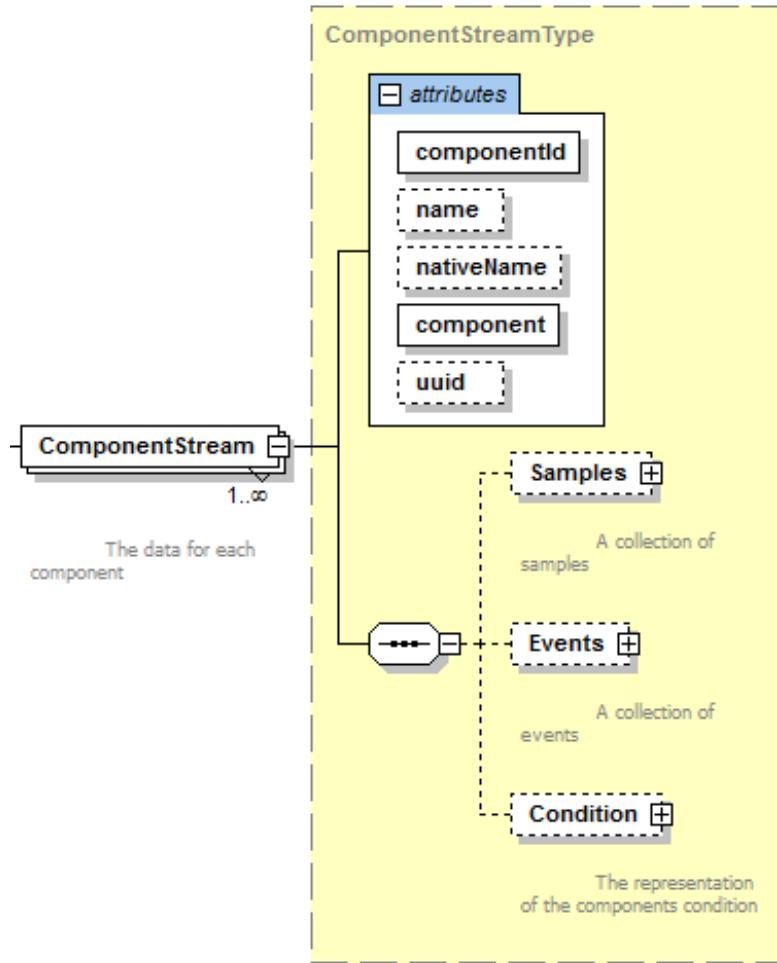


Figure 4: ComponentStream Schema Diagram

456 ComponentStream is similar to DeviceStream in that the attributes uniquely iden-
 457 tify the *Structural Element* with which the data reported is directly associated. This infor-
 458 mation does not have to be repeated for each *Data Entity*. In the case of the DeviceS-
 459 tream, the attributes uniquely identify the piece of equipment associated with the data.
 460 In the case of the ComponentStream, the attributes identify the specific *Structural El-*
 461 *ement* within a piece of equipment associated with each *Data Entity*.

462 4.3.2 Attributes for ComponentStream

463 The *Table 5* defines the attributes used to uniquely identify the specific *Structural Ele-*
 464 *ment(s)* of a piece of equipment associated with the data reported in the MTConnect-
 465 Streams document.

Table 5: Attributes for ComponentStream

| Attribute | Description | Occurrence |
|-------------|---|------------|
| componentId | <p>The identifier of the <i>Structural Element</i> (Device element, <i>Top Level Component</i> element, or <i>Lower Level Component</i> element) as defined by the <i>id</i> attribute of the corresponding <i>Structural Element</i> in the MTConnectDevices XML document.</p> <p>componentId is a required attribute.</p> <p>The identifier MUST be the same as that defined in the MTConnectDevices document to associate the data reported in the ComponentStream container with the <i>Structural Element</i> identified in the MTConnectDevices document.</p> | 1 |
| name | <p>The name of the ComponentStream element.</p> <p>name is an optional attribute.</p> <p>If name is not defined for a specific <i>Structural Element</i> in the MTConnectDevices document, it MUST NOT be provided for the corresponding ComponentStream element in the MTConnectStreams document.</p> <p>If name is defined for a specific <i>Structural Element</i> in the MTConnectDevices document, it MAY be provided for the corresponding ComponentStream element in the MTConnectStreams document.</p> <p>If provided, the value reported for name MUST be the same as the value defined for the name attribute of the corresponding <i>Structural Element</i> (Device element, <i>Top Level Component</i> element, or <i>Lower Level Component</i> element) defined in the MTConnectDevices XML document.</p> <p>An NMTOKEN XML type.</p> | 0..1 |

| Continuation of Table 5 | | |
|-------------------------|--|------------|
| Attribute | Description | Occurrence |
| nativeName | <p>nativeName identifies the common name normally associated with the ComponentStream element.</p> <p>nativeName is an optional attribute.</p> <p>If nativeName is not defined for a specific <i>Structural Element</i> in the MTConnectDevices document, it MUST NOT be provided for the corresponding ComponentStream element in the MTConnectStreams document.</p> <p>If nativeName is defined for a specific <i>Structural Element</i> in the MTConnectDevices document, it MAY be provided for the corresponding ComponentStream element in the MTConnectStreams document.</p> <p>If provided, the value reported for nativeName MUST be the same as the value defined for the nativeName attribute of the corresponding <i>Structural Element</i> (Device element, <i>Top Level</i> Component element, or <i>Lower Level</i> Component element) defined in the MTConnectDevices XML document.</p> | 0..1 |

| Continuation of Table 5 | | |
|-------------------------|---|------------|
| Attribute | Description | Occurrence |
| component | <p>component identifies the <i>Structural Element</i> (<i>Device</i>, <i>Top Level Component</i>, or <i>Lower Level Component</i>) associated with the <i>ComponentStream</i> element.</p> <p>component is a required attribute.</p> <p>The value reported for component MUST be the same as the value defined for the Element Name of the XML container representing the corresponding <i>Structural Element</i> (<i>Device</i> element, <i>Top Level Component</i> element, or <i>Lower Level Component</i> element) defined in the <i>MTConnectDevices</i> XML document.</p> <p>Examples of Component are Device, Axes, Controller, Linear, Electric and Loader.</p> | 1 |
| uuid | <p>uuid of the <i>ComponentStream</i> element.</p> <p>uuid is an optional attribute.</p> <p>If uuid is not defined for a specific <i>Structural Element</i> in the <i>MTConnectDevices</i> document, it MUST NOT be provided for the corresponding <i>ComponentStream</i> element in the <i>MTConnectStreams</i> document.</p> <p>If uuid is defined for a specific <i>Structural Element</i> in the <i>MTConnectDevices</i> document, it MAY be provided for the corresponding <i>ComponentStream</i> element in the <i>MTConnectStreams</i> document, but it is not required.</p> <p>If provided, the value reported for uuid MUST be the same as the value defined for the <i>uuid</i> attribute of the corresponding <i>Structural Element</i> (<i>Device</i> element, <i>Top Level Component</i> element, or <i>Lower Level Component</i> element) defined in the <i>MTConnectDevices</i> XML document.</p> | 0..1 |

466 4.3.3 Elements for ComponentStream

467 In the ComponentStream container, an **Agent MUST** organize the data reported in
 468 each ComponentStream into individual Samples, Events, or Condition XML
 469 containers based on the value of the category attribute (i.e., SAMPLE, EVENT, or CON-
 470 DITION) defined for each *Data Entity* defined in the MTConnectDevices XML doc-
 471 ument.

472 Each ComponentStream element **MUST** include at least one Events, Samples, or
 473 Condition XML container element. *Data Entities* returned in each of the Compo-
 474 nentStream container elements are defined in the *Table 6*.

Table 6: Elements for ComponentStream

| Element | Description | Occurrence |
|-----------|---|-------------------|
| Samples | An XML container type element. Samples organizes the SAMPLE type <i>Data Entities</i> defined in the MTConnectDevices document that are reported in each ComponentStream XML element. | 0..1 [†] |
| Events | An XML container type element. Events organizes the EVENT type <i>Data Entities</i> defined in the MTConnectDevices document that are reported in each ComponentStream XML element. | 0..1 [†] |
| Condition | An XML container type element. Condition organizes the CONDITION type <i>Data Entities</i> defined in the MTConnectDevices document that are reported in each ComponentStream XML element. | 0..1 [†] |

475 Note: [†]The ComponentStream element **MUST** contain at least one of these ele-
 476 ment types.

477 5 Data Entities

478 When a piece of equipment reports values associated with `DataItem` elements defined
 479 in the `MTConnectDevices` document, that information is organized as *Data Entities*
 480 in the `MTConnectStreams` document. These *Data Entities* are organized in containers
 481 within each `ComponentStream` element based on the `category` attribute defined for
 482 the corresponding `DataItem` in the `MTConnectDevices` document:

483 `DataItem` elements defined with a `category` attribute of `SAMPLE` in the `MTCon-`
 484 `nnectDevices` document are mapped to the `Samples` XML container in the associated
 485 `ComponentStream` element.

486 `DataItem` elements defined with a `category` attribute of `EVENT` in the `MTCon-`
 487 `nnectDevices` document are mapped to the `Events` XML container in the associated
 488 `ComponentStream` element.

489 `DataItem` elements defined with a `category` attribute of `CONDITION` in the `MT-`
 490 `ConnectDevices` document are mapped to the `Condition` XML container in the
 491 associated `ComponentStream` element.

492 The XML tree in *Figure 5* demonstrates how *Data Entities* are organized in these contain-
 493 ers.

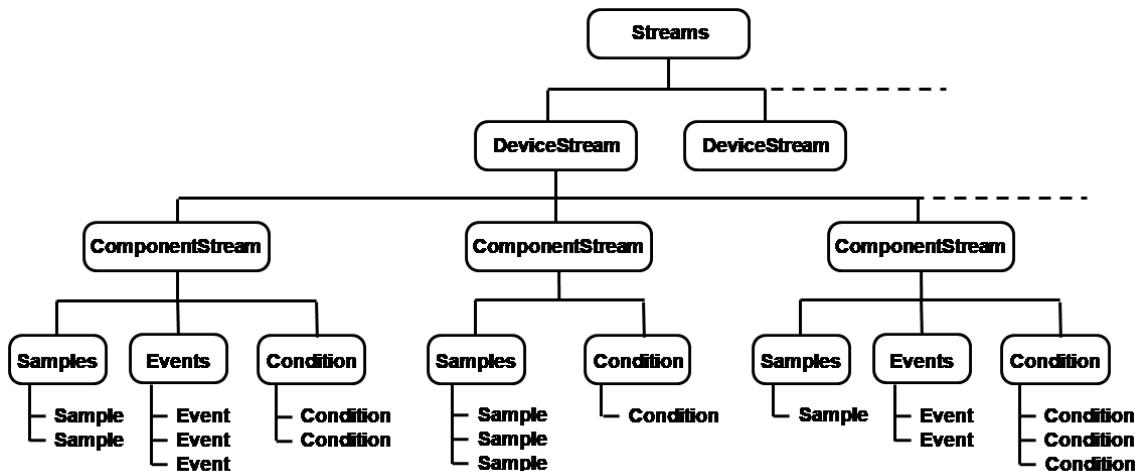


Figure 5: ComponentStream XML Tree Diagram

494 *Example 2* is an illustration of the structure of an XML document demonstrating how *Data*
 495 *Entities* are reported in a `MTConnectStreams` document:

Example 2: Example of MTConnectStreams

```

496 1  <MTConnectStreams>
497 2    <Header/>
498 3    <Streams>
499 4      <DeviceStream>
500 5        <ComponentStream>
501 6          <Samples>
502 7            <Sample/>
503 8            <Sample/>
504 9          </Samples>
505 10         <Events>
506 11           <Event/>
507 12           <Event/>
508 13         </Events>
509 14         <Condition>
510 15           <Condition/>
511 16           <Condition/>
512 17         </Condition>
513 18       </ComponentStream>
514 19       <ComponentStream>
515 20         <Samples>
516 21           <Sample/>
517 22           <Sample/>
518 23         </Samples>
519 24         <Events>
520 25           <Event/>
521 26           <Event/>
522 27         </Events>
523 28         <Condition>
524 29           <Condition/>
525 30           <Condition/>
526 31         </Condition>
527 32       </ComponentStream>
528 33     </DeviceStream>
529 34   </Streams>
530 35 </MTConnectStreams>

```

531 Note: There are no specific requirements defining the sequence in which the Com-
 532 ponentStream XML elements are organized in the MTConnectStreams
 533 document. They **MAY** be organized in any sequence based on the implemen-
 534 tation of an *Agent*. The sequence in which the ComponentStream XML
 535 elements appear does not impact the ability for a client software application to
 536 interpret the information that it receives in the document.

537 When an *Agent* responds to a current HTTP request, the information returned in the
 538 MTConnectStreams document **MUST** include the most current value for every *Data*
 539 *Entity* defined in the MTConnectDevices document subject to any filtering included
 540 within the request.

541 When an *Agent* responds to a sample HTTP request, the information returned in the
 542 MTConnectStreams document **MUST** include the occurrences for each *Data Entity*
 543 that are available to an *Agent* subject to filtering and the count parameter included within
 544 the request (see *MTConnect Standard Part 1.0 - Overview and Fundamentals* for a full
 545 definition of the protocol).

546 5.1 Element Names for Data Entities

547 In the MTConnectDevices document, *Data Entities* are grouped as *DataItem* XML
 548 elements within each *Device*, *Top Level Component*, and *Lower Level Component*
 549 *Structural Element*. The *Data Entities* reported in the MTConnectStreams document
 550 associated with each of these *Structural Elements* are represented with an *Element Name*
 551 based on the category and type defined for each of the *DataItem* elements in the
 552 MTConnectDevices document.

553 5.1.1 Element Names when MTConnectDevices category is SAMPLE 554 or EVENT

555 The *Data Entities* reported in the MTConnectStreams document associated with each
 556 *DataItem* element defined in the MTConnectDevices document with a category
 557 attribute of SAMPLE or EVENT **MUST** be identified in the MTConnectStreams docu-
 558 ment with an *Element Name* derived from the type attribute defined for that *DataItem*
 559 element in the MTConnectDevices document.

560 *Example 3* describes the most common method used to derive the *Element Name* for a *Data*
 561 *Entity* reported in the MTConnectStreams document from the information describing
 562 that *DataItem* element in the MTConnectDevices document:

563 DataItem Represented in the MTConnectDevices Document

Example 3: DataItem Represented in MTConnectDevices Document

```
564 1 <DataItem type="AXIS_FEEDRATE" id="xf" name="Xfrt"  

  565 2   category="SAMPLE" units="MILLIMETER/SECOND"  

  566 3   nativeUnits="MILLIMETER/SECOND/>
```

567 • *DataItem*: The XML *Element Name* for this *Data Entity*.

568 Note: *Element Name* must not be confused with the name attribute for the data
 569 item element.

- 570 • type, category, units, and nativeUnits: Attributes that provide additional
 571 information regarding each data item in the MTConnectDevices document.
 572

573 Response Format reported in the MTConnectStreams Document

Example 4: Response Format reported in the MTConnectStreams Document

```
574 1 <AxisFeedrate name="Xfrt" sequence="61315517"  

  575 2   timestamp="2016-07-28T02:06:01.364428Z"  

  576 3   dataItemId="xf">10.83333</AxisFeedrate>
```

- 577 • AXIS_FEEDRATE: The *Element Name* provided in the MTConnectStreams response
 578 format for the data item. The *Element Name* for a data item is defined by
 579 the type attribute of AXIS_FEEDRATE in the MTConnectDevices document.
 580 The *Element Name* **MUST** be provided in Pascal case format (first letter of each
 581 word is capitalized).

582 **5.1.2 Changes to Element Names when representation attribute is
 583 used**

584 The *Element Name* for a *Data Entity* reported in the MTConnectStreams document is
 585 extended when the representation attribute is used to further describe that DataItem
 586 element in the MTConnectDevices document.

587 **5.1.3 Element Names when MTConnectDevices category is CONDITION**

589 *Data Entities* defined in the MTConnectDevices document with a category attribute
 590 of CONDITION are reported with an *Element Name* that is defined differently from other
 591 *Data Entity* types. The *Element Name* for these *Data Entities* are defined based on
 592 the *Fault State* (Normal, Warning, or Fault) associated with each *Data Entity* at the
 593 time that a value for that *Data Entity* is reported. See *Section 5.7.1 - Element Names for
 594 Condition* and *Section 5.8 - Unavailability of Fault State for Condition* for details on how
 595 these *Data Entities* are reported in the MTConnectStreams document.

596 5.2 Samples Container

597 Samples is a XML container type element. Samples organizes the *Data Entities* re-
 598 turned in the MTConnectStreams XML document for those DataItem elements de-
 599 fined with a category attribute of SAMPLE in the MTConnectDevices document.

600 A separate Samples container will be provided for the data returned for the DataItem
 601 elements associated with each *Structural Element* of a piece of equipment defined in the
 602 MTConnectDevices document.

Table 7: MTConnect Samples Element

| Element | Description | Occurrence |
|---------|--|------------|
| Samples | <p>An XML container type element that organizes the data reported in the MTConnectStreams document for DataItem elements defined in the MTConnectDevices document with a category attribute of SAMPLE.</p> <p>A separate Samples container MUST be provided for each ComponentStream element for which data is returned for a DataItem element defined in the MTConnectDevices document with a category attribute of SAMPLE.</p> <p>If provided in the document, a Samples XML container MUST contain at least one Sample element.</p> | 0..1 |

603 5.3 Sample Data Entities

604 A Sample XML element provides the information and data reported from a piece of
 605 equipment for those DataItem elements defined with a category attribute of SAMPLE
 606 in the MTConnectDevices document.

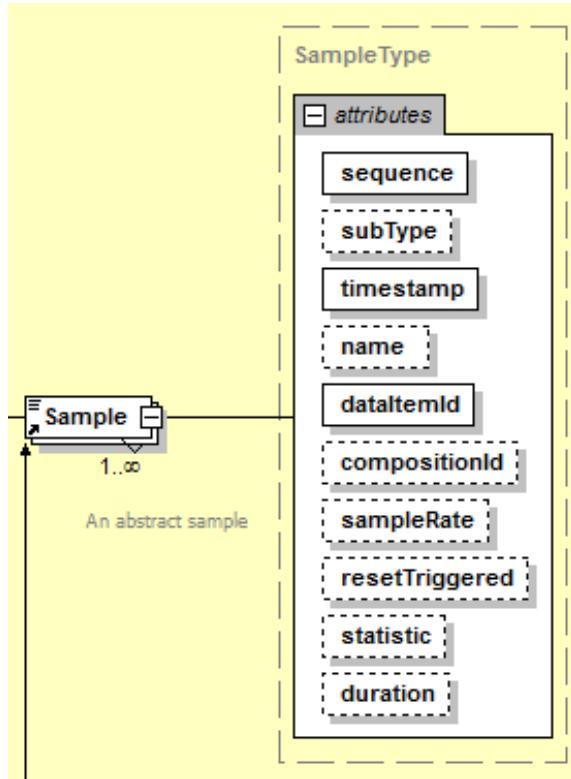
607 Sample is an abstract type XML element and will never appear directly in the MTCon-
 608 nectStreams XML document. As an abstract type XML element, Sample will be
 609 replaced in the XML document by a specific type of Sample specified by the *Element*
 610 *Name* for that *Data Entity*. The different types of Sample elements are defined in
 611 *Section 6.1 - Sample Element Names*. Examples of XML elements representing Sample
 612 include PathPosition, Temperature.

Table 8: MTConnect Sample Element

| Element | Description | Occurrence |
|---------|---|------------|
| Sample | <p>An XML element that provides the information and data reported from a piece of equipment for those DataItem elements defined with a category attribute of SAMPLE in the MTConnectDevices document.</p> <p>Sample is an abstract type XML element. It is replaced in the MTConnectStreams document by a specific type of Sample element.</p> <p>There MAY be multiple types of Sample elements in a Samples container.</p> | 1..* |

613 5.3.1 XML Schema Structure for Sample

614 The XML schema in *Figure 6* represents the structure of a Sample XML element showing the attributes defined for Sample elements.
 615

**Figure 6:** Sample Schema Diagram

616 5.3.2 Attributes for Sample

617 The *Table 9* defines the attributes used to provide additional information for a `Sample`
 618 XML element.

Table 9: Attributes for Sample

| Attribute | Description | Occurrence |
|-----------------------|--|------------|
| <code>sequence</code> | A number representing the sequential position of an occurrence of the <code>Sample</code> in the data buffer of an <i>Agent</i> . <code>sequence</code> is a required attribute. <code>sequence</code> MUST have a value represented as an unsigned 64-bit value from 1 to $2^{64} - 1$. | 1 |

| Continuation of Table 9 | | |
|-------------------------|--|------------|
| Attribute | Description | Occurrence |
| subType | <p>The subType of the <i>Data Entity</i>.</p> <p>subType is an optional attribute.</p> <p>subType MUST match the subType attribute of the DataItem element as defined in the MTConnectDevices document that the Sample element represents.</p> | 0..1 |
| timestamp | <p>The most accurate time available to a piece of equipment that represents the point in time that the data reported for the Sample was measured.</p> <p>When the Sample element represents a DataItem element defined in the MTConnectDevices document with a representation or statistic attribute, timestamp MUST represent the time that the data collection was completed.</p> <p>timestamp is a required attribute.</p> | 1 |
| name | <p>The name of the Sample element.</p> <p>name is an optional attribute.</p> <p>name MUST match the name attribute of the DataItem element defined in the MTConnectDevices document that the Sample element represents.</p> <p>An NMOKEN XML type.</p> | 0..1 |
| dataItemId | <p>The unique identifier for the Sample element.</p> <p>dataItemId is a required attribute.</p> <p>dataItemId MUST match the id attribute of the DataItem element defined in the MTConnectDevices document that the Sample element represents.</p> | 1 |

| Continuation of Table 9 | | |
|-------------------------|---|------------|
| Attribute | Description | Occurrence |
| sampleRate | <p>The rate at which successive samples of the value of a data item are recorded.</p> <p>sampleRate is expressed in terms of samples per second.</p> <p>sampleRate is an optional attribute.</p> <p>If the sampleRate is smaller than one, the number can be represented as a decimal type floating-point number. For example, a rate of 1 per 10 seconds would be 0.1</p> <p>sampleRate MUST be provided when the representation attribute of the DataItem element defined in the MTConnectDevices document that this Sample element represents is TIME_SERIES.</p> <p>For DataItem elements where the representation attribute defined in the MTConnectDevices document that this Sample element represents is not TIME_SERIES, it MUST be assumed that the data reported is represented by a single value and sampleRate MUST NOT be reported in the MTConnectStreams document.</p> | 0..1 |
| statistic | <p>The type of statistical calculation defined by the statistic attribute of the DataItem element defined in the MTConnectDevices document that this Sample element represents.</p> <p>statistic is an optional attribute.</p> | 0..1 |

| Continuation of Table 9 | | |
|-------------------------|--|------------|
| Attribute | Description | Occurrence |
| duration | <p>The time-period over which the data was collected.</p> <p>duration is an optional attribute.</p> <p>duration MUST be provided when the statistic attribute of the DataItem element is defined in the MTConnectDevices document that this Sample element represents.</p> | 0..1 |
| resetTriggered | <p>For those DataItem elements that report data that may be periodically reset to an initial value, resetTriggered identifies when a reported value has been reset and what has caused that reset to occur.</p> <p>resetTriggered is an optional attribute.</p> <p>resetTriggered MUST only be provided for the specific occurrence of a <i>Data Entity</i> reported in the MTConnectStreams document when the reset occurred and MUST NOT be provided for any other occurrence of the <i>Data Entity</i> reported in a MTConnectStreams document.</p> | 0..1 |
| compositionId | <p>The identifier of the Composition element defined in the MTConnectDevices document associated with the data reported for the Sample element.</p> <p>compositionId is an optional attribute.</p> | 0..1 |

619 **5.3.2.1 duration Attribute for Sample**

620 Sample elements that represent the result of a computed value of a statistic **MUST** contain a duration attribute. For these *Data Entities*, the timestamp associated with
 621 the Sample **MUST** reference the time the data collection was completed. timestamp
 622 **MUST NOT** represent any other time associated with the data collection or the calculation
 623 of the statistic. The actual time the interval began can be computed by subtracting the
 624 duration from the timestamp.

626 Two Sample elements **MAY** have overlapping time periods when statistics are computed
 627 at different frequencies. For example, there may be two *Data Entities* reporting a statistic
 628 representing the average value for the readings of the same measured signal calculated over
 629 one and five minute intervals. These *Data Entities* can both have the same start time for
 630 their calculations (e.g., 05:10:00), but the timestamp and duration will be 05:11:00
 631 and 60 seconds, respectively, for the *Data Entity* reporting the one-minute average and
 632 05:15:00 and 300 seconds, respectively, for the *Data Entity* reporting the five-minute av-
 633 erage. This allows for varying statistical methods to be applied with different interval
 634 lengths each having different values for the timestamp and duration attributes.

635 **5.3.2.2 resetTriggered Attribute for Sample**

636 Some *Data Entities* **MAY** have their reported value reset to an initial value. These reset
 637 actions may be based upon a specific elapsed time or may be triggered by a physical or
 638 logical reset action that causes the reset to occur. Examples of *Data Entities* that **MAY**
 639 have their reported value reset to an initial value are *Data Entities* representing a counter,
 640 a timer, or a statistic.

641 resetTriggered defines the type of reset action that caused the value of the reported
 642 data to be reset. The value reported for resetTriggered **MAY** be defined by the
 643 ResetTrigger element for the *Data Entity* in the MTConnectDevices document
 644 that this Sample element represents. If the ResetTrigger element is not defined in the
 645 MTConnectDevices document, a resetTriggered attribute **SHOULD** be reported
 646 in the MTConnectStreams document if the type of reset action can be determined and
 647 reported by the piece of equipment.

648 resetTriggered **MUST** only be reported for the first occurrence of a *Data Entity*
 649 after a reset action has occurred and **MUST NOT** be provided for any other occurrence
 650 of the *Data Entity* reported in a MTConnectStreams document. When a reset occurs,
 651 the piece of equipment **MUST** report an occurrence of the *Data Entity* that was reset even
 652 if that occurrence of the *Data Entity* would normally be suppressed based on the filtering
 653 criteria established in the MTConnectDevices document that this Sample element
 654 represents.

655 The *Table 10* provides the values that **MAY** be reported for `resetTriggered`:

Table 10: Values for `resetTriggered`

| Value for <code>resetTriggered</code> | Description |
|---------------------------------------|---|
| ACTION_COMPLETE | The value of the <i>Data Entity</i> that is measuring an action or operation was reset upon completion of that action or operation. |
| ANNUAL | The value of the <i>Data Entity</i> was reset at the end of a 12-month period. |
| DAY | The value of the <i>Data Entity</i> was reset at the end of a 24-hour period. |
| MAINTENANCE | The value of the <i>Data Entity</i> was reset upon completion of a maintenance event. |
| MANUAL | The value of the <i>Data Entity</i> was reset based on a physical reset action. |
| MONTH | The value of the <i>Data Entity</i> was reset at the end of a monthly period. |
| POWER_ON | The value of the <i>Data Entity</i> was reset when power was applied to the piece of equipment after a planned or unplanned interruption of power has occurred. |
| SHIFT | The value of the <i>Data Entity</i> was reset at the end of a work shift. |
| WEEK | The value of the <i>Data Entity</i> was reset at the end of a 7-day period. |

656 **5.3.3 Response for SAMPLE category DataItem Elements with a rep-**
 657 **resentation Attribute of TIME_SERIES**

658 SAMPLE category DataItem elements defined in the MTConnectDevices document
 659 with a representation attribute of `TIME_SERIES` **MUST** be represented in the MT-
 660 ConnectStreams document as Sample elements that report data that includes multi-
 661 ple values representing a series of readings of a measured value taken at a specific sample
 662 rate. Such a DataItem element can be defined for collecting high frequency readings of
 663 a measured value and then providing the entire series of values to a client software appli-
 664 cation as the data reported for a single *Data Entity*. In this case, the `sampleCount` and

665 sampleRate attributes **MUST** be provided.

666 Note: sampleCount is an attribute that **MUST** only be provided for Sample
 667 elements that represent SAMPLE category DataItem elements defined in
 668 the MTConnectDevices document with a representation attribute of
 669 TIME_SERIES.

670 The CDATA provided for the *Data Entity* **MUST** be a series of space delimited floating-
 671 point numbers. The number of values **MUST** match the sampleCount.

672 5.3.3.1 XML Schema Structure for Sample when reporting Time Series Data

673 The XML schema in *Figure 7* represents the extended structure of a Sample XML el-
 674 ement that represents a SAMPLE category DataItem element defined in the MTCon-
 675 nectDevices document with a representation attribute of TIME_SERIES.

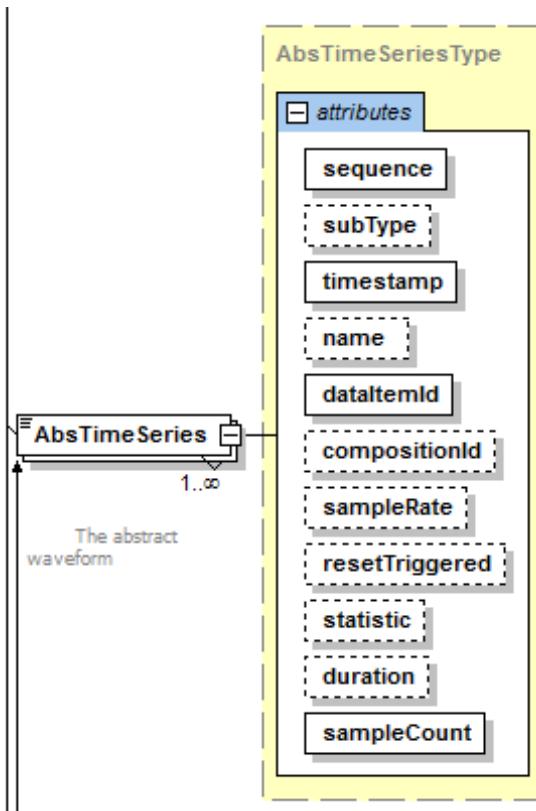


Figure 7: AbsTimeSeries Schema Diagram

676 Note: The AbsTimeSeries element shown in the XML schema is an abstract
 677 type element and will be replaced in the MTConnectStreams document by

678 the *Element Name* derived from the `type` attribute defined for the associated
 679 DataItem element defined in the MTConnectDevices document.

680 **5.3.3.2 Attributes for a Sample when reporting Time Series Data**

681 *Table 11* defines the additional attribute provided for a Sample XML element that rep-
 682 resents a SAMPLE category DataItem element defined in the MTConnectDevices
 683 document with a representation attribute of TIME_SERIES.

Table 11: MTConnect sampleCount Attribute

| Attribute | Description | Occurrence |
|-------------|--|------------|
| sampleCount | <p>The number of readings reported in the data returned for the DataItem element defined in the MTConnectDevices document that this Sample element represents.</p> <p>sampleCount is an optional attribute.</p> <p>sampleCount MUST be provided when the representation attribute of the DataItem element is TIME_SERIES.</p> <p>sampleCount MUST NOT be provided when the representation attribute is defined as DISCRETE (DEPRECATED in Version 1.5) or VALUE, or when it is not defined.</p> | 0..1 |

684 **5.3.4 Response for SAMPLE category DataItem Elements with a rep-
 685 resentation attribute of DATA_SET**

686 SAMPLE category DataItem elements defined in the MTConnectDevices document
 687 with a representation attribute of DATA_SET **MUST** be represented in the MTCon-
 688 nectStreams document as Sample XML Elements reported as a *Data Set* of *key-value*
 689 *pairs*. DATA_SET provides the capability to report a set of related data values as a single
 690 *Data Entity*.

691 The Sample XML Element acts as a container for Entry elements to provide a *Data Set*
 692 of *key-value pairs* where each key attribute of the Entry **MUST** be unique and acts as
 693 the identity of the *key-value pair*. The CDATA of the Entry element represents the value

694 portion of the *key-value pair* and has the same constraints as the *Data Entity* type defined
 695 for the DataItem type.

696 5.3.4.1 XML Schema Structure for Sample when reporting Data Set data

697 *Figure 8* represents the XML schema of a Sample XML element that represents a SAM-
 698 PLE category DataItem element defined in the MTConnectDevices document with
 699 a representation attribute of DATA_SET.

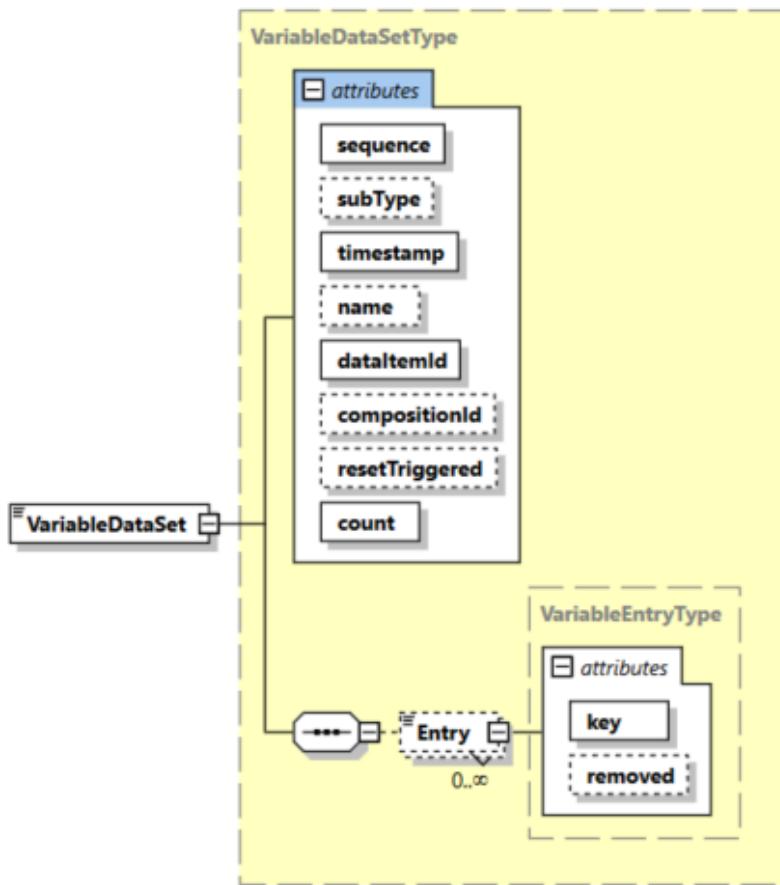


Figure 8: Sample Data Set Schema Diagram

700 5.3.4.2 Attributes for Sample when reporting Data Set data

701 *Table 12* defines the additional attribute provided for a Sample XML element that rep-
 702 resents a SAMPLE category DataItem element defined in the MTConnectDevices
 703 document with a representation attribute of DATA_SET.

Table 12: Attributes for DataSet

| Attribute | Description | Occurrence |
|-----------|---|------------|
| count | Represents the number of <i>key-value pairs</i> represented as Entry elements as the contents of the Sample element. count MUST be provided when the representation attribute of the DataItem element is DATA_SET. count MUST NOT be provided when the representation attribute is defined as DISCRETE (DEPRECATED in Version 1.5), TIME_SERIES, or VALUE, or when it is not defined. | 0..1 |

704 5.3.4.3 Elements for Sample when reporting Data Set data

705 *Table 13* defines the elements provided for a Sample XML element that represents a
 706 SAMPLE category DataItem element defined in the MTConnectDevices document
 707 with a representation attribute of DATA_SET. Entry is the only child element that
 708 **MAY** be associated with a *Data Entity* with a representation attribute of DATA_--
 709 SET. Each Entry element represents a unique *key-value pair*.

Table 13: Elements for DataSet

| Element | Description | Occurrence |
|---------|---|------------|
| Entry | A XML element representing a <i>key-value pair</i> published as part of a <i>Data Set</i> . | 0..* |

710 5.3.4.3.1 XML Schema Structure for Entry Element for a Data Entity

711 *Figure 9* represents the XML Schema structure for a Entry XML element that represents
 712 the information published for a *key-value pair*. Any number of Entry elements **MAY** be
 713 provided for a *Data Entity* defined with a representation attribute of DATA_SET.

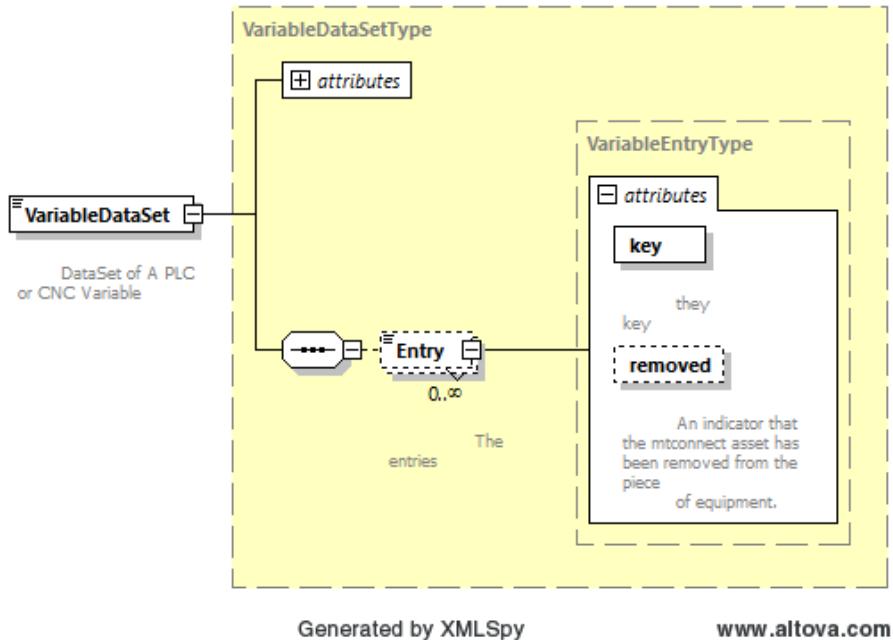


Figure 9: Entry Element Schema Diagram

714 Note: The `VariableDataSet` element shown in the XML schema is an example
 715 that illustrates the schema for a *Data Entity* element and its associated `Entry`
 716 elements representing a *Data Set*.

717 The following example demonstrates how multiple *key-value pairs*, each defined by an
 718 `Entry` element, are structured in a MTConnectStreams document.

Example 5: Example of multiple key-value pairs Reported for a Data Entity

```
719 1 <VariableDataSet timestamp="..." sequence="..." count="2">
720 2   <Entry key="a101">100.21</Entry>
721 3   <Entry key="a102">609</Entry>
722 4   <Entry key="a103" removed="true" />
723 5 </VariableDataSet>
```

724 **5.3.4.3.2 Attributes for Entry Element for a Data Entity**

725 The *Table 14* defines the attributes provided for a `Entry` XML element.

Table 14: Attributes for Entry

| Attribute | Description | Occurrence |
|-----------|--|------------|
| key | <p>A unique identifier for each <i>key-value pair</i>.</p> <p>The value provided for key MUST be unique in any given set of <i>Entry</i> elements.</p> <p>The value provided for key MUST be a XML NMTOKEN type.</p> | 1 |
| removed | <p>A indicator defining whether a specific <i>key-value pair</i> has been removed from the set of <i>key-value pairs</i> associated with this <i>Data Set</i>.</p> <p><i>removed</i> is an XML Boolean type that MUST have a value of true or false.</p> <p>true indicates that the <i>key-value pair</i> has been removed from the <i>Data Set</i>.</p> <p>false indicates that the <i>key-value pair</i> has not been removed from the <i>Data Set</i>.</p> <p>If not specified, the default value for removed is false</p> | 0..1 |

726 5.3.5 Valid Data Values for Sample

- 727 All *Sample* elements reported in an MTConnectStreams XML document **MUST** provide a value in the CDATA of the *Data Entity*.
- 729 The value returned in the CDATA **MUST** be reported as either a *Valid Data Value* representing the information reported from a piece of equipment or UNAVAILABLE when a *Valid Data Value* cannot be determined.
- 732 The *Valid Data Value* reported for a *Sample* represents the reading of the value of a continuously variable or analog data source.
- 734 The representation attribute for a SAMPLE category *DataItem* element defined in the MTConnectDevices document specifies how an *Agent* **MUST** record instances of the data associated with that data item and how often that data **MUST** be reported as a *Sample* element in the MTConnectStreams document.
- 738 The data reported for a *Sample* element associated with a SAMPLE category *DataItem*

739 element with a representation of VALUE can be measured at any point-in-time and
 740 **MUST** always produce a result with a single data value.

741 Note: If a representation attribute is not specified in the MTConnectDe-
 742 vices document for a DataItem element, it **MUST** be assumed that the
 743 data reported in the MTConnectStreams document for the *Data Entity* has
 744 a representation type of VALUE.

745 In the case of a Sample element associated with a SAMPLE category DataItem element
 746 with a representation attribute of TIME_SERIES, the data provided **MUST** be a
 747 series of data values representing multiple sequential samples of the measured value that
 748 will be provided only at the end of the completion of a sampling period. (See Section
 749 *Section 5.3.3 - Response for SAMPLE category DataItem Elements with a representation*
 750 *Attribute of TIME_SERIES* for more information on TIME_SERIES type data).

751 In the case of a Sample element associated with a SAMPLE category DataItem element
 752 with a representation attribute of DATA_SET, the data reported for each *key-value*
 753 **MUST** be provided in the same *Valid Data Values* and units as specified by the type
 754 attribute for the DataItem element.

755 When an Agent responds to a *Current Request*, the information returned in the MTCon-
 756 nectStreams document for a *Data Entity* defined to represent a *Data Set* **MUST** in-
 757 clude the full set of *key-value pairs* that are valid for that *Data Entity*. If the *Current*
 758 *Request* includes an *at query parameter*, the Agent **MUST** provide the set of *key-value*
 759 *pairs* that are valid at the specified *sequence number*.

760 When an Agent responds to a *Sample Request*, the information returned in the MTCon-
 761 nectStreams document for a *Data Entity* defined to represent a *Data Set* **MUST** in-
 762 clude only those *key-value pairs* that are valid for the *Data Entity* at each *sequence number*.

763 Data values provided for a Sample **MUST** always be a floating-point number. In the
 764 MTConnect Standard, floating-point numbers are defined as XML xs:float type numbers
 765 as defined by W3C. Any of the following number formats are valid XML floating type
 766 numbers: 1267.43233E12, -1E4, 12.78e-2, 12, 137.2847, 0, and INF.

767 Note: For some Sample elements, the *Valid Data Value* **MAY** be restricted to spe-
 768 cific formats. See Section 6.1 of this document for a description of any restric-
 769 tions of the acceptable format for *Valid Data Value*.

770 For Sample elements, a client software application can determine the appropriate accu-
 771 racy of the value reported for the *Data Entity* by applying the significantDigits attribute
 772 defined for the corresponding DataItem element defined in the MTConnectDevices
 773 document.

774 The *Valid Data Value* reported as CDATA for a `Sample` element **MUST** be formatted as
 775 part of the content between the element tags in the XML element representing that *Data Entity*. As an example, a `Position` is formatted as shown in *Example 6*.

Example 6: Example showing CDATA of a `DataItem` Element

```
777 1 <Position sequence="112" name="Xabs"
778 2     timestamp="2016-07-28T02:06:01.364428Z"
779 3     dataItemId="10">123.3333</Position>
```

780 In this example, the 123.3333 is the CDATA for `Position`. All CDATA in a `Sample` element is typed, which means that the value reported for the *Data Entity* **MUST** be formatted as defined in Section 6.1 for each *Data Entity* so that it can be validated.

783 5.3.6 Unavailability of Valid Data Values for Sample

784 If an *Agent* cannot determine a *Valid Data Value* for a `Sample` element, the value returned
 785 for the CDATA for the *Data Entity* **MUST** be reported as UNAVAILABLE.

786 *Example 7* demonstrates how an *Agent* reports the value for a `Sample` in the CDATA
 787 when it is unable to determine a *Valid Data Value*:

Example 7: Example of CDATA when Data Entity is UNAVAILABLE

```
788 1 <Samples>
789 2   <PathPosition dataItemId="p2"
790 3     timestamp="2009-03-04T19:45:50.458305"
791 4     subType="ACTUAL" name="Zact"
792 5     sequence="15065113">UNAVAILABLE</PathPosition>
793 6   <Temperature dataItemId="t6"
794 7     timestamp="2009-03-04T19:45:50.458305" name="temp"
795 8     sequence="150651134">UNAVAILABLE</Temperature>
796 9 </Samples>
```

797 5.4 Events Container

798 `Events` is a XML container type element. `Events` organizes the *Data Entities* returned
 799 in the `MTConnectStreams` XML document for those `DataItem` elements defined
 800 with a `category` attribute of `EVENT` in the `MTConnectDevices` document.

801 A separate `Events` container will be provided for the data returned for the `DataItem`
 802 elements associated with each *Structural Element* of a piece of equipment defined in the
 803 `MTConnectDevices` document.

Table 15: MTConnect Event Element

| Element | Description | Occurrence |
|---------|---|------------|
| Events | <p>An XML container type element that organizes the data reported in the MTConnect Streams document for DataItem elements defined in the MTConnectDevices document with a category attribute of EVENT.</p> <p>A separate Events container MUST be provided for each ComponentStream element for which data is returned for a DataItem element defined in the MTConnectDevices document with a category attribute of EVENT.</p> <p>If provided in the document, an Events XML container MUST contain at least one Event element.</p> | 0..1 |

804 5.5 Event Data Entities

805 An Event XML element provides the information and data provided from a piece of
 806 equipment for those DataItem elements defined with a category attribute of EVENT
 807 in the MTConnectDevices document.

808 Event is an abstract type XML element and will never appear directly in the MTCon-
 809 nectStreams XML document. As an abstract type XML element, Event will be
 810 replaced in the XML document by a specific type of Event specified by the *Element*
 811 *Name* for that *Data Entity*. The different types of Event elements are defined in *Sec-
 812 tion 6.2 - Event Element Names*. Examples of XML elements representing Event include
 813 Block and Execution.

814 Event is similar to Sample, but its value can change with unpredictable frequency.
 815 Events do not report intermediate values. As an example, when Availability trans-
 816 transitions from UNAVAILABLE to AVAILABLE, there is no intermediate state that can be
 817 inferred.

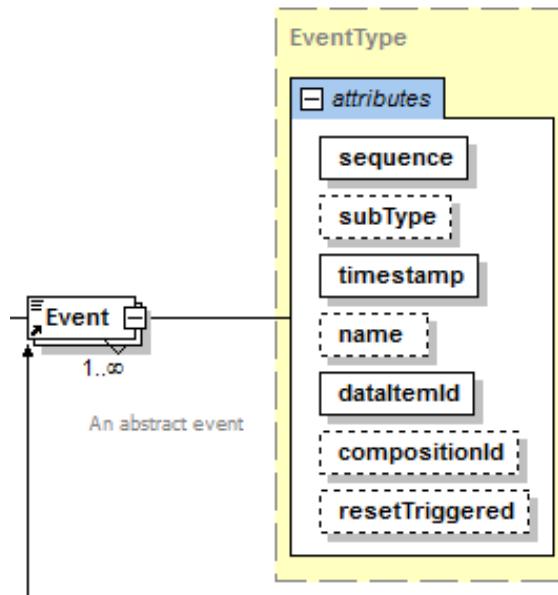
818 Event elements **MAY** report data values defined by a controlled vocabulary as speci-
 819 fied in *Section 6.2 - Event Element Names*, by numeric values, or by a character string
 820 representing text or a message provided by the piece of equipment.

Table 16: MTConnect Event Element

| Element | Description | Occurrence |
|---------|--|------------|
| Event | An XML element which provides the information and data reported from a piece of equipment for those DataItem elements defined with a category attribute of EVENT in the MTConnectDevices document. Event is an abstract type XML element. It is replaced in the MTConnectStreams document by a specific type of Event element. There MAY be multiple types of Event elements in a Events container. | 1..* |

821 5.5.1 XML Schema Structure for Event

822 The XML schema in *Figure 10* represents the structure of an Event XML element showing
 823 the attributes defined for Event elements.

**Figure 10:** Event Schema Diagram

824 5.5.2 Attributes for Event

825 *Table 17* defines the attributes that **MAY** be used to provide additional information for an
 826 Event XML element.

Table 17: Attributes for Event

| Attribute | Description | Occurrence |
|-----------|---|------------|
| sequence | <p>A number representing the sequential position of an occurrence of the <i>Event</i> in the data buffer of an <i>Agent</i>.</p> <p>sequence is a required attribute.</p> <p>sequence MUST have a value represented as an unsigned 64-bit value from 1 to $2^{64} - 1$.</p> | 1 |
| subType | <p>The subType of the <i>Data Entity</i>.</p> <p>subType is an optional attribute.</p> <p>subType MUST match the subType attribute of the DataItem element as defined in the MTConnectDevices document that the Event element represents.</p> | 0..1 |
| timestamp | <p>The most accurate time available to a piece of equipment that represents the point in time that the data reported for the Event was measured.</p> <p>timestamp is a required attribute.</p> | 1 |
| name | <p>The name of the Event element.</p> <p>name is an optional attribute.</p> <p>name MUST match the name attribute of the DataItem element defined in the MTConnectDevices document that the Event element represents.</p> <p>An NMOKEN XML type.</p> | 0..1 |

| Continuation of Table 17 | | |
|--------------------------|--|------------|
| Attribute | Description | Occurrence |
| dataItemId | <p>The unique identifier for theEvent element.</p> <p>dataItemId is a required attribute.</p> <p>dataItemId MUST match the id attribute of the DataItem element defined in the MTConnectDevices document that the Event element represents.</p> | 1 |
| resetTriggered | <p>For those DataItem elements that report data that may be periodically reset to an initial value, resetTriggered identifies when a reported value has been reset and what has caused that reset to occur.</p> <p>resetTriggered is an optional attribute.</p> <p>resetTriggered MUST only be provided for the specific occurrence of a <i>Data Entity</i> reported in the MTConnectStreams document when the reset occurred and MUST NOT be provided for any other occurrence of the <i>Data Entity</i> reported in a MTConnectStreams document.</p> | 0..1 |
| compositionId | <p>The identifier of the Composition element defined in the MTConnectDevices document associated with the data reported for the Event element.</p> <p>compositionId is an optional attribute.</p> | 0..1 |

827 **5.5.3 Response for EVENT category DataItem Elements with a rep-**
 828 **resentation attribute of DATA_SET**

829 The behavior of EVENT category DataItem elements defined in the MTConnectDe-
 830 vices document with a representation attribute of DATA_SET function exactly
 831 the same as SAMPLE category DataItem elements with a representation attribute
 832 of DATA_SET. Refer to *Section 5.3.4 - Response for SAMPLE category DataItem Ele-*
833 ments with a representation attribute of DATA_SET for details on DataItem elements
 834 with a representation attribute of DATA_SET.

835 5.5.4 Valid Data Values for Event

836 Event elements reported in an MTConnectStreams XML document **MUST** provide
837 a value in the CDATA of the *Data Entity*.

838 The value reported in the CDATA **MUST** be reported as either a *Valid Data Value* rep-
839 resenting the information reported from a piece of equipment or UNAVAILABLE when a
840 *Valid Data Value* cannot be determined.

841 The *Valid Data Value* reported for an Event represents a distinct piece of information
842 provided from a piece of equipment. Unlike Sample, Event does not report intermediate
843 values that vary over time. Event reports information that, when provided at any specific
844 point in time, represents the current state of the piece of equipment.

845 The representation attribute for an EVENT category data item defined in the MT-
846 ConnectDevices document specifies how an *Agent* **MUST** record instances of data
847 associated with that data item and how that data **MUST** be reported as an Event element
848 in the MTConnectStreams document.

849 The data reported for an Event element associated with an EVENT category data item
850 with a representation attribute of VALUE **MUST** be either an integer, a floating-
851 point number, a descriptive value (text string) representing one of two or more state values
852 defined for that data item, or a text string representing a message.

853 If a representation attribute is not specified for a data item in an MTConnectDe-
854 vices document, the designation for the representation attribute **MUST** be inter-
855 preted as VALUE.

856 In the case of an Event element associated with a EVENT category DataItem element
857 with a representation attribute of DATA_SET, the data reported for each *key-value*
858 **MUST** be provided in the same *Valid Data Values* and units as specified by the type
859 attribute for the DataItem element.

860 When an *Agent* responds to a *Current Request*, the information returned in the MTCon-
861 nectStreams document for a *Data Entity* defined to represent a *Data Set* **MUST** in-
862 clude the full set of *key-value pairs* that are valid for that *Data Entity*. If the *Current*
863 *Request* includes an *at query parameter*, the *Agent* **MUST** provide the set of *key-value*
864 *pairs* that are valid at the specified *sequence number*.

865 When an *Agent* responds to a *Sample Request*, the information returned in the MTCon-
866 nectStreams document for a *Data Entity* defined to represent a *Data Set* **MUST** in-
867 clude only those *key-value pairs* that are valid for the *Data Entity* at each *sequence number*
868 The *Valid Data Value* reported as CDATA for an Event element **MUST** be formatted as

869 part of the content between the element tags in the XML element representing that *Data*
 870 *Entity*. As an example, Event elements are formatted as shown in *Example 8*:

Example 8: Example of Event Element

```

871 1 <PartCount dataItemId="pc4"
872 2     timestamp="2009-02-26T02:02:36.48303"
873 3     name="pcount" sequence="185">238</PartCount>
874 4 <ControllerMode dataItemId="p3"
875 5     timestamp="2009-02-26T02:02:35.716224"
876 6     name="mode" sequence="192">AUTOMATIC</ControllerMode>
877 7     <Block dataItemId="cn2" name="block" sequence="206"
878 8         timestamp="2009-02-26T02:02:37.394055">G0Z1</Block>
```

879 In these examples, 238 is the CDATA for PartCount and is a numeric value; AUTO-
 880 MATIC is the CDATA for the ControllerMode and is a descriptive value representing
 881 a state for the *Data Entity*; and G0Z1 is a text string representing a message describing the
 882 program code associated with the Block *Data Entity*.

883 5.5.5 Unavailability of Valid Data Value for Event

884 If an *Agent* cannot determine a *Valid Data Value* for an Event element, the value returned
 885 for the CDATA for the *Data Entity* **MUST** be reported as UNAVAILABLE.

886 The example in *Example 9* demonstrates how an *Agent* reports the value for an Event in
 887 the CDATA when it is unable to determine a *Valid Data Value*:

Example 9: Example of Event Element when data value is UNAVAILABLE

```

888 1 <Events>
889 2     <ControllerMode dataItemId="p3"
890 3         timestamp="2009-02-26T02:02:35.716224" name="mode"
891 4         sequence="182">UNAVAILABLE</ControllerMode>
892 5 </Events>
```

893 5.6 Condition Container

894 Condition is a XML container type element. Condition organizes the *Data Entities*
 895 returned in the MTConnectStreams XML document for those DataItem elements
 896 defined with a category attribute of CONDITION in the MTConnectDevices docu-
 897 ment.

898 A separate Condition container will be provided for the data returned for the DataItem

899 elements associated with each *Structural Element* of a piece of equipment defined in the
 900 MTConnectDevices document.

Table 18: MTConnect Condition Element Container

| Element | Description | Occurrence |
|-----------|---|------------|
| Condition | <p>An XML container type element that organizes the data reported in the MTConnectStreams document for DataItem elements defined in the MTConnectDevices document with a category attribute of CONDITION.</p> <p>A separate Condition container MUST be provided for each ComponentStream element for which data is returned for a DataItem element defined in the MTConnectDevices document with a category attribute of CONDITION.</p> <p>If provided in the document, a Condition XML container MUST contain at least one Condition element.</p> | 0..1 |

901 5.7 Condition Data Entity

902 A Condition XML element provides the information and data provided from a piece of
 903 equipment for those DataItem elements defined with a category attribute of CON-
 904 DITION in the MTConnectDevices document.

905 Condition provides information reported by a piece of equipment describing its health
 906 and ability to function.

907 Condition is an abstract type XML element and will never appear directly in the MT-
 908 ConnectStreams XML document. As an abstract type XML element, Condition
 909 will be replaced in the XML document by a *Data Entity* representing the CONDITION
 910 category DataItem element defined in the MTConnectDevices document that this
 911 Condition element represents.

912 The *Data Entities* represented by Condition are structured differently than the *Data*
 913 *Entities* representing Sample and Event. The *Element Name* for each Condition
 914 element reported in the MTConnectStreams document defines the *Fault State* of the
 915 *Data Entity*. A Condition element is identified by the *Structural Element* to which it is

916 associated, along with the type and dataItemId defined for the element. *Section 6.3*
 917 - *Types of Condition Elements* provides details on the different types of Condition
 918 elements.

Table 19: MTConnect Condition Element

| Element | Description | Occurrence |
|-----------|---|------------|
| Condition | <p>An XML element which provides the information and data reported from a piece of equipment for those DataItem elements defined with a category attribute of CONDITION in the MTConnectDevices document.</p> <p>Condition is an abstract type XML element. It is replaced in the MTConnectStreams document by a specific type of Condition element.</p> <p>There MAY be multiple types of Condition elements in a Conditions container.</p> | 1..* |

919 CONDITION type DataItem elements defined in the MTConnectDevices document
 920 **MAY** report multiple simultaneous *Fault States* in the MTConnectStreams document.
 921 This is unlike a SAMPLE or EVENT DataItem element that can only report a single
 922 occurrence of a Sample or Event element in the MTConnectStreams document at
 923 any one point in time.

924 For example, a controller on a piece of equipment may detect and report multiple for-
 925 mat errors in a motion program. Each error represents a separate *Fault State* from the
 926 controller. Each *Fault State* is represented as a separate Condition element in the MT-
 927 ConnectStreams document since each *Fault State* **MUST** be identified and tracked
 928 individually in the document.

929 5.7.1 Element Names for Condition

930 Condition elements are reported differently from other *Data Entity* types. The *El-*
 931 *ement Name* reported for a Condition element represents the *Fault State* (Normal,
 932 Warning, or Fault) associated with each Condition.

933 Examples of XML elements representing Condition elements for each of the possible
 934 *Fault States* are shown in *Example 10*:

Example 10: Example of Condition Element Fault States

```

935 1 <Normal type="MOTION_PROGRAM" dataItemId="cc2" sequence="25"
936 2     timestamp="2010-04-06T06:19:35.153141"></Normal>
937 3 <Fault type="COMMUNICATIONS" dataItemId="cc1" sequence="26"
938 4     nativeCode="IO1231" timestamp="2010-04-
939 5     06T06:19:35.153141">Communications error</Fault>
940 6 <Warning type="LOGIC_PROGRAM" dataItemId="pm6" sequence="32"
941 7     timestamp="2010-04-06T06:19:35.153141"><Warning/>
```

942 5.7.2 XML Schema Structure for Condition

943 The XML schema in *Figure 11* represents the structure of a Condition XML element
 944 showing the attributes defined for Condition elements.

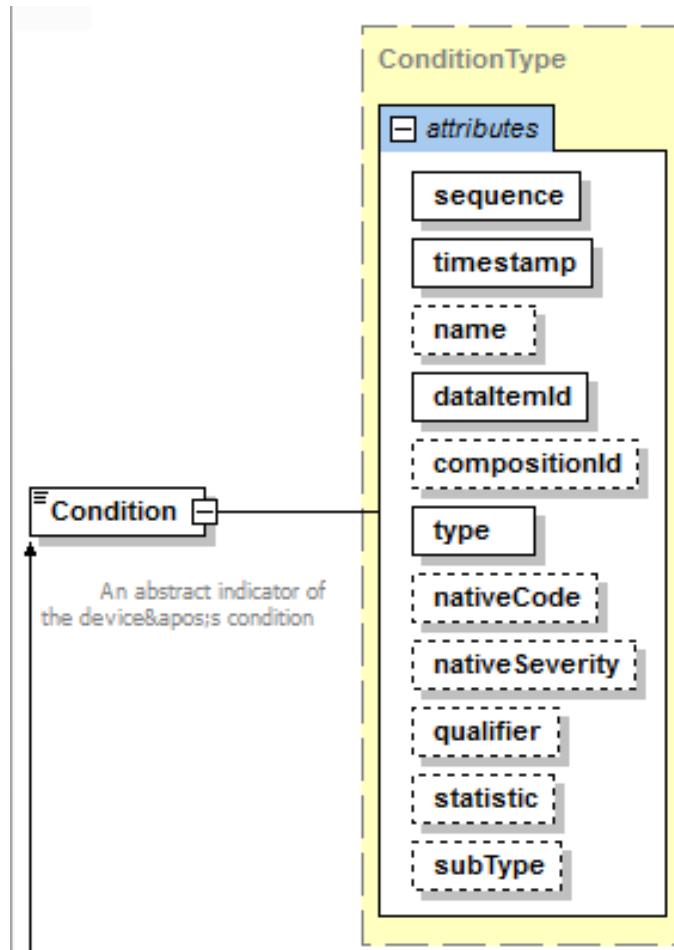


Figure 11: Condition Schema Diagram

945 5.7.3 Attributes for Condition

946 *Table 20* defines the attributes used to provide additional information for a Condition
 947 XML element.

Table 20: Attributes for Condition

| Attribute | Description | Occurrence |
|------------|--|------------|
| sequence | A number representing the sequential position of an occurrence of the Condition in the data buffer of an MTConnect Agent. sequence is a required attribute. sequence MUST have a value represented as an unsigned 64-bit value from 1 to $2^{64} - 1$. | 1 |
| timestamp | The most accurate time available to a piece of equipment that represents the point in time that the data reported for the Condition was measured. timestamp is a required attribute. | 1 |
| name | The name of the Condition element. name is an optional attribute. name MUST match the name attribute of the DataItem element defined in the MTConnectDevices document that the Condition element represents. An NMOKEN XML type. | 0..1 |
| dataItemId | The unique identifier for theCondition element. dataItemId is a required attribute. dataItemId MUST match the id attribute of the DataItem element defined in the MTConnectDevices document that the Condition element represents. | 1 |

| Continuation of Table 20 | | |
|--------------------------|--|------------|
| Attribute | Description | Occurrence |
| type | <p>An identifier of the type of fault represented by the Condition element.</p> <p>type is a required attribute.</p> <p>type MUST match the type attribute of the DataItem element defined in the MTConnectDevices document that this Condition element represents.</p> | 1 |
| nativeCode | <p>The native code (usually an alpha-numeric value) generated by the controller of a piece of equipment providing a reference identifier for a Condition.</p> <p>nativeCode is an optional attribute.</p> <p>This is the same information an operator or maintenance personnel may see as a reference code designating a specific fault code provided by the piece of equipment.</p> | 0..1 |
| nativeSeverity | <p>If the piece of equipment designates a severity level to a fault, nativeSeverity reports that severity information to a client software application.</p> <p>nativeSeverity is an optional attribute.</p> | 0..1 |

| Continuation of Table 20 | | |
|--------------------------|--|------------|
| Attribute | Description | Occurrence |
| qualifier | <p>qualifier provides additional information regarding a <i>Fault State</i> associated with the measured value of a process variable.</p> <p>qualifier is an optional attribute.</p> <p>qualifier defines whether the <i>Fault State</i> represented by the Condition indicates a measured value that is above or below an expected value of a process variable.</p> <p>If the <i>Fault State</i> represents a measured value that is greater than the expected value for the process variable, qualifier MUST report a value of HIGH.</p> <p>If the <i>Fault State</i> represents a measured value that is less than the expected value for the process variable, qualifier MUST report a value of LOW.</p> | 0..1 |
| statistic | <p>statistic provides additional information describing the meaning of the Condition element.</p> <p>statistic is an optional attribute.</p> <p>statistic MUST match the statistic attribute of the DataItem element defined in the MTConnectDevices document that this Condition element represents.</p> | 0..1 |
| subType | <p>subType provides additional information describing the meaning of the Condition element.</p> <p>subType is an optional attribute.</p> <p>subType MUST match the subType attribute of the DataItem element defined in the MTConnectDevices document that this Condition element represents.</p> | 0..1 |

| Continuation of Table 20 | | |
|--------------------------|---|------------|
| Attribute | Description | Occurrence |
| compositionId | The identifier of the Composition element defined in the MTConnectDevices document associated with the data reported for the Condition element. compositionId is an optional attribute. | 0..1 |
| xs:lang | An optional attribute that specifies the language of the CDATA returned for the Condition. Refer to IETF RFC 4646 (http://www.ietf.org/rfc/rfc4646.txt) or successor for a full definition of the values for this attribute. xs:lang does not appear in the schema diagram. | 0..1 |

948 **5.7.3.1 qualifier Attribute for Condition**

949 Many Condition elements report the *Fault State* associated with the measured value of
 950 a process variable.

951 qualifier provides an indication whether the measured value is above or below an
 952 expected value of a process variable.

953 As an example, a Condition element with a type attribute of AMPERAGE may differ-
 954 entiate between a higher than expected amperage and a lower than expected amperage by
 955 using the qualifier attribute.

956 When a qualifier of either HIGH or LOW is used with Fault and Warning, the
 957 *Fault States* can be differentiated as follows:

958 Fault,LOW

959 Warning,LOW

960 Normal

961 Warning,HIGH

962 Fault,HIGH

963 *Example 11* is an example of an XML element representing Condition using qualifi-
964 fier:

Example 11: Example of a Condition Element using qualifier

```
965 1 <Warning type="FILL_LEVEL" dataItemId="pm6"
966 2     qualifier="HIGH" sequence="32"
967 3     timestamp="2009-11-13T08:32:18">...</Warning>
```

968 5.7.4 Valid Data Value for Condition

969 Condition elements reported in an MTConnect Streams XML document **MAY** pro-
970 vide a value in the CDATA of the *Data Entity* when additional information regarding the
971 *Fault State* is available.

972 A *Valid Data Value* for the CDATA included in a Condition element **MAY** be any text
973 string. A *Valid Data Value* is not required to be reported for a Condition category *Data*
974 *Entity*. The *Fault State* and the attributes provided in a Condition element **MAY** be
975 sufficient to fully describe the *Data Entity*.

976 The *Valid Data Value* reported as CDATA for a Condition element **MUST** be formatted
977 as part of the content between the element tags in the XML element representing that *Data*
978 *Entity*. As an example, Condition elements are formatted as shown in *Example 12*:

Example 12: Example of CDATA for Condition

```
979 1 <Warning type="FILL_LEVEL" dataItemId="pm6"
980 2     qualifier="HIGH" sequence="32" timestamp=
981 3     "2009-11-13T08:32:18">Fill Level on Tank
982 4     #12 is reaching a high level</Warning>
```

983 In this example, the “Fill Level on Tank #12 is reaching a high level” is the CDATA for
984 the *Data Entity*.

985 5.8 Unavailability of Fault State for Condition

986 When an *Agent* cannot determine a valid *Fault State* for a Condition element, it **MUST**
987 report the *Element Name* for the *Data Entity* as Unavailable.

988 *Example 13* demonstrates how an *Agent* reports a Condition category *Data Entity* when
989 it is unable to determine a valid *Fault State*:

Example 13: Example of Condition when Fault State is UNAVAILABLE

```
990 1 <Unavailable type="MOTION_PROGRAM" dataItemId="cc2"
991 2     sequence="25" timestamp=
992 3         "2009-11-13T08:32:18">...</Unavailable>
993 4 <Unavailable type="COMMUNICATIONS" dataItemId="cc1"
994 5     sequence="26" timestamp=
995 6         "2009-11-13T08:32:18">...</Unavailable>
996 7 <Unavailable type="LOGIC_PROGRAM" dataItemId="cc3"
997 8     sequence="28" timestamp=
998 9         "2009-11-13T08:32:18">...</Unavailable>
999 10 <Unavailable type="LOGIC_PROGRAM" dataItemId="pm6"
1000 11     sequence="32" timestamp=
1001 12         "2009-11-13T08:32:18">...</Unavailable>
```

1002 6 Listing of Data Entities

1003 *Data Entities* that report data in MTConnectStreams documents are represented by
 1004 Sample, Event, or Condition elements based upon the category and type at-
 1005 tributes defined for the corresponding DataItem XML element in the MTConnectDe-
 1006 vices document.

1007 Each *Data Entity* in the MTConnectStreams document has an *Element Name*, as de-
 1008 fined in the following sections, based upon the corresponding category attribute defined
 1009 for that DataItem element in the MTConnectDevices document.

1010 6.1 Sample Element Names

1011 *Table 21* lists the XML elements that can be placed in the Samples container of the
 1012 ComponentStream element.

1013 The *Table 21* shows both the type attribute for each SAMPLE category DataItem ele-
 1014 ment as defined in the MTConnectDevices document and the corresponding *Element*
 1015 *Name* for the *Data Entity* that **MUST** be reported as a Sample element in the MTCon-
 1016 nectStreams document.

Table 21: Element Names for Sample

| DataItem Type | Element Name | Description |
|---------------|--------------|---|
| ACCELERATION | Acceleration | The measurement of the rate of change of velocity. Acceleration MUST be reported in units of MILLIMETER/SECOND ² . |

| Continuation of Table 21: Element Names for Sample | | |
|--|-----------------|--|
| DataItem Type | Element Name | Description |
| ACCUMULATED_TIME | AccumulatedTime | <p>The measurement of accumulated time for an activity or event.</p> <p>AccumulatedTime MUST be reported in units of MILLIMETER/SECOND².</p> <p>DEPRECATION WARNING : May be deprecated in the future. Recommend using ProcessTimer and EquipmentTimer.</p> |
| AMPERAGE | Amperage | <p>The measurement of electrical current.</p> <p>Subtypes of Amperage are ALTERNATING, DIRECT, ACTUAL, and TARGET.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subType of ACTUAL.</p> <p>Amperage MUST be reported in units of AMPERE.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|---------------------|---|
| DataItem Type | Element Name | Description |
| ANGLE | Angle | <p>The measurement of angular position.</p> <p>Subtypes of Angle are ACTUAL and COMMANDED.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subType of ACTUAL.</p> <p>Angle MUST be reported in units of DEGREE.</p> |
| ANGULAR_ACCELERATION | AngularAcceleration | <p>The measurement rate of change of angular velocity.</p> <p>AngularAcceleration MUST be reported in units of DEGREE/SECOND².</p> |
| ANGULAR_VELOCITY | AngularVelocity | <p>The measurement of the rate of change of angular position.</p> <p>AngularVelocity MUST be reported in units of DEGREE/SECOND.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|-----------------|--|
| DataItem Type | Element Name | Description |
| AXIS_FEEDRATE | AxisFeedrate | <p>The measurement of the feedrate of a linear axis.</p> <p>Subtypes of AxisFeedrate are ACTUAL, COMMANDED, JOG, PROGRAMMED, and RAPID.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subType of PROGRAMMED.</p> <p>AxisFeedrate MUST be reported in units of MILLIMETER/SECOND.</p> |
| CAPACITY_FLUID | CapacityFluid | <p>The fluid capacity of an object or container.</p> <p>CapacityFluid MUST be reported in units of MILLILITER.</p> |
| CAPACITY_SPATIAL | CapacitySpatial | <p>The geometric capacity of an object or container.</p> <p>CapacitySpatial MUST be reported in units of CUBIC_MILLIMETER.</p> |
| CLOCK_TIME | ClockTime | <p>The value provided by a timing device at a specific point in time.</p> <p>ClockTime MUST be reported in W3C ISO 8601 format of yyyy-mm-ddhh:mm:ss.ffff.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|---------------|---|
| DataItem Type | Element Name | Description |
| CONCENTRATION | Concentration | <p>The measurement of the percentage of one component within a mixture of components</p> <p>Concentration MUST be reported in units of PERCENT.</p> |
| CONDUCTIVITY | Conductivity | <p>The measurement of the ability of a material to conduct electricity.</p> <p>Conductivity MUST be reported in units of SIEMENS/METER.</p> |
| CUTTING_SPEED | CuttingSpeed | <p>The speed difference (relative velocity) between the cutting mechanism and the surface of the workpiece it is operating on.</p> <p>Subtypes of CUTTING_SPEED are ACTUAL, COMMANDED, and PROGRAMMED.</p> <p>If no subType is specified, the reported value must default to PROGRAMMED.</p> <p>CuttingSpeed is reported in units of MILLIMETER/SECOND.</p> |
| DENSITY | Density | <p>The volumetric mass of a material per unit volume of that material.</p> <p>Density MUST be reported in units of MILLIGRAM/CUBIC_- MILLIMETER.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|----------------------------------|--|
| DataItem Type | Element Name | Description |
| DEPOSITION_- ACCELERATION_- VOLUMETRIC | DepositionAccelerationVolumetric | <p>The rate of change in spatial volume of material deposited in an additive manufacturing process.</p> <p>Subtypes of DepositionAccelerationVolumetric are ACTUAL and COMMANDED.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subtype of ACTUAL.</p> <p>DepositionAccelerationVolumetric MUST be reported in units of CUBIC_- MILLIMETER/SECOND².</p> |
| DEPOSITION_- DENSITY | DepositionDensity | <p>The density of the material deposited in an additive manufacturing process per unit of volume.</p> <p>Subtypes of DepositionDensity are ACTUAL and COMMANDED.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subtype of ACTUAL.</p> <p>DepositionDensity MUST be reported in units of MILLIGRAM/CUBIC_- MILLIMETER.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|--------------------------|---|
| DataItem Type | Element Name | Description |
| DEPOSITION_MASS | DepositionMass | <p>The mass of the material deposited in an additive manufacturing process.</p> <p>Subtypes of DepositionMass are ACTUAL and COMMANDED.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subtype of ACTUAL.</p> <p>DepositionMass MUST be reported in units of MILLIGRAM.</p> |
| DEPOSITION_-RATE_VOLUMETRIC | DepositionRateVolumetric | <p>The rate at which a spatial volume of material is deposited in an additive manufacturing process.</p> <p>Subtypes of DepositionRateVolumetric are ACTUAL and COMMANDED.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subtype of ACTUAL.</p> <p>DepositionRateVolumetric MUST be reported in units of CUBIC_MILLIMETER/SECOND.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|------------------|---|
| DataItem Type | Element Name | Description |
| DEPOSITION_-VOLUME | DepositionVolume | <p>The spatial volume of material deposited in an additive manufacturing process.</p> <p>Subtypes of DepositionVolume are ACTUAL and COMMANDED.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subtype of ACTUAL.</p> <p>DepositionVolume MUST be reported in units of CUBIC_MILLIMETER.</p> |
| DISPLACEMENT | Displacement | <p>The measurement of the change in position of an object.</p> <p>Displacement MUST be reported in units of MILLIMETER.</p> |
| ELECTRICAL_-ENERGY | ElectricalEnergy | <p>The measurement of electrical energy consumption by a component.</p> <p>ElectricalEnergy MUST be reported in units of WATT_SECOND.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|----------------|--|
| DataItem Type | Element Name | Description |
| EQUIPMENT_TIMER | EquipmentTimer | <p>The measurement of the amount of time a piece of equipment or a sub-part of a piece of equipment has performed specific activities.</p> <p>Subtypes of EquipmentTimer are LOADED, WORKING, OPERATING, POWERED, and DELAY.</p> <p>A subType MUST always be specified.</p> <p>EquipmentTimer MUST be reported in units of SECOND.</p> |
| FILL_LEVEL | FillLevel | <p>The measurement of the amount of a substance remaining compared to the planned maximum amount of that substance.</p> <p>FillLevel MUST be reported in units of PERCENT.</p> |
| FLOW | Flow | <p>The measurement of the rate of flow of a fluid.</p> <p>Flow MUST be reported in units of LITER/SECOND.</p> |
| FREQUENCY | Frequency | <p>The measurement of the number of occurrences of a repeating event per unit time.</p> <p>Frequency MUST be reported in units of HERTZ.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|----------------|---|
| DataItem Type | Element Name | Description |
| GLOBAL_POSITION | GlobalPosition | DEPRECATED in Version 1.1 |
| LENGTH | Length | <p>The measurement of the length of an object.</p> <p>Subtypes of Length are STANDARD, REMAINING, and USEABLE.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subType of REMAINING.</p> <p>Length MUST be reported in units of MILLIMETER.</p> |
| LEVEL | Level | DEPRECATED in Version 1.2. See FILL_LEVEL |
| LINEAR_FORCE | LinearForce | <p>The measurement of the push or pull introduced by an actuator or exerted on an object.</p> <p>LinearForce MUST be reported in units of NEWTON.</p> |
| LOAD | Load | <p>The measurement of the actual versus the standard rating of a piece of equipment.</p> <p>Load MUST be reported in units of PERCENT.</p> |
| MASS | Mass | <p>The measurement of the mass of an object(s) or an amount of material.</p> <p>Mass MUST be reported in units of KILOGRAM.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|---------------------------|---|
| DataItem Type | Element Name | Description |
| PATH_FEEDRATE | PathFeedrate | <p>The measurement of the feedrate for the axes, or a single axis, associated with a Path component-a vector.</p> <p>Subtypes of PathFeedrate are ACTUAL, COMMANDED, JOG, PROGRAMMED, and RAPID.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subType of PROGRAMMED.</p> <p>PathFeedrate MUST be reported in units of MILLIMETER/SECOND.</p> |
| PATH_FEEDRATE_PER_REVOLUTION | PathFeedratePerRevolution | <p>The feedrate for the axes, or a single axis.</p> <p>PathFeedratePerRevolution is reported in units of MILLIMETER/REVOLUTION.</p> <p>Subtypes of PathFeedratePerRevolution are ACTUAL, COMMANDED, and PROGRAMMED.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|--------------|--|
| DataItem Type | Element Name | Description |
| PATH_POSITION | PathPosition | <p>A measured or calculated position of a control point reported by a piece of equipment expressed in WORK coordinates. The coordinate system will revert to MACHINE coordinates if WORK coordinates are not available.</p> <p>Subtypes of PathPosition are ACTUAL, PROGRAMMED, COMMANDED, TARGET, and PROBE.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subtype of ACTUAL.</p> <p>PathPosition MUST be reported as a set of space-delimited floating-point numbers representing a point in 3-D space. The position of the control point MUST be reported in units of MILLIMETER and listed in order of X, Y, and Z referenced to the coordinate system of the piece of equipment.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|--------------|--|
| DataItem Type | Element Name | Description |
| PATH_POSITION (Continued) | PathPosition | An example of the value reported for PathPosition would be: <code><PathPosition ...>10.123 55.232 100.981 </PathPosition></code> Where X = 10.123, Y = 55.232, and Z=100.981. |
| PH | PH | A measure of the acidity or alkalinity of a solution. PH MUST be reported in units of PH. |

| Continuation of Table 21: Element Names for Sample | | |
|--|--------------|--|
| DataItem Type | Element Name | Description |
| POSITION | Position | <p>A measured or calculated position of a Component element as reported by a piece of equipment.</p> <p>Subtypes of Position are ACTUAL, COMMANDED, PROGRAMMED, and TARGET.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subType of ACTUAL.</p> <p>When Position is provided representing a measured value for the physical axes of the piece of equipment, the data MUST be provided in MACHINE coordinates.</p> <p>When Position is provided representing a logical or calculated position, the data MUST be provided in WORK coordinates and is associated with a Path element of the equipment controller.</p> <p>Position MUST be reported in units of MILLIMETER.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|--------------|---|
| DataItem Type | Element Name | Description |
| POWER_FACTOR | PowerFactor | <p>The measurement of the ratio of real power flowing to a load to the apparent power in that AC circuit.</p> <p>PowerFactor MUST be reported in units of PERCENT.</p> |
| PRESSURE | Pressure | <p>The measurement of force per unit area exerted by a gas or liquid. The measurement of force per unit area exerted by a gas or liquid.</p> <p>Pressure MUST be reported in units of PASCAL.</p> |
| PROCESS_TIMER | ProcessTimer | <p>The measurement of the amount of time a piece of equipment has performed different types of activities associated with the process being performed at that piece of equipment.</p> <p>Subtypes of ProcessTimer are PROCESS, and DELAY.</p> <p>A subType MUST always be specified.</p> <p>ProcessTimer MUST be reported in units of SECOND.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|----------------|---|
| DataItem Type | Element Name | Description |
| RESISTANCE | Resistance | <p>The measurement of the degree to which a substance opposes the passage of an electric current.</p> <p>Resistance MUST be reported in units of OHM.</p> |
| ROTARY_VELOCITY | RotaryVelocity | <p>The measurement of the rotational speed of a rotary axis.</p> <p>Subtypes of RotaryVelocity are ACTUAL, COMMANDED and PROGRAMMED.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subType of ACTUAL.</p> <p>RotaryVelocity MUST be reported in units of REVOLUTION/MINUTE.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|--------------|---|
| DataItem Type | Element Name | Description |
| SOUND_LEVEL | SoundLevel | <p>The measurement of a sound level or sound pressure level relative to atmospheric pressure.</p> <p>Subtypes of SoundLevel are NO_SCALE, A_SCALE, B_SCALE, C_SCALE and D_SCALE.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subType of NO_SCALE.</p> <p>SoundLevel MUST be reported in units of DECIBEL.</p> |
| SPINDLE_SPEED | SpindleSpeed | <p>DEPRECATED in Version 1.2. Replaced by ROTARY_VELOCITY</p> |
| STRAIN | Strain | <p>The measurement of the amount of deformation per unit length of an object when a load is applied.</p> <p>Strain MUST be reported in units of PERCENT.</p> |
| TEMPERATURE | Temperature | <p>The measurement of temperature.</p> <p>Temperature MUST be reported in units of CELSIUS.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|--------------|--|
| DataItem Type | Element Name | Description |
| TENSION | Tension | The measurement of a force that stretches or elongates an object. Tension MUST be reported in units of NEWTON. |
| TILT | Tilt | The measurement of angular displacement. Tilt MUST be reported in units of MICRO_RADIAN. |
| TORQUE | Torque | The measurement of the turning force exerted on an object or by an object. Torque MUST be reported in units of NEWTON_METER. |

| Continuation of Table 21: Element Names for Sample | | |
|--|--------------|--|
| DataItem Type | Element Name | Description |
| VELOCITY | Velocity | <p>The measurement of the rate of change of position of a Component.</p> <p>When provided as the Velocity of the Axes Component, it represents the value of the velocity vector for all given axes, similar to PathFeedrate.</p> <p>When provided as the Velocity of an individual Axis Component, it represents the value of the velocity for that specific axis with no influence of the relative velocity of any other axes.</p> <p>Velocity MUST be reported in units of MILLIMETER/SECOND.</p> |
| VISCOSITY | Viscosity | <p>The measurement of a fluids resistance to flow.</p> <p>Viscosity MUST be reported in units of PASCAL_SECOND.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|--------------------|--|
| DataItem Type | Element Name | Description |
| VOLTAGE | Voltage | <p>The measurement of electrical potential between two points.</p> <p>Subtypes of Voltage are ALTERNATING, DIRECT, ACTUAL and TARGET.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subType of ACTUAL.</p> <p>Voltage MUST be reported in units of VOLT.</p> |
| VOLT_AMPERE | VoltAmpere | <p>The measurement 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).</p> <p>VoltAmpere MUST be reported in units of VOLT_AMPERE.</p> |
| VOLT_AMPERE_-REACTIVE | VoltAmpereReactive | <p>The measurement of reactive power in an AC electrical circuit (commonly referred to as VAR).</p> <p>VoltAmpereReactive MUST be reported in units of VOLT_AMPERE_-REACTIVE.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|---------------|---|
| DataItem Type | Element Name | Description |
| VOLUME_FLUID | VolumeFluid | <p>The fluid volume of an object or container.</p> <p>Subtypes of VolumeFluid are ACTUAL and CONSUMED.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subtype of ACTUAL.</p> <p>VolumeFluid MUST be reported in units of MILLILITER.</p> |
| VOLUME_SPATIAL | VolumeSpatial | <p>The geometric volume of an object or container.</p> <p>Subtypes of VolumeSpatial are ACTUAL and CONSUMED.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subtype of ACTUAL.</p> <p>VolumeSpatial MUST be reported in units of CUBIC_MILLIMETER.</p> |

| Continuation of Table 21: Element Names for Sample | | |
|--|--------------|--|
| DataItem Type | Element Name | Description |
| WATTAGE | Wattage | <p>The measurement of power flowing through or dissipated by an electrical circuit or piece of equipment.</p> <p>Subtypes of Wattage are ACTUAL and TARGET.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subType of ACTUAL.</p> <p>Wattage MUST be reported in units of WATT.</p> |

1017 Note: The Sample response format **MUST** be extended when the representation attribute for the data item is TIME_SERIES. See *Section 5.3.3 -*
 1018 *Response for SAMPLE category DataItem Elements with a representation At-*
 1019 *tribute of TIME_SERIES* for details on extending the response format.
 1020

1021 6.2 Event Element Names

1022 *Table 22* lists the XML elements that can be placed in the Events container of the Com-
 1023 ponentStream element.

1024 The *Table 21* shows both the type for each EVENT category DataItem element defined
 1025 in the MTConnectDevices document and the corresponding *Element Name* for the
 1026 *Data Entity* that **MUST** be reported as an Event element in the MTConnectStreams
 1027 document.

1028 The table also defines the *Valid Data Value* for those Event type data items where the
 1029 reported values are restricted to a *Controlled Vocabulary*.

Table 22: Element Names for Event

| DataItem Type | Element Name | Description |
|---------------|--------------|---|
| ACTIVE_AXES | ActiveAxes | <p>The set of axes currently associated with a Path or Controller <i>Structural Element</i>.</p> <p>The <i>Valid Data Value</i> reported SHOULD be a space-delimited set of axes names. The names returned SHOULD match the name attribute of the Linear or Rotary <i>Structural Elements</i> defined in the MTConnectDevices document that this Event element represents. If name is not available, nativeName MUST be returned to identify the Linear or Rotary <i>Structural Elements</i>.</p> <p>For example:</p> <pre><ActiveAxes ...>X Y Z W S</ActiveAxes></pre> <p>where X, Y, Z, W, and S are the nativeName attributes of the <i>Structural Elements</i>.</p> <p>If it is not specified elsewhere in the MTConnectDevices document, it MUST be assumed that all of the axes are associated with the Path component.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|---------------|---|
| DataItem Type | Element Name | Description |
| ACTUATOR_-STATE | ActuatorState | <p>Represents the operational state of an apparatus for moving or controlling a mechanism or system.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> ACTIVE: The actuator is operating INACTIVE: The actuator is not operating |
| ALARM | Alarm | <p>DEPRECATED : Replaced with CONDITION category data items in Version 1.1.0.</p> |
| AVAILABILITY | Availability | <p>Represents the <i>Agent's</i> ability to communicate with the data source.</p> <p>Availability MUST be provided for each <i>Device Structural Element</i> and MAY be provided for any other <i>Structural Element</i>.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> AVAILABLE: The <i>Structural Element</i> is active and capable of providing data. AVAILABLE: The <i>Structural Element</i> is either inactive or not capable of providing data. |

| Continuation of Table 22: Element Names for Event | | |
|---|--------------|---|
| DataItem Type | Element Name | Description |
| AXIS_-COUPLING | AxisCoupling | <p>Describes the way the axes will be associated to each other.</p> <p>This is used in conjunction with COUPLED_AXES to indicate the way they are interacting.</p> <p>The coupling of the axes MUST be viewed from the perspective of a specified axis. Therefore, a MASTER coupling indicates that this axis is the master for the COUPLED_AXES.</p> <p>AxisCoupling MUST be provided for each axis element associated with a set of axes defined by the COUPLED_AXES data item element defined in the MTConnectDevices document.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> TANDEM: The axes are physically connected to each other and operate as a single unit. SYNCHRONOUS: The axes are not physically connected to each other but are operating together in lockstep. MASTER: The axis is the master of the CoupledAxes SLAVE: The axis is a slave to the CoupledAxes |

| Continuation of Table 22: Element Names for Event | | |
|---|----------------------|---|
| DataItem Type | Element Name | Description |
| AXIS_- FEEDRATE_- OVERRIDE | AxisFeedrateOverride | <p>The value of a signal or calculation issued to adjust the feedrate of an individual linear type axis.</p> <p>The value provided for AxisFeedrateOverride is expressed as a percentage of the designated feedrate for the axis.</p> <p>Subtypes of AxisFeedrateOverride are JOG, PROGRAMMED, and RAPID.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subType of PROGRAMMED.</p> <p>The <i>Valid Data Value MUST</i> be a floating-point number.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|---------------|---|
| DataItem Type | Element Name | Description |
| AXIS_-INTERLOCK | AxisInterlock | <p>An indicator of the state of the axis lockout function when power has been removed and the axis is allowed to move freely.</p> <p><i>Valid Data Values:</i></p> <p>ACTIVE: The axis lockout function is activated, power has been removed from the axis, and the axis is allowed to move freely.</p> <p>INACTIVE: The axis lockout function has not been activated, the axis may be powered, and the axis is capable of being controlled by another component.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|--------------|---|
| DataItem Type | Element Name | Description |
| AXIS_STATE | AxisState | <p>An indicator of the controlled state of a Linear or Rotary component representing an axis.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> HOME: The axis is in its home position. TRAVEL: The axis is in motion PARKED: The axis has been moved to a fixed position and is being maintained in that position either electrically or mechanically. Action is required to release the axis from this position. STOPPED: The axis is stopped |
| BLOCK | Block | <p>The line of code or command being executed by a Controller <i>Structural Element</i>.</p> <p>Block MUST include the entire expression for a line of program code, including all parameters</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p> |
| BLOCK_COUNT | BlockCount | <p>The total count of the number of blocks of program code that have been executed since execution started.</p> <p>The <i>Valid Data Value</i> MUST be an integer.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|----------------|---|
| DataItem Type | Element Name | Description |
| CHUCK_-INTERLOCK | ChuckInterlock | <p>An indication of the state of an interlock function or control logic state intended to prevent the associated CHUCK component from being operated.</p> <p>A CHUCK component or composition element may be controlled by more than one type of ChuckInterlock function. When the</p> <p>ChuckInterlock function is provided by an operator controlled interlock that can inhibit the ability to initiate an unclamp action of an electronically controlled chuck, this</p> <p>ChuckInterlock function SHOULD be further characterized by specifying a subType of MANUAL_UNCLAMP.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> ACTIVE: The chuck cannot be unclamped INACTIVE: The chuck can be unclamped. |

| Continuation of Table 22: Element Names for Event | | |
|---|--------------|---|
| DataItem Type | Element Name | Description |
| CHUCK_STATE | ChuckState | <p>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 piece of equipment.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> OPEN: The CHUCK component or composition element is open to the point of a positive confirmation CLOSED: The CHUCK component or composition element is closed to the point of a positive confirmation UNLATCHED: The CHUCK component or composition element is not closed to the point of a positive confirmation and not open to the point of a positive confirmation. It is in an intermediate position. |
| CODE | Code | DEPRECATED in Version 1.1. |

| Continuation of Table 22: Element Names for Event | | |
|---|------------------|--|
| DataItem Type | Element Name | Description |
| COMPOSITION_-STATE | CompositionState | <p>An indication of the operating condition of a mechanism represented by a Composition type element.</p> <p>Subtypes of CompositionState are ACTION, LATERAL, MOTION, SWITCHED, and VERTICAL.</p> <p>A subType MUST be provided.</p> <p><i>Valid Data Values</i> for subType ACTION are:</p> <ul style="list-style-type: none"> ACTIVE: The Composition element is operating INACTIVE: The Composition element is not operating. <p><i>Valid Data Values</i> for subType LATERAL are:</p> <ul style="list-style-type: none"> RIGHT : The position of the Composition element is oriented to the right to the point of a positive confirmation LEFT : The position of the Composition element is oriented to the left to the point of a positive confirmation |

| Continuation of Table 22: Element Names for Event | | |
|---|------------------|---|
| DataItem Type | Element Name | Description |
| COMPOSITION_-STATE (Continued) | CompositionState | <p><i>Valid Data Values</i> for subType SWITCHED are:</p> <p>ON : The activation state of the Composition element is in an ON condition, it is operating, or it is powered.</p> <p>OFF : The activation state of the Composition element is in an OFF condition, it is not operating, or it is not powered. <i>Valid Data Values</i> for subType VERTICAL are:</p> <p>UP : The position of the Composition element is oriented in an upward direction to the point of a positive confirmation</p> <p>DOWN : The position of the Composition element is oriented in a downward direction to the point of a positive confirmation</p> <p>TRANSITIONING : The position of the Composition element is not oriented in an upward direction to the point of a positive confirmation and is not oriented in a downward direction to the point of a positive confirmation. It is in an intermediate position.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|------------------|--|
| DataItem Type | Element Name | Description |
| COMPOSITION_-STATE (Continued) | CompositionState | <p>TRANSITIONING : The position of the Composition element is not oriented to the right to the point of a positive confirmation and is not oriented to the left to the point of a positive confirmation. It is in an intermediate position.</p> <p><i>Valid Data Values for subType MOTION are:</i></p> <ul style="list-style-type: none"> OPEN: The position of the Composition element is open to the point of a positive confirmation CLOSED: The position of the Composition element is closed to the point of a positive confirmation UNLATCHED: The position of the Composition element is not open to the point of a positive confirmation and is not closed to the point of a positive confirmation. It is in an intermediate position. |

| Continuation of Table 22: Element Names for Event | | |
|---|----------------|---|
| DataItem Type | Element Name | Description |
| CONTROLLER_- MODE | ControllerMode | <p>The current operating mode of the Controller component.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> AUTOMATIC: The controller is configured to automatically execute a program. MANUAL: The controller is not executing an active program. It is capable of receiving instructions from an external source – typically an operator. The controller executes operations based on the instructions received from the external source. MANUAL_DATA_INPUT: The operator can enter a series of operations for the controller to perform. The controller will execute this specific series of operations and then stop. SEMI_AUTOMATIC: The controller is operating in a mode that restricts the active program from processing its next process step without operator intervention. EDIT: The controller is currently functioning as a programming device and is not capable of executing an active program. |

| Continuation of Table 22: Element Names for Event | | |
|---|------------------------|---|
| DataItem Type | Element Name | Description |
| CONTROLLER_- MODE_- OVERRIDE | ControllerModeOverride | <p>A setting or operator selection that changes the behavior of a piece of equipment.</p> <p>Subtypes of ControllerModeOverride are DRY_RUN, SINGLE_BLOCK, MACHINE_AXIS_LOCK, OPTIONAL_STOP, and TOOL_CHANGE_STOP.</p> <p>A subType MUST always be specified.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> ON : The indicator of the ControllerModeOverride is in the ON state and the mode override is active. OFF : The indicator of the ControllerModeOverride is in the OFF state and the mode override is inactive |

| Continuation of Table 22: Element Names for Event | | |
|---|--------------|--|
| DataItem Type | Element Name | Description |
| COUPLED_AXES | CoupledAxes | <p>Refers to the set of associated axes.</p> <p>Used in conjunction with AxisCoupling to describe how the CoupledAxes relate to each other.</p> <p>The <i>Valid Data Value</i> reported SHOULD be a space-delimited set of axes names. The names returned SHOULD match the name attribute of the Linear or Rotary <i>Structural Elements</i> defined in the MTConnectDevices document that this Event element represents. If name is not available, nativeName MUST be returned to identify the Linear or Rotary <i>Structural Elements</i>.</p> <p>Example:</p> <pre><CoupledAxes ...>Y1 Y2</CoupledAxes></pre> |
| DATE_CODE | DateCode | <p>The time and date code associated with a material or other physical item.</p> <p>Subtypes of DateCode are MANUFACTURE, EXPIRATION, and FIRST_USE.</p> <p>A subType MUST always be specified.</p> <p>DateCode MUST be reported in ISO 8601 format.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|--------------|--|
| DataItem Type | Element Name | Description |
| DEVICE_UUID | DeviceUuid | <p>The identifier of another piece of equipment that is temporarily associated with a component of this piece of equipment to perform a particular function.</p> <p><i>Valid Data Values</i> are the value of the UUID attribute of the associated device - a NMOKEN XML type.</p> |
| DIRECTION | Direction | <p>The direction of motion.</p> <p>Subtypes of Direction are ROTARY and LINEAR.</p> <p>A subType MUST always be specified. <i>Valid Data Values</i> for subType ROTARY are:</p> <ul style="list-style-type: none"> CLOCKWISE : A Rotary type component is rotating in a clockwise fashion using the right-hand rule. COUNTER_CLOCKWISE : A Rotary type component is rotating in a counter clockwise fashion using the right-hand rule. <i>Valid Data Values</i> for subType LINEAR are: POSITIVE : A Linear type component is moving in the direction of increasing position value NEGATIVE : A Linear type component is moving in the direction of decreasing position value |

| Continuation of Table 22: Element Names for Event | | |
|---|---------------|---|
| DataItem Type | Element Name | Description |
| DOOR_STATE | DoorState | <p>The operational state of a DOOR type component or composition element.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> OPEN: The DOOR is open to the point of a positive confirmation CLOSED: The DOOR is closed to the point of a positive confirmation UNLATCHED: The DOOR is not closed to the point of a positive confirmation and is not open to the point of a positive confirmation. It is in an intermediate position. |
| EMERGENCY_STOP | EmergencyStop | <p>The current state of the emergency stop signal for a piece of equipment, controller path, or any other component or subsystem of a piece of equipment.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> ARMED : The emergency stop circuit is complete and the piece of equipment, component, or composition element is allowed to operate. TRIGGERED : The emergency stop circuit is open and the operation of the piece of equipment, component, or composition element is inhibited. |

| Continuation of Table 22: Element Names for Event | | |
|---|---------------|---|
| DataItem Type | Element Name | Description |
| END_OF_BAR | EndOfBar | <p>An indication of whether the end of a piece of bar stock being feed by a bar feeder has been reached.</p> <p>Subtypes of EndOfBar are PRIMARY and AUXILIARY.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subType of PRIMARY.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> YES : The EndOfBar has been reached. NO : The EndOfBar has not been reached. |
| EQUIPMENT__MODE | EquipmentMode | <p>An indication that a piece of equipment, or a sub-part of a piece of equipment, is performing specific types of activities.</p> <p>Subtypes of EquipmentMode are LOADED, WORKING, OPERATING, and POWERED.</p> <p>A subType MUST always be specified.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> ON : The equipment is functioning in the mode designated by the subType. OFF : The equipment is not functioning in the mode designated by the subType. |

| Continuation of Table 22: Element Names for Event | | |
|---|--------------|---|
| DataItem Type | Element Name | Description |
| EXECUTION | Execution | <p>The execution status of the Controller component.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> READY: The controller is ready to execute instructions. It is currently idle. ACTIVE: The controller is actively executing an instruction. INTERRUPTED: The execution of the controller's program has been suspended due to an external signal. Action is required to resume execution. WAIT: The execution of the controller's program is suspended while a secondary operation is executing or completing. Execution will resume automatically once the secondary operation is completed. FEED_HOLD: Motion of the device has been commanded to stop at its current position. The controller remains able to execute instructions but cannot complete the current set of instructions until after motion resumes. The command to stop the motion must be removed before execution can resume. |

| Continuation of Table 22: Element Names for Event | | |
|---|--------------|---|
| DataItem Type | Element Name | Description |
| EXECUTION (Continued) | Execution | <p>STOPPED: The execution of the controller's program has been stopped in an unplanned manner and execution of the program cannot be resumed without intervention by an operator or external signal.</p> <p>OPTIONAL_STOP: The controller's program has been intentionally stopped using an M01 or similar command. The program may be stopped at the designated location based upon the state of a secondary indication provided to the controller indicating whether the program execution must be stopped at this location or program execution should continue.</p> <p>PROGRAM_STOPPED: The execution of the controller's program has been stopped by a command from within the program. Action is required to resume execution.</p> <p>PROGRAM_COMPLETED: The program has completed execution.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|----------------|--|
| DataItem Type | Element Name | Description |
| FUNCTIONAL_MODE | FunctionalMode | <p>The current intended production status of the device or component.</p> <p>Typically, the FunctionalMode SHOULD be associated with the Device <i>Structural Element</i>, but it MAY be associated with any <i>Structural Element</i> in the XML document.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> PRODUCTION : The Device element or another <i>Structural Element</i> is currently producing product, ready to produce product, or its current intended use is to be producing product. SETUP : The Device element or another <i>Structural Element</i> is not currently producing product. It is being prepared or modified to begin production of product. TEARDOWN : The Device element or another <i>Structural Element</i> is not currently producing product. Typically, it has completed the production of a product and is being modified or returned to a neutral state such that it may then be prepared to begin production of a different product. |

| Continuation of Table 22: Element Names for Event | | |
|---|----------------|--|
| DataItem Type | Element Name | Description |
| FUNCTIONAL_MODE (Continued) | FunctionalMode | <p>MAINTENANCE : The Device element or another <i>Structural Element</i> is not currently producing product. It is currently being repaired, waiting to be repaired, or has not yet been returned to a normal production status after maintenance has been performed.</p> <p>PROCESS_DEVELOPMENT : The Device element or another <i>Structural Element</i> is being used to prove-out a new process, testing of equipment or processes, or any other active use that does not result in the production of product.</p> |
| HARDNESS | Hardness | <p>The measurement of the hardness of a material.</p> <p>Subtypes of Hardness are ROCKWELL, VICKERS, SHORE, BRINELL, LEEB, and MOHS.</p> <p>A subType MUST always be specified.</p> <p>The <i>Valid Data Value</i> MUST be a floating-point number.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|----------------|---|
| DataItem Type | Element Name | Description |
| INTERFACE__STATE | InterfaceState | <p>The current functional or operational state of an <i>Interface</i> type element indicating whether the <i>Interface</i> is active or not currently functioning.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> ENABLED: The <i>Interface</i> is currently operational and performing as expected. DISABLED: The <i>Interface</i> is currently not operational. <p>When the INTERFACE_STATE is DISABLED, the state of all data items that are specific for the <i>Interaction Model</i> associated with that <i>Interface</i> MUST be set to NOT_READY.</p> |
| LINE | Line | DEPRECATED in Version 1.4.0. |
| LINE_LABEL | LineLabel | <p>An optional identifier for a BLOCK of code in a PROGRAM.</p> <p>The <i>Valid Data Value</i> MUST be any text string.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|---------------|---|
| DataItem Type | Element Name | Description |
| LINE_NUMBER | LineNumber | <p>A reference to the position of a block of program code within a control program.</p> <p>Subtypes of LineNumber are ABSOLUTE and INCREMENTAL.</p> <p>A subType MUST always be specified.</p> <p>The <i>Valid Data Value</i> MUST be an integer.</p> |
| MATERIAL | Material | <p>The identifier of a material used or consumed in the manufacturing process.</p> <p>The <i>Valid Data Value</i> MUST be any text string.</p> |
| MATERIAL_LAYER | MaterialLayer | <p>Designates the layers of material applied to a part or product as part of an additive manufacturing process.</p> <p>Subtypes of MaterialLayer are ACTUAL and TARGET.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subtype of ACTUAL.</p> <p>The <i>Valid Data Value</i> MUST be an integer.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|--------------|---|
| DataItem Type | Element Name | Description |
| MESSAGE | Message | <p>Any text string of information to be transferred from a piece of equipment to a client software application.</p> <p>The <i>Valid Data Value</i> MUST be any text string.</p> |
| OPERATOR_ID | OperatorId | <p>The identifier of the person currently responsible for operating the piece of equipment.</p> <p>The <i>Valid Data Value</i> MAY be any text string.</p> <p>DEPRECATION WARNING : May be deprecated in the future. See USER below.</p> |
| PALLET_ID | PalletId | <p>The identifier for a pallet.</p> <p>The <i>Valid Data Value</i> MAY be any text string.</p> |
| PART_COUNT | PartCount | <p>The current count of parts produced as represented by the Controller component.</p> <p>Subtypes of PartCount are ALL, GOOD, BAD, TARGET, and REMAINING.</p> <p>PartCount will not be accumulated by an Agent and MUST only be supplied if the Controller provides the count.</p> <p>The <i>Valid Data Value</i> MUST be a floating-point number, usually an integer.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|--------------|--|
| DataItem Type | Element Name | Description |
| PART_DETECT | PartDetect | <p>An indication designating whether a part or work piece has been detected or is present.</p> <p>The <i>Valid Data Value</i> MUST be:</p> <ul style="list-style-type: none"> PRESENT: if a part or work piece has been detected or is present. NOT_PRESENT: if a part or work piece is not detected or is not present. |
| PART_ID | PartId | <p>An identifier of a part in a manufacturing operation.</p> <p>The <i>Valid Data Value</i> MAY be any text string.</p> |
| PART_NUMBER | PartNumber | <p>An identifier of a part or product moving through the manufacturing process.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p> <p>DEPRECATION WARNING : May be deprecated in the future.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|----------------------|---|
| DataItem Type | Element Name | Description |
| PATH_- FEEDRATE_- OVERRIDE | PathFeedrateOverride | <p>The value of a signal or calculation issued to adjust the feedrate for the axes associated with a Path component that may represent a single axis or the coordinated movement of multiple axes.</p> <p>The value provided for PathFeedrateOverride is expressed as a percentage of the designated feedrate for the path.</p> <p>Sub-types of PathFeedrateOverride are JOG, PROGRAMMED, and RAPID.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subType of PROGRAMMED.</p> <p>The <i>Valid Data Value</i> MUST be a floating-point number.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|--------------|--|
| DataItem Type | Element Name | Description |
| PATH_MODE | PathMode | <p>Describes the operational relationship between a <i>Path Structural Element</i> and another <i>Path Structural Element</i> for pieces of equipment comprised of multiple logical groupings of controlled axes or other logical operations.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> INDEPENDENT : The path is operating independently and without the influence of another path. MASTER: The path provides the reference motion for a SYNCHRONOUS or MIRROR type path to follow. For non-motion type paths, the MASTER provides information or state values that influences the operation of other paths SYNCHRONOUS: The axes associated with the path are following the motion of the MASTER type path. MIRROR : The axes associated with the path are mirroring the motion of the MASTER path. When PathMode is not specified, the operational mode of the path MUST be interpreted as INDEPENDENT . |

| Continuation of Table 22: Element Names for Event | | |
|---|--------------|---|
| DataItem Type | Element Name | Description |
| POWER_STATE | PowerState | <p>The indication of the status of the source of energy for a <i>Structural Element</i> to allow it to perform its intended function or the state of an enabling signal providing permission for the <i>Structural Element</i> to perform its functions.</p> <p>Subtypes of PowerState are LINE and CONTROL.</p> <p>When the subType is LINE, PowerState represents the primary source of energy for a <i>Structural Element</i>.</p> <p>When the subType is CONTROL, PowerState represents an enabling signal providing permission for the <i>Structural Element</i> to perform its function(s).</p> <p>If a subType is not specified, the reported value for the data MUST default to the subType of LINE.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|--------------|---|
| DataItem Type | Element Name | Description |
| POWER_STATE (Continued) | PowerState | <p><i>Valid Data Values:</i></p> <p>ON : The source of energy for a <i>Structural Element</i> or the enabling signal providing permission for the <i>Structural Element</i> to perform its function(s) is present and active.</p> <p>OFF : The source of energy for a <i>Structural Element</i> or the enabling signal providing permission for the <i>Structural Element</i> to perform its function(s) is not present or is disconnected.</p> <p>DEPRECATION WARNING : PowerState may be deprecated in the future.</p> |
| POWER_STATUS | PowerStatus | DEPRECATED in Version 1.1.0. |
| PROCESS_TIME | ProcessTime | <p>The time and date associated with an activity or event.</p> <p>Subtypes of ProcessTime are START, COMPLETE, and TARGET_COMPLETION.</p> <p>A subType MUST always be specified.</p> <p>ProcessTime MUST be reported in ISO 8601 format.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|----------------|---|
| DataItem Type | Element Name | Description |
| PROGRAM | Program | <p>The identity of the logic or motion program being executed.</p> <p>The <i>Valid Data Value</i> MUST be any text string.</p> <p>Subtypes of PROGRAM are SCHEDULE, MAIN and ACTIVE.</p> <p>If a subType is not specified, it is assumed to be MAIN.</p> |
| PROGRAM_COMMENT | ProgramComment | <p>A comment or non-executable statement in the control program.</p> <p>The <i>Valid Data Value</i> MUST be any text string.</p> <p>Subtypes of PROGRAM_COMMENT are SCHEDULE, MAIN and ACTIVE.</p> <p>If a subType is not specified, it is assumed to be MAIN.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|-----------------|--|
| DataItem Type | Element Name | Description |
| PROGRAM_EDIT | ProgramEdit | <p>An indication of the status of the Controller components program editing mode.</p> <p>On many controls, a program can be edited while another program is currently being executed.</p> <p>ProgramEdit provides an indication of whether the controller is being used to edit programs in either case.</p> <p><i>Valid Data Values:</i></p> <p>ACTIVE: The controller is in the program edit mode.</p> <p>READY : The controller is capable of entering the program edit mode and no function is inhibiting a change to that mode.</p> <p>NOT_READY : A function is inhibiting the controller from entering the program edit mode.</p> |
| PROGRAM_EDIT_NAME | ProgramEditName | <p>The name of the program being edited.</p> <p>This is used in conjunction with PROGRAM_EDIT when in ACTIVE state.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|-----------------|--|
| DataItem Type | Element Name | Description |
| PROGRAM_- HEADER | ProgramHeader | <p>The non-executable header section of the control program.</p> <p>The content SHOULD be limited to 512 bytes.</p> <p>The <i>Valid Data Value</i> MUST be any text string.</p> |
| PROGRAM_- LOCATION | ProgramLocation | <p>The Uniform Resource Identifier (URI) for the source file associated with PROGRAM.</p> <p>The <i>Valid Data Value</i> MUST be any text string.</p> <p>A subType MUST always be specified.</p> <p>Subtypes of PROGRAM_LOCATION are SCHEDULE, MAIN, and ACTIVE.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|---------------------|---|
| DataItem Type | Element Name | Description |
| PROGRAM_-LOCATION_-TYPE | ProgramLocationType | <p>Defines whether the logic or motion program defined by PROGRAM is being executed from the local memory of the controller or from an outside source.</p> <p>A subType MUST always be specified.</p> <p>Subtypes of PROGRAM_-LOCATION_TYPE are SCHEDULE, MAIN, and ACTIVE.</p> <p><i>Valid Data Values</i> are:</p> <ul style="list-style-type: none"> LOCAL: Managed by the controller. EXTERNAL: Not managed by the controller. |
| PROGRAM_-NEST_LEVEL | ProgramNestLevel | <p>An indication of the nesting level within a control program that is associated with the code or instructions that is currently being executed.</p> <p>If an initial value is not defined, the nesting level associated with the highest or initial nesting level of the program MUST default to zero (0).</p> <p>The value reported for ProgramNestLevel MUST be an integer.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|------------------------|--|
| DataItem Type | Element Name | Description |
| ROTARY_MODE | RotaryMode | <p>The current operating mode for a Rotary type axis.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> SPINDLE: The axis is functioning as a spindle. Generally, it is configured to rotate at a defined speed. INDEX: The axis is configured to index to a set of fixed positions or to incrementally index by a fixed amount. CONTOUR: The position of the axis is being interpolated as part of the PathPosition defined by the Controller <i>Structural Element</i>. |
| ROTARY_- VELOCITY_- OVERRIDE | RotaryVelocityOverride | <p>The value of a command issued to adjust the programmed velocity for a Rotary type axis.</p> <p>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.</p> <p>RotaryVelocityOverride is expressed as a percentage of the programmed RotaryVelocity.</p> <p>The <i>Valid Data Value MUST</i> be a floating-point number.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|------------------|---|
| DataItem Type | Element Name | Description |
| SERIAL_-NUMBER | SerialNumber | The serial number associated with a Component, Asset, or Device. The <i>Valid Data Value</i> MUST be a text string. |
| SPINDLE_-INTERLOCK | SpindleInterlock | An indication of the status of the spindle for a piece of equipment when power has been removed and it is free to rotate. <i>Valid Data Values:</i> ACTIVE: Power has been removed and the spindle cannot be operated. INACTIVE: Spindle has not been deactivated. |
| TOOL_ASSET_-ID | ToolAssetId | The identifier of an individual tool asset. The <i>Valid Data Value</i> MUST be a text string. |
| TOOL_GROUP | ToolGroup | An identifier for the tool group associated with a specific tool. Commonly used to designate spare tools. The <i>Valid Data Value</i> MUST be any text string. |
| TOOL_ID | ToolId | DEPRECATED in Version 1.2.0. See TOOL_ASSET_ID. The identifier of the tool currently in use for a given Path. |

| Continuation of Table 22: Element Names for Event | | |
|---|--------------|---|
| DataItem Type | Element Name | Description |
| TOOL_NUMBER | ToolNumber | <p>The identifier assigned by the Controller component to a cutting tool when in use by a piece of equipment.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p> |
| TOOL_OFFSET | ToolOffset | <p>A reference to the tool offset variables applied to the active cutting tool.</p> <p>Subtypes of ToolOffset are RADIAL and LENGTH.</p> <p>DEPRECATED in V1.5 A subType MUST always be specified.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p> |
| USER | User | <p>The identifier of the person currently responsible for operating the piece of equipment.</p> <p>Subtypes of User are OPERATOR, MAINTENANCE, and SET_UP.</p> <p>A subType MUST always be specified.</p> <p>The <i>Valid Data Value</i> MUST be any text string.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|--------------|---|
| DataItem Type | Element Name | Description |
| VARIABLE | Variable | A data value whose meaning may change over time due to changes in the operation of a piece of equipment or the process being executed on that piece of equipment. The <i>Valid Data Value</i> MUST be a string. |

| Continuation of Table 22: Element Names for Event | | |
|---|--------------|---|
| DataItem Type | Element Name | Description |
| WAIT_STATE | WaitState | <p>An indication of the reason that EXECUTION is reporting a value of WAIT.</p> <p><i>Valid Data Values</i> are:</p> <ul style="list-style-type: none"> POWERING_UP: An indication that execution is waiting while the equipment is powering up and is not currently available to begin producing parts or products. POWERING_DOWN: An indication that the execution is waiting while the equipment is powering down but has not fully reached a stopped state. PART_LOAD: An indication that the execution is waiting while one or more discrete workpieces are being loaded. PART_UNLOAD: An indication that the execution is waiting while one or more discrete workpieces are being unloaded. TOOL_LOAD: An indication that the execution is waiting while a tool or tooling is being loaded. TOOL_UNLOAD: An indication that the execution is waiting while a tool or tooling is being unloaded. |

| Continuation of Table 22: Element Names for Event | | |
|---|--------------|--|
| DataItem Type | Element Name | Description |
| WAIT_STATE (Continued) | WaitState | <p>MATERIAL_LOAD: An indication that the execution is waiting while bulk material or the container for bulk material used in the production process is being loaded. Bulk material includes those materials from which multiple workpieces may be created.</p> <p>MATERIAL_UNLOAD: An indication that the execution is waiting while bulk material or the container for bulk material used in the production process is being unloaded. Bulk material includes those materials from which multiple workpieces may be created.</p> <p>SECONDARY_PROCESS: An indication that the execution is waiting while another process is completed before the execution can resume.</p> <p>PAUSING: An indication that the execution is waiting while the equipment is pausing but the piece of equipment has not yet reached a fully paused state.</p> <p>RESUMING: An indication that the execution is waiting while the equipment is resuming the production cycle but has not yet resumed execution.</p> |

| Continuation of Table 22: Element Names for Event | | |
|---|---------------|---|
| DataItem Type | Element Name | Description |
| WIRE | Wire | <p>The identifier for the type of wire used as the cutting mechanism in Electrical Discharge Machining or similar processes.</p> <p>The <i>Valid Data Value</i> MUST be any text string.</p> |
| WORKHOLDING_ID | WorkholdingId | <p>The identifier for the current workholding or part clamp in use by a piece of equipment.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p> |
| WORK_OFFSET | WorkOffset | <p>A reference to the offset variables for a work piece or part associated with a Path in a Controller type component.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p> |

1030 6.3 Types of Condition Elements

1031 As described in *Section 5.7 - Condition Data Entity*, Condition *Data Entities* are re-
 1032 ported differently from other data item types. They are reported based on the *Fault State*
 1033 for each Condition. Unlike Sample and Event data items that are identified by their
 1034 *Element Name*, Condition data items are defined by the *type* and *subType* (where
 1035 applicable) attributes defined for each Condition.

1036 The *type* and *subType* (where applicable) attributes for a Condition element **MAY**
 1037 be any of the *type* and *subType* attributes defined for SAMPLE category or EVENT
 1038 category data item listed in the *Devices Information Model*.

1039 Table *Section 5.7.1 - Element Names for Condition* lists additional Condition *Data En-*
 1040 *tities* that have been defined to represent the health and fault status of *Structural Elements*.
 1041 The table defines the *type* attribute for each of these additional Condition category

1042 elements that **MAY** be reported in the MTConnect Streams document.

Table 23: Element Names for Condition

| DataItem Type | Description |
|-----------------|--|
| ACTUATOR | An indication of a fault associated with an actuator. |
| CHUCK_INTERLOCK | An indication of the operational condition of the interlock function for an electronically controller chuck. |
| COMMUNICATIONS | An indication that the piece of equipment has experienced a communications failure. |
| DATA_RANGE | An indication that the value of the data associated with a measured value or a calculation is outside of an expected range. |
| DIRECTION | An indication of a fault associated with the direction of motion of a <i>Structural Element</i> . |
| END_OF_BAR | An indication that the end of a piece of bar stock has been reached. |
| HARDWARE | An indication of a fault associated with the hardware subsystem of the <i>Structural Element</i> . |
| INTERFACE_STATE | An indication of the operation condition of an <i>Interface</i> component. |
| LOGIC_PROGRAM | An indication that an error occurred in the logic program or programmable logic controller (PLC) associated with a piece of equipment. |
| MOTION_PROGRAM | An indication that an error occurred in the motion program associated with a piece of equipment. |
| SYSTEM | An indication of a fault associated with a piece of equipment or component that cannot be classified as a specific type. |

1043 Appendices

1044 A Bibliography

- 1045 Engineering Industries Association. *EIA Standard - EIA-274-D*, Interchangeable Variable,
1046 Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically
1047 Controlled Machines. Washington, D.C. 1979.
- 1048 ISO TC 184/SC4/WG3 N1089. *ISO/DIS 10303-238*: Industrial automation systems and
1049 integration Product data representation and exchange Part 238: Application Protocols: Ap-
1050 plication interpreted model for computerized numerical controllers. Geneva, Switzerland,
1051 2004.
- 1052 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
1053 tems and integration – Physical device control – Data model for computerized numerical
1054 controllers – Part 10: General process data. Geneva, Switzerland, 2004.
- 1055 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
1056 tems and integration – Physical device control – Data model for computerized numerical
1057 controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.
- 1058 International Organization for Standardization. *ISO 6983/1* – Numerical Control of ma-
1059 chines – Program format and definition of address words – Part 1: Data format for posi-
1060 tioning, line and contouring control systems. Geneva, Switzerland, 1982.
- 1061 Electronic Industries Association. *ANSI/EIA-494-B-1992*, 32 Bit Binary CL (BCL) and
1062 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines.
1063 Washington, D.C. 1992.
- 1064 National Aerospace Standard. *Uniform Cutting Tests* - NAS Series: Metal Cutting Equip-
1065 ment Specifications. Washington, D.C. 1969.
- 1066 International Organization for Standardization. *ISO 10303-11*: 1994, Industrial automa-
1067 tion systems and integration Product data representation and exchange Part 11: Descrip-
1068 tion methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.
- 1069 International Organization for Standardization. *ISO 10303-21*: 1996, Industrial automa-
1070 tion systems and integration – Product data representation and exchange – Part 21: Imple-
1071 mentation methods: Clear text encoding of the exchange structure. Geneva, Switzerland,
1072 1996.
- 1073 H.L. Horton, F.D. Jones, and E. Oberg. *Machinery's Handbook*. Industrial Press, Inc.

- 1074 New York, 1984.
- 1075 International Organization for Standardization. *ISO 841-2001: Industrial automation sys-*
1076 *tems and integration - Numerical control of machines - Coordinate systems and motion*
1077 *nomenclature.* Geneva, Switzerland, 2001.
- 1078 ASME B5.57: *Methods for Performance Evaluation of Computer Numerically Controlled*
1079 *Lathes and Turning Centers,* 1998.
- 1080 ASME/ANSI B5.54: *Methods for Performance Evaluation of Computer Numerically Con-*
1081 *trolled Machining Centers.* 2005.
- 1082 OPC Foundation. *OPC Unified Architecture Specification, Part 1: Concepts Version 1.00.*
1083 July 28, 2006.
- 1084 IEEE STD 1451.0-2007, *Standard for a Smart Transducer Interface for Sensors and Ac-*
1085 *tuators – Common Functions, Communication Protocols, and Transducer Electronic Data*
1086 *Sheet (TEDS) Formats,* IEEE Instrumentation and Measurement Society, TC-9, The In-
1087 *stitute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH99684,*
1088 *October 5, 2007.*
- 1089 IEEE STD 1451.4-1994, Standard for a Smart Transducer Interface for Sensors and Ac-
1090 *tuators – Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet*
1091 *(TEDS) Formats,* IEEE Instrumentation and Measurement Society, TC-9, The Institute of
1092 *Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH95225, December*
1093 *15, 2004.*



MTConnect® Standard
Part 4.0 – Assets Information Model
Version 1.5.0

Prepared for: MTConnect Institute
Prepared on: December 2, 2019

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/>.

MTConnect Specification and Materials

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 | 7 |
| 2.3 MTConnect References | 8 |
| 3 MTConnect Assets | 9 |
| 3.1 Overview | 9 |
| 3.2 MTConnectAssets | 10 |
| 3.2.1 MTConnectAssets Header | 10 |
| 3.2.1.1 Header Attributes | 11 |
| 3.2.2 Assets | 13 |
| 3.2.3 Asset | 13 |
| 3.2.3.1 Common Asset Attributes | 14 |
| 3.2.3.2 Common Asset Elements | 16 |
| 4 MTConnect Assets Architecture | 17 |
| 4.1 Agent Asset Storage | 17 |
| 4.2 Asset Protocol | 18 |
| 4.2.1 Asset by assetId | 18 |
| 4.2.2 Asset for a Given Type | 19 |
| 4.2.3 Assets Including Removed Assets | 19 |
| 4.2.4 Assets for a Piece of Equipment | 20 |
| 5 Extensions to Part 2.0 - Devices Information Model | 21 |
| 5.1 Data Item Types added for EVENT Category | 21 |
| 5.1.1 ASSET_CHANGED Data Item Type | 21 |
| 5.1.2 ASSET_REMOVED Data Item Type | 22 |
| 6 Extensions to Part 3.0 - Streams Information Model | 23 |
| 6.1 AssetChanged Extension to Events | 23 |
| 6.1.1 AssetChanged event Attributes | 24 |
| 6.2 AssetRemoved Extension to Events | 24 |
| 6.2.1 AssetRemoved Attributes | 25 |
| Appendices | 26 |
| A Bibliography | 26 |

Table of Figures

| | |
|---|----|
| Figure 1: MTConnectAssets Schema | 10 |
| Figure 2: MTConnectAssets Header | 11 |
| Figure 3: Asset Schema | 14 |
| Figure 4: Description Schema | 16 |
| Figure 5: MTConnect Assets storage as First in First Out | 17 |
| Figure 6: MTConnect Assets storage as Key/Value pairs | 18 |
| Figure 7: AssetChanged Schema | 23 |
| Figure 8: AssetRemoved Schema | 24 |

List of Tables

| | |
|--|----|
| Table 1: MTConnectAssets Header | 12 |
| Table 2: MTConnect Assets Element | 13 |
| Table 3: MTConnect Asset Element | 13 |
| Table 4: Attributes for Asset | 14 |
| Table 5: Elements for Asset | 16 |
| Table 6: DataItem Type for EVENT category | 21 |
| Table 7: Attributes for AssetChanged | 24 |
| Table 8: Attributes for AssetRemoved | 25 |

1 Purpose of This Document

2 This document, *MTConnect Standard: Part 4.0 - Assets Information Model* of the MTCon-
3 nect Standard, details information that is common to all types of *MTConnect Assets*. Part
4 4.0 and its sub-parts of the MTConnect Standard provide semantic models for entities that
5 are used in the manufacturing process, but are not considered to be a piece of equipment.
6 These entities are defined as *MTConnect Assets*. These *Assets* may be removed from a
7 piece of equipment without detriment to the function of the equipment and can be associ-
8 ated with other pieces of equipment during their lifecycle. The data associated with these
9 *Assets* may be retrieved from multiple sources that are each responsible for providing their
10 knowledge of the *Asset*.

11 2 Terminology and Conventions

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

15 2.1 Glossary

16 CDATA

17 General meaning:

18 An abbreviation for Character Data.

19 CDATA is used to describe a value (text or data) published as part of an XML ele-
 20 ment.

21 For example, "This is some text" is the CDATA in the XML element:

22 <Message ...>This is some text</Message>

23 Appears in the documents in the following form: CDATA

24 NMTOKEN

25 The data type for XML identifiers.

26 Note: The identifier must start with a letter, an underscore "_" or a colon. The next
 27 character must be a letter, a number, or one of the following ".", "-", "_", ":". The
 28 identifier must not have any spaces or special characters.

29 Appears in the documents in the following form: NMTOKEN.

30 Agent

31 Refers to an MTConnect Agent.

32 Software that collects data published from one or more piece(s) of equipment, orga-
 33 nizes that data in a structured manner, and responds to requests for data from client
 34 software systems by providing a structured response in the form of a *Response Doc-
 35 ument* that is constructed using the *semantic data models* defined in the Standard.

36 Appears in the documents in the following form: *Agent*.

37 Asset

38 General meaning:

39 Typically referred to as an *MTConnect Asset*.

40 An *MTConnect Asset* is something that is used in the manufacturing process, but is
41 not permanently associated with a single piece of equipment, can be removed from
42 the piece of equipment without compromising its function, and can be associated
43 with other pieces of equipment during its lifecycle.

44 Used to identify a storage area in an *Agent*:

45 See description of *buffer*.

46 Used as an *Information Model*:

47 Used to describe an *Information Model* that contains the rules and terminology that
48 describe information that may be included in electronic documents representing *MT-*
49 *Connect Assets*.

50 The *Asset Information Models* defines the structure for the *Assets Response Docu-*
51 *ment*.

52 Individual *Information Models* describe the structure of the *Asset Documents* rep-
53 resent each type of *MTConnect Asset*. Appears in the documents in the following
54 form: *Asset Information Models* or (asset type) *Information Model*.

55 Used when referring to an *MTConnect Asset*:

56 Refers to the information related to an *MTConnect Asset* or a group of *MTConnect*
57 *Assets*.

58 Appears in the documents in the following form: *Asset* or *Assets*.

59 Used as an XML container or element:

60 ● When used as an XML container that consists of one or more types of *Asset*
61 XML elements.

62 Appears in the documents in the following form: *Assets*.

63 ● When used as an abstract XML element. It is replaced in the XML document
64 by types of *Asset* elements representing individual *Asset* entities.

65 Appears in the documents in the following form: *Asset*.

66 Used to describe information stored in an *Agent*:

67 Identifies an electronic document published by a data source and stored in the *assets*
68 *buffer* of an *Agent*.

69 Appears in the documents in the following form: *Asset Document*.

70 Used as an XML representation of an *MTConnect Response Document*:

71 Identifies an electronic document encoded in XML and published by an *Agent* in
72 response to a *Request* for information from a client software application relating to
73 *MTConnect Assets*.

74 Appears in the documents in the following form: *MTConnectAssets*.

75 Used as an *MTConnect Request*:

76 Represents a specific type of communications request between a client software ap-
77 plication and an *Agent* regarding *MTConnect Assets*.

78 Appears in the documents in the following form: *Asset Request*.

79 Used as part of an *HTTP Request*:

80 Used in the path portion of an *HTTP Request Line*, by a client software applica-
81 tion, to initiate an *Asset Request* to an *Agent* to publish an *MTConnectAssets*
82 document.

83 Appears in the documents in the following form: *asset*.

84 ***Asset Document***

85 An electronic document published by an *Agent* in response to a *Request* for infor-
86 mation from a client software application relating to Assets.

87 ***buffer***

88 General meaning:

89 A section of an *Agent* that provides storage for information published from pieces
90 of equipment.

91 Used relative to *Streaming Data*:

92 A section of an *Agent* that provides storage for information relating to individual
93 pieces of *Streaming Data*.

94 Appears in the documents in the following form: *buffer*.

95 Used relative to *MTConnect Assets*:

96 A section of an *Agent* that provides storage for *Asset Documents*.

97 Appears in the documents in the following form: *assets buffer*.

98 ***Data Entity***

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

102 Appears in the documents in the following form: *Data Entity*.

103 ***Document***

104 General meaning:

105 A piece of written, printed, or electronic matter that provides information.

106 Used to represent an *MTConnect Document*:

107 Refers to printed or electronic document(s) that represent a *Part(s)* of the MTCon-
108 nect Standard.

109 Appears in the documents in the following form: *MTConnect Document*.

110 Used to represent a specific representation of an *MTConnect Document*:

111 Refers to electronic document(s) associated with an *Agent* that are encoded using
112 XML; *Response Documents* or *Asset Documents*.

113 Appears in the documents in the following form: *MTConnect XML Document*.

114 Used to describe types of information stored in an *Agent*:

115 In an implementation, the electronic documents that are published from a data source
116 and stored by an *Agent*.

117 Appears in the documents in the following form: *Asset Document*.

118 Used to describe information published by an *Agent*:

119 A document published by an *Agent* based upon one of the *semantic data models*
120 defined in the MTConnect Standard in response to a request from a client.

121 Appears in the documents in the following form: *Response Document*.

122 ***Equipment Metadata***

123 See *Metadata*

124 ***HTTP Request***

125 In the MTConnect Standard, a communications command issued by a client soft-
126 ware application to an *Agent* requesting information defined in the *HTTP Request*
127 *Line*.

128 Appears in the documents in the following form: *HTTP Request*.

129 ***HTTP Request Line***

130 In the MTConnect Standard, the first line of an *HTTP Request* describing a specific
131 *Response Document* to be published by an *Agent*.

132 Appears in the documents in the following form: *HTTP Request Line*.

133 ***Information Model***

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

136 For example, an information model is used to define the structure for each *MTCon-*
137 *nnect Response Document*; the definition of each piece of information within those
138 documents and the relationship between pieces of information.

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

140 ***MTConnect Document***

141 See *Document*.

142 ***MTConnect Request***

143 A communication request for information issued from a client software application
144 to an *Agent*.

145 Appears in the documents in the following form: *MTConnect Request*.

146 ***MTConnect XML Document***

147 See *Document*.

148 ***Request***

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

151 Appears in the documents in the following form: *Request*.

152 ***Response Document***

153 See *Document*.

154 ***semantic data model***

155 A methodology for defining the structure and meaning for data in a specific logical
156 way.

157 It provides the rules for encoding electronic information such that it can be inter-
158 preted by a software system.

159 Appears in the documents in the following form: *semantic data model*.

160 ***Streaming Data***

161 The values published by a piece of equipment for the *Data Entities* defined by the
162 *Equipment Metadata*.

163 Appears in the documents in the following form: *Streaming Data*.

164 ***Valid Data Value***

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

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

168 **2.2 Acronyms**

169 **AMT**

170 The Association for Manufacturing Technology

171 **2.3 MTConnect References**

172 [MTConnect Part 1.0] *MTConnect Standard Part 1.0 - Overview and Fundamentals*. Version 1.5.0.

174 [MTConnect Part 3.0] *MTConnect Standard: Part 3.0 - Streams Information Model*. Version 1.5.0.

176 [MTConnect Part 4.0] *MTConnect Standard: Part 4.0 - Assets Information Model*. Version 1.5.0.

178 [MTConnect Part 4.1] *MTConnect Standard: Part 4.1 - Cutting Tools*. Version 1.5.0.

179 3 MTConnect Assets

180 3.1 Overview

181 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
182 MTConnectAssets document with a unique identifier and allows for some predefined
183 mechanisms to find, create, request, updated, and delete these electronic documents in a
184 way that provides for consistency across multiple pieces of equipment.
185

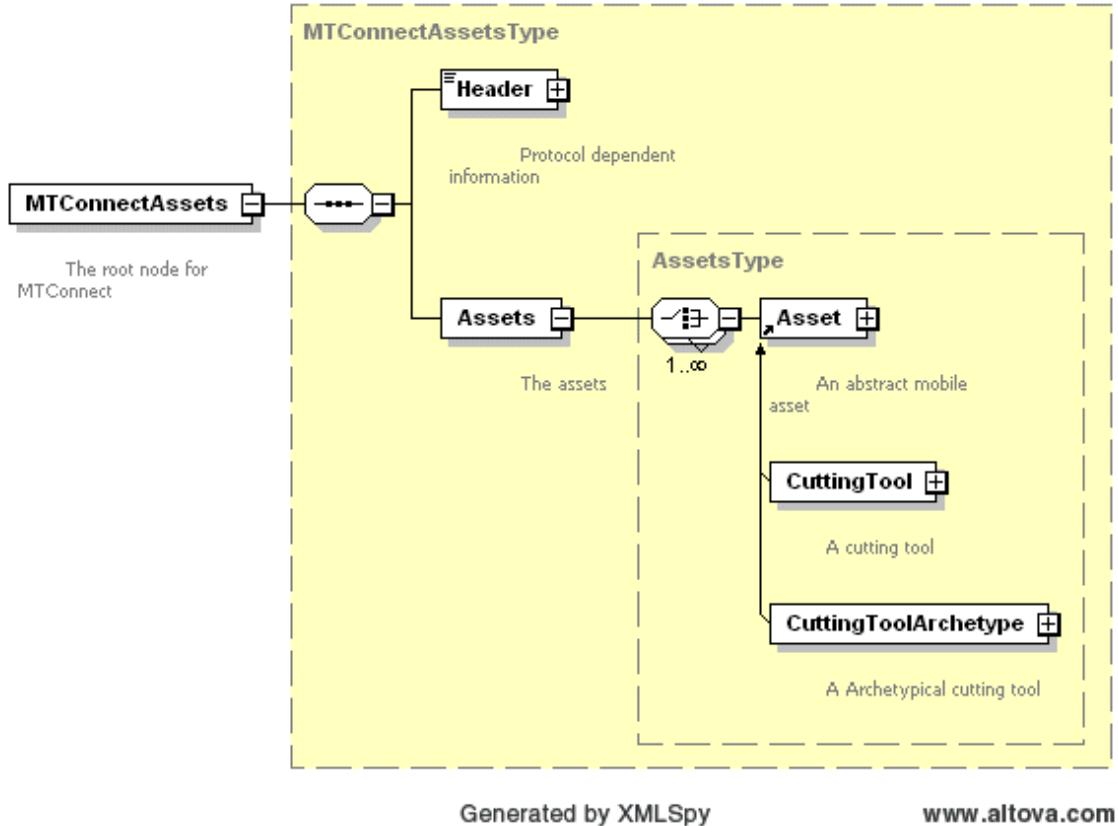
186 The protocol provides a limited mechanism of accessing *MTConnect Assets* using the following properties: `assetId`, *Asset* type (element name of *Asset* root), and the piece of equipment associated with the *Asset*. These access strategies will provide the following services and answer the following 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`?
187

188 Although these mechanisms are provided, an *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 need-ed 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 an-other system to provide the source data upon initialization. An *Agent* is always providing the best-known equipment centric view of the data given the limitations of that piece of equipment.
189

190 Note: Currently only cutting tools have been addressed by the MTConnect Standard
191 and other MTConnect Assets will be defined in later versions of the Standard.
192

203

3.2 MTConnectAssets

**Figure 1:** MTConnectAssets Schema

- 204 At the top level of the `MTConnectAssets` document is a standard header, as stated in
 205 *MTConnect Standard Part 1.0 - Overview and Fundamentals*, and one or more `MTConnect`
 206 `Assets`. Each `Asset` is required to have an `assetId` that serves as a unique identifier of
 207 that `Asset`. `assetId` allows an application to request the `Asset` data from an `Agent`.
- 208 In the remaining *Part 4.x* sub-part documents of *MTConnect Assets*, various types of `Assets`
 209 will be introduced such as cutting tools and other `Asset` types. Currently only cutting
 210 tools have been defined in *MTConnect Standard: Part 4.1 - Cutting Tools*.

211

3.2.1 MTConnectAssets Header

- 212 The `MTConnectAssets` header is where the protocol sequence information **MUST** be
 213 provided. The XML schema in *Figure 2* represents the structure of the `MTConnectAssets`-
 214 `header` showing the attributes defined for `MTConnectAssets`.

215 Refer to *MTConnect Standard Part 1.0 - Overview and Fundamentals* for more information
 216 on headers.

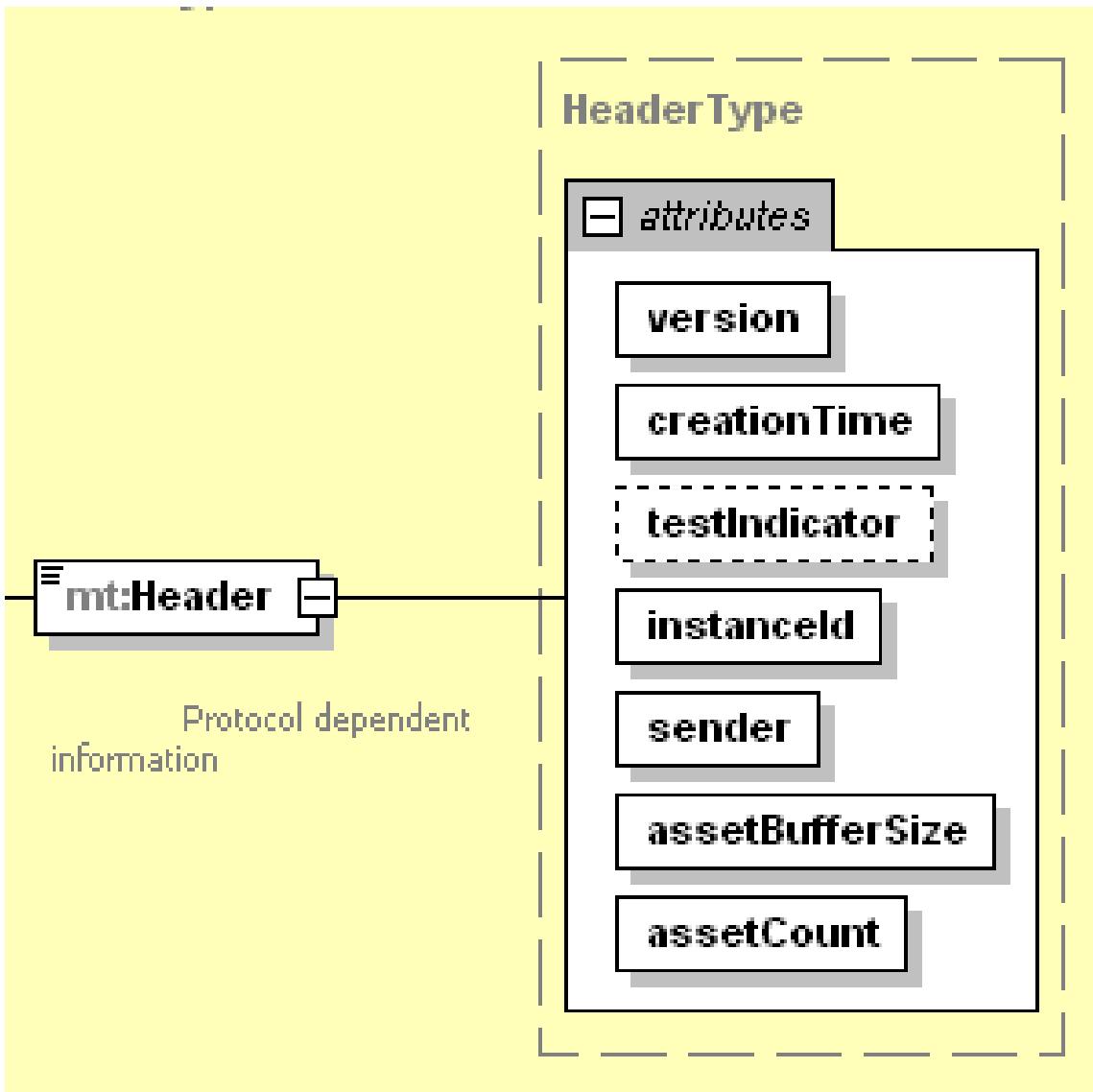


Figure 2: MTConnectAssets Header

217 **3.2.1.1 Header Attributes**

218 *Table 1* defines the attributes used to provide information for an MTConnectAssets
 219 header.

Table 1: MTConnectAssets Header

| Attribute | Description | Occurrence |
|-----------------|---|------------|
| version | The protocol version number. This is the <i>major</i> and <i>minor</i> 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 |
| 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 <i>Agent</i> . This is used to differentiate between separate instances of the <i>Agent</i> . 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 <i>Agent</i> identification information. sender is a required attribute. | 1 |
| assetBufferSize | The maximum number of <i>MTConnect Assets</i> that will be retained by the <i>Agent</i> . 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 <i>MTConnect Assets</i> in anAgent. 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: MTConnectAssets Header Example

```

220 1 <Header creationTime="2010-03-13T07:59:11+00:00"
221 2     sender="localhost" instanceId="1268463594"
222 3     assetBufferSize="1024" version="1.1"
223 4     assetCount="12" />

```

224 3.2.2 Assets

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

Table 2: MTConnect Assets Element

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

227 3.2.3 Asset

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

Table 3: MTConnect Asset Element

| 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..* |

233 There are various types of entities or Asset types. Each type of Asset is described in sub-
 234 parts of *MTConnect Standard: Part 4.0 - Assets Information Model*. These sub-parts are

235 designated by a *Part 4.x* document number. Currently only the *MTConnect Asset* type of
 236 cutting tools has been defined in *MTConnect Standard: Part 4.1 - Cutting Tools*.

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

239 **3.2.3.1 Common Asset Attributes**

240 The XML schema in *Figure 3* represents the structure of *Asset* showing the attributes
 241 defined for *Asset*.

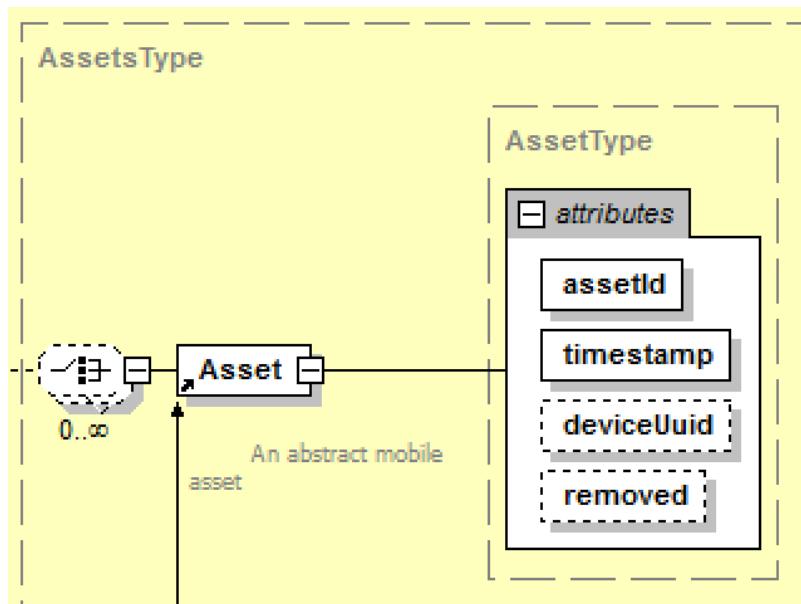


Figure 3: Asset Schema

242 *Table 4* defines the attributes that are used to provide information for the *Asset* element.

Table 4: Attributes for Asset

| Attribute | Description | Occurrence |
|-----------|---|------------|
| assetId | The unique identifier for the <i>MTConnect Asset</i> . The identifier MUST be unique with respect to all other <i>Assets</i> in an MTConnect installation. The identifier SHOULD be globally unique with respect to all other <i>Assets</i> . assetId is a required attribute. | 1 |

| Continuation of Table 4 | | |
|-------------------------|--|------------|
| Attribute | Description | Occurrence |
| timestamp | The time this <i>MTConnect Asset</i> was last modified. Always given in UTC. The <code>timestamp</code> MUST be provided in UTC (Universal Time Coordinate, also known as GMT). This is the time the <i>Asset</i> data was last modified. <code>timestamp</code> is a required attribute. | 1 |
| deviceUuid | The piece of equipments UUID that supplied this data. This is an optional element references to the <code>UUID</code> attribute given in the <code>Device</code> element. This can be any series of numbers and letters as defined by the XML type <code>NMTOKEN</code> . | 0..1 |
| removed | This is an optional attribute that is an indicator that the <i>MTConnect Asset</i> has been removed from the piece of equipment. If the <i>Asset</i> is marked as removed, it will not be visible to the client application unless the <code>=true</code> parameter is provided in the URL. If this attribute is not present it MUST be assumed to be false. The value is an <code>xsi:boolean</code> type and MUST be <code>true</code> or <code>false</code> . | 0..1 |

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

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

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

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

262 **3.2.3.2 Common Asset Elements**

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

264 The XML schema in *Figure 4* represents the structure of *Description*.

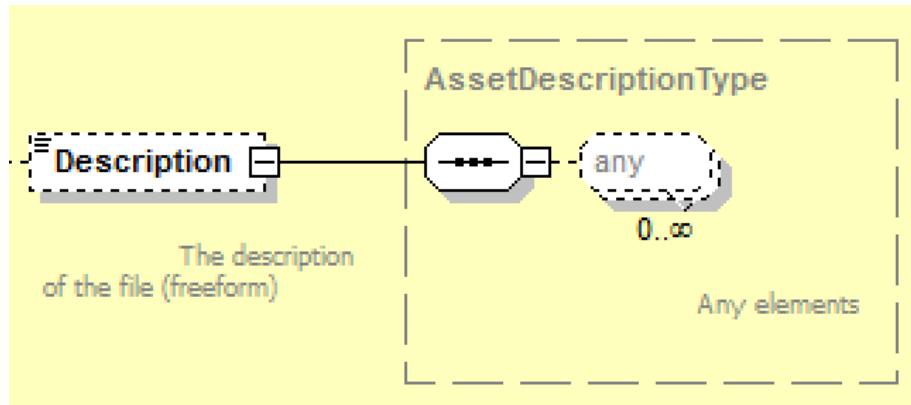


Figure 4: Description Schema

265 *Table 5* defines the elements that are used to provide information for *Asset*.

Table 5: Elements for Asset

| Elements | 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 |

266 4 MTConnect Assets Architecture

267 4.1 Agent Asset Storage

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

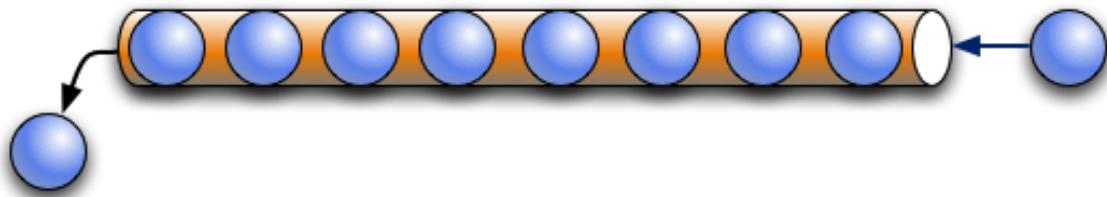


Figure 5: MTConnect Assets storage as First in First Out

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

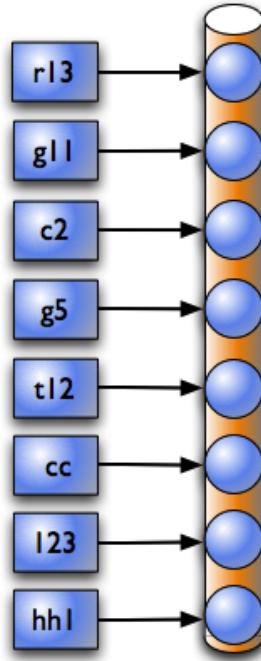


Figure 6: MTConnect Assets storage as Key/Value pairs

280 As in *Figure 6* , each of the *Assets* is referred to by their key. The key is independent of
 281 the order in the *asset buffer* storage.

282 4.2 Asset Protocol

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

286 The URL format is similar to the `probe` and `sample` structure. Reference each as-
 287 set `assetId` directly to request an *MTConnect Asset* by `assetId`.

288 4.2.1 Asset by assetId

Example 2: Asset by assetId Example

```
289 1 url: http://example.com/asset/e39d23ba-ef2d-
290 2       11e6-b12c15028cf91a82ef
```

291 *Example 2* returns the MTConnectAssets document for Asset e39d23ba-ef2d-
 292 11e6-b12c-28cfe91a82ef

293 Request multiple Assets by each assetId:

Example 3: Assets by assetId Example

294 1 url: http://example.com/asset/e39d23ba-ef2d-11e6-b12c155;
 295 2 8cfe91a82ef;e46d5256-ef2d-11e6-96aa-28cfe91a82ef

296 *Example 3* returns the MTConnectAssets document for Assets e39d23ba-ef2d-
 297 11e6-b12c-28cfe91a82ef and e46d5256-ef2d-11e6-96aa-28cfe91a82ef.

298 Request for all the Assets in the Agent:

Example 4: Get all Assets Example

299 1 url: http://example.com/assets

300 *Example 4* returns all available MTConnect Assets in the Agent. The Agent **MAY** return
 301 a limited set if there are too many Asset records. The Assets **MUST** be added to the
 302 beginning with the most recently modified Asset.

303 4.2.2 Asset for a Given Type

Example 5: Asset for a Given Type Example

304 1 url: http://example.com/assets?type="CuttingTool"

305 *Example 5* returns all available CuttingTool Assets from the Agent of the type Cut-
 306 tingTool. The Agent **MAY** return a limited set if there are too many Asset records. The
 307 Assets **MUST** be added to the beginning with the most recently modified assets.

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

Example 6: Asset for a Given Type with Maximum count Example

309 1 url: http://example.com/assets?type="CuttingTool"

310 *Example 6* returns all available CuttingTool Assets from the Agent. The Agent **MUST**
 311 return up to 1000 Assets beginning with the most recently modified Assets if they exist.

312 4.2.3 Assets Including Removed Assets

Example 7: Assets Including Removed Assets Example

313 1 url: http://example.com/assets?type=CuttingTool&removed=true

314 *Example 7* returns all available *CuttingTool Assets* from the *Agent*. With the removed
 315 flag, *Assets* that have been removed but are included in the result set.

316 4.2.4 Assets for a Piece of Equipment

317 If no *assetId* is provided with a general *Assets* request, it would be as shown in *Exam-*
 318 *ple 8*:

Example 8: Assets For a Piece of Equipment Example

319 1 url: <http://example.com/Mill123/assets>

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

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

Example 9: Assets For a Piece of Equipment For a Given Type Example

325 1 url: <http://example.com/Mill123/asset/>
 326 2 ?type=CuttingTool&count=100

327 The request in *Example 9* will get the newest 100 *Cutting Tool Instance Assets* from the
 328 *Agent* for *Mill123*. Similarly:

Example 10: Assets For a Piece of Equipment For a Given Type Example 2

329 1 url: <http://example.com/Mill123/asset/>
 330 2 ?type=CuttingToolArchetype

331 *Example 10* will provide all *Cutting Tool Archetype Assets* with the *deviceUuid* of
 332 *Mill123*.

333 5 Extensions to Part 2.0 - Devices Information Model

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

338 5.1 Data Item Types added for EVENT Category

Table 6: DataItem Type for EVENT category

| DataItem 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. |

339 5.1.1 ASSET_CHANGED Data Item Type

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

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

349 5.1.2 ASSET_REMOVED Data Item Type

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

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

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

364 6 Extensions to Part 3.0 - Streams Information Model

365 The associated modifications **MUST** be added to *MTConnect Standard: Part 3.0 - Streams*
 366 *Information Model* to add the following event to the Events in the streams.

367 6.1 AssetChanged Extension to Events

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

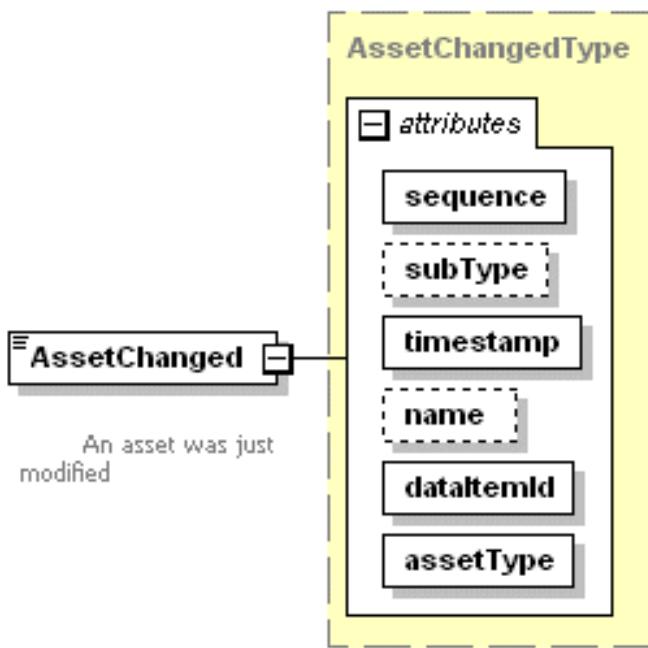


Figure 7: AssetChanged Schema

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

376 **6.1.1 AssetChanged event Attributes**

Table 7: Attributes for AssetChanged

| Attribute | Description | Occurrence |
|-----------|---|------------|
| assetType | <p>The type of asset changed. assetType is a required attribute.</p> <p><i>Valid Data Values:</i></p> <p>Cutting Tool</p> | 1 |

377 **6.2 AssetRemoved Extension to Events**

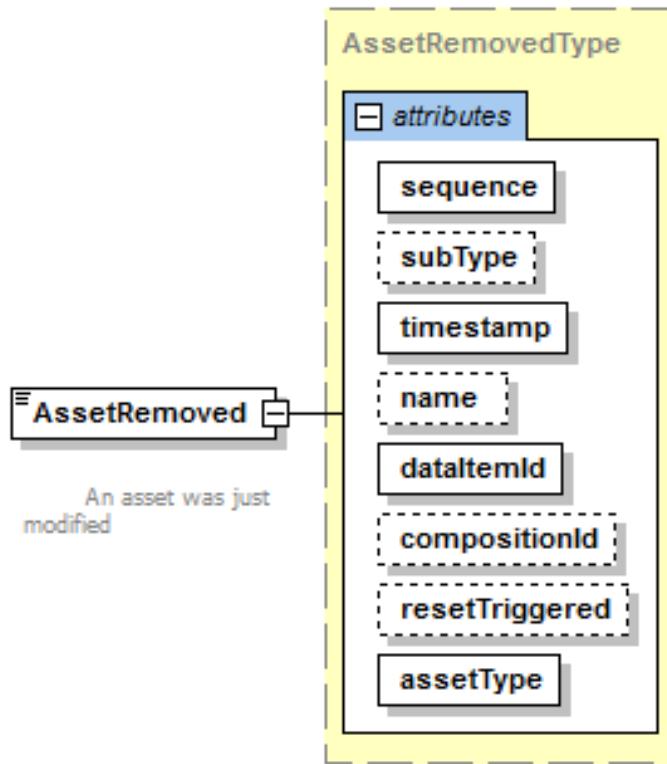


Figure 8: AssetRemoved Schema

378 AssetRemoved: An *MTConnect Asset* has been removed. The CDATA for the As-
379 setRemoved element **MUST** be the assetId of the *Asset* that has been removed.

380 **6.2.1 AssetRemoved Attributes**

Table 8: Attributes for AssetRemoved

| Attribute | Description | Occurrence |
|-----------|--|------------|
| assetType | <p>The type of asset that was removed. <code>assetType</code> is a required attribute.</p> <p><i>Valid Data Values:</i></p> <p>Cutting Tool</p> | 1 |

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

384 Appendices

385 A Bibliography

- 386 Engineering Industries Association. *EIA Standard - EIA-274-D*, Interchangeable Variable,
387 Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically
388 Controlled Machines. Washington, D.C. 1979.
- 389 ISO TC 184/SC4/WG3 N1089. *ISO/DIS 10303-238*: Industrial automation systems and
390 integration Product data representation and exchange Part 238: Application Protocols: Ap-
391 plication interpreted model for computerized numerical controllers. Geneva, Switzerland,
392 2004.
- 393 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
394 tems and integration – Physical device control – Data model for computerized numerical
395 controllers – Part 10: General process data. Geneva, Switzerland, 2004.
- 396 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
397 tems and integration – Physical device control – Data model for computerized numerical
398 controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.
- 399 International Organization for Standardization. *ISO 6983/1* – Numerical Control of ma-
400 chines – Program format and definition of address words – Part 1: Data format for posi-
401 tioning, line and contouring control systems. Geneva, Switzerland, 1982.
- 402 Electronic Industries Association. *ANSI/EIA-494-B-1992*, 32 Bit Binary CL (BCL) and
403 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines.
404 Washington, D.C. 1992.
- 405 National Aerospace Standard. *Uniform Cutting Tests* - NAS Series: Metal Cutting Equip-
406 ment Specifications. Washington, D.C. 1969.
- 407 International Organization for Standardization. *ISO 10303-11*: 1994, Industrial automa-
408 tion systems and integration Product data representation and exchange Part 11: Descrip-
409 tion methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.
- 410 International Organization for Standardization. *ISO 10303-21*: 1996, Industrial automa-
411 tion systems and integration – Product data representation and exchange – Part 21: Imple-
412 mentation methods: Clear text encoding of the exchange structure. Geneva, Switzerland,
413 1996.
- 414 H.L. Horton, F.D. Jones, and E. Oberg. *Machinery's Handbook*. Industrial Press, Inc.

415 New York, 1984.

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

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

421 *ASME/ANSI B5.54: Methods for Performance Evaluation of Computer Numerically Con-*
422 *trolled Machining Centers.* 2005.

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

425 International Organization for Standardization. *ISO 13399: Cutting tool data representa-*
426 *tion and exchange.* Geneva, Switzerland, 2000.



MTConnect® Standard
Part 4.1 – Cutting Tools
Version 1.5.0

Prepared for: MTConnect Institute
Prepared on: December 2, 2019

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/>.

MTConnect Specification and Materials

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 Cutting Tool and Cutting Tool Archetype | 9 |
| 3.1 XML Schema Structure for CuttingTool and CuttingToolArchetype | 9 |
| 3.2 Common Attributes for CuttingTool and CuttingToolArchetype | 11 |
| 3.3 Common Elements for CuttingTool and CuttingToolArchetype | 13 |
| 3.3.1 Description Element for CuttingTool and CuttingToolArchetype . . | 13 |
| 4 CuttingToolArchetype Information Model | 14 |
| 4.1 Attributes for CuttingToolArchetype | 18 |
| 4.2 Elements for CuttingToolArchetype | 18 |
| 4.2.1 CuttingToolDefinition Element for CuttingToolArchetype | 19 |
| 4.2.1.1 Attributes for CuttingToolDefinition | 19 |
| 4.2.1.1.1 format Attribute for CuttingToolDefinition | 19 |
| 4.2.1.2 Elements for CuttingToolDefinition | 20 |
| 4.2.1.3 ISO13399 Standard | 20 |
| 4.2.2 CuttingToolLifeCycle Element for CuttingToolArchetype | 20 |
| 5 CuttingTool Information model | 21 |
| 5.1 Attributes for CuttingTool | 21 |
| 5.2 Elements for CuttingTool | 21 |
| 5.2.1 CuttingToolLifeCycle Elements for CuttingTool Only | 22 |
| 5.2.1.1 CutterStatus Element for CuttingToolLifeCycle | 22 |
| 5.2.1.1.1 Status Element for CutterStatus | 23 |
| 5.2.1.2 ToolLife Element for CuttingToolLifeCycle | 25 |
| 5.2.1.2.1 Attributes for ToolLife | 26 |
| 5.2.1.2.2 type Attribute for ToolLife | 26 |
| 5.2.1.2.3 countDirection Attribute for ToolLife | 27 |
| 5.2.1.3 Location Element for CuttingToolLifeCycle | 27 |
| 5.2.1.3.1 Attributes for Location | 28 |
| 5.2.1.3.2 type Attribute for Location | 28 |
| 5.2.1.3.3 positiveOverlap Attribute for Location | 29 |
| 5.2.1.3.4 negativeOverlap Attribute for Location | 29 |
| 5.2.1.4 ReconditionCount Element for CuttingToolLifeCycle . . | 29 |
| 5.2.1.4.1 Attributes for ReconditionCount | 29 |
| 5.2.2 CuttingToolArchetypeReference Element for Cutting Tool | 30 |

| | |
|---|-----------|
| 5.2.2.1 source Attribute for CuttingToolArcheTypeReference | 30 |
| 6 Common Entity CuttingToolLifeCycle | 31 |
| 6.1 CuttingToolLifeCycle | 31 |
| 6.1.1 XML Schema Structure for CuttingToolLifeCycle | 31 |
| 6.2 Elements for CuttingToolLifeCycle | 33 |
| 6.2.1 ProgramToolGroup Element for CuttingToolLifeCycle | 34 |
| 6.2.2 ProgramToolNumber Element for CuttingToolLifeCycle | 34 |
| 6.2.3 ProcessSpindleSpeed Element for CuttingToolLifeCycle | 35 |
| 6.2.3.1 Attributes for ProcessSpindleSpeed | 35 |
| 6.2.4 ProcessFeedRate Element for CuttingToolLifeCycle | 36 |
| 6.2.4.1 Attributes for ProcessFeedRate | 36 |
| 6.2.5 ConnectionCodeMachineSide Element for CuttingToolLifeCycle . | 37 |
| 6.2.6 xs:any Element for CuttingToolLifeCycle | 37 |
| 6.2.7 Measurements Element for CuttingToolLifeCycle | 37 |
| 6.2.8 Measurement | 38 |
| 6.2.8.1 Attributes for Measurement | 39 |
| 6.2.8.2 Measurement Subtypes for CuttingToolLifeCycle | 40 |
| 6.2.9 CuttingItems Element for CuttingToolLifeCycle | 44 |
| 6.2.9.1 Attributes for CuttingItems | 45 |
| 6.2.10 CuttingItem | 45 |
| 6.2.10.1 Attributes for CuttingItem | 47 |
| 6.2.10.1.1 indices Attribute for CuttingItem | 47 |
| 6.2.10.1.2 itemId Attribute for CuttingItem | 47 |
| 6.2.10.1.3 manufacturers Attribute for CuttingItem | 47 |
| 6.2.10.1.4 grade Attribute for CuttingItem | 48 |
| 6.2.10.2 Elements for CuttingItem | 48 |
| 6.2.10.2.1 Description Element for CuttingItem | 48 |
| 6.2.10.2.2 Locus Element for CuttingItem | 48 |
| 6.2.10.2.3 ItemLife Element for CuttingItem | 49 |
| 6.2.10.2.4 Attributes for ItemLife | 50 |
| 6.2.10.2.5 type Attribute for ItemLife | 50 |
| 6.2.10.2.6 countDirection Attribute for ItemLife | 51 |
| 6.2.10.3 Measurement Subtypes for CuttingItem | 51 |
| Appendices | 58 |
| A Bibliography | 58 |
| B Additional Illustrations | 60 |
| C Cutting Tool Example | 64 |
| C.1 Shell Mill | 64 |
| C.2 Step Drill | 67 |
| C.3 Shell Mill with Individual Loci | 69 |

| | | |
|-----|--|----|
| C.4 | Drill with Individual Loci | 71 |
| C.5 | Shell Mill with Different Inserts on First Row | 73 |

Table of Figures

| | |
|---|----|
| Figure 1: Cutting Tool Schema | 10 |
| Figure 2: Cutting Tool Parts | 14 |
| Figure 3: Cutting Tool Composition | 15 |
| Figure 4: Cutting Tool, Tool Item, and Cutting Item | 16 |
| Figure 5: Cutting Tool, Tool Item, and Cutting Item 2 | 16 |
| Figure 6: Cutting Tool Measurements | 17 |
| Figure 7: Cutting Tool Asset Structure | 17 |
| Figure 8: CuttingToolDefinition Schema | 19 |
| Figure 9: CutterStatus Schema | 22 |
| Figure 10:ToolLife Schema | 25 |
| Figure 11:Location Schema | 27 |
| Figure 12:ReconditionCount Schema | 29 |
| Figure 13:CuttingToolArcheTypeReference Schema | 30 |
| Figure 14:CuttingToolLifeCycle Schema | 32 |
| Figure 15:ProcessSpindleSpeed Schema | 35 |
| Figure 16:ProcessFeedRate Schema | 36 |
| Figure 17:Measurement Schema | 38 |
| Figure 18:Cutting Tool Measurement Diagram 1 | 40 |
| Figure 19:Cutting Tool Measurement Diagram 2 | 41 |
| Figure 20:CuttingItems Schema | 44 |
| Figure 21:CuttingItem Schema | 46 |
| Figure 22:ItemLife Schema | 49 |
| Figure 23:Cutting Tool | 52 |
| Figure 24:Cutting Item | 52 |
| Figure 25:Cutting Item Measurement Diagram 3 | 53 |
| Figure 26:Cutting Item Drive Angle | 53 |
| Figure 27:Cutting Tool Measurement Diagram 1 (Cutting Tool, Cutting Item, and Assembly Item – ISO 13399) | 60 |
| Figure 28:Cutting Tool Measurement Diagram 2 (Cutting Tool, Cutting Item, and Assembly Item – ISO 13399) | 61 |
| Figure 29:Cutting Tool Measurement Diagram 3 (Cutting Item – ISO 13399) | 61 |
| Figure 30:Cutting Tool Measurement Diagram 4 (Cutting Item – ISO 13399) | 62 |
| Figure 31:Cutting Tool Measurement Diagram 5 (Cutting Item – ISO 13399) | 62 |
| Figure 32:Cutting Tool Measurement Diagram 6 (Cutting Item – ISO 13399) | 63 |
| Figure 33:Shell Mill Side View | 64 |
| Figure 34:Indexable Insert Measurements | 64 |
| Figure 35:Step Mill Side View | 67 |
| Figure 36:Shell Mill with Explicate Loci | 69 |
| Figure 37:Step Drill with Explicate Loci | 71 |
| Figure 38:Shell Mill with Different Inserts on First Row | 73 |

List of Tables

| | |
|--|----|
| Table 1: Attributes for CuttingTool and CuttingToolArchetype | 11 |
| Table 2: Common Elements for CuttingTool and CuttingToolArchetype | 13 |
| Table 3: Elements for CuttingToolArchetype | 18 |
| Table 4: Attributes for CuttingToolDefinition | 19 |
| Table 5: Values for format attribute of CuttingToolDefinition | 20 |
| Table 6: Elements for CuttingTool | 21 |
| Table 7: Elements for CutterStatus | 23 |
| Table 8: Values for Status Element of CutterStatus | 23 |
| Table 9: Attributes for ToolLife | 26 |
| Table 10: Values for type of ToolLife | 27 |
| Table 11: Values for countDirection | 27 |
| Table 12: Attributes for Location | 28 |
| Table 13: Values for type of Location | 28 |
| Table 14: Attributes for ReconditionCount | 29 |
| Table 15: Attributes for CuttingToolArchetypeReference | 30 |
| Table 16: Elements for CuttingToolLifeCycle | 33 |
| Table 17: Attributes for ProcessSpindleSpeed | 35 |
| Table 18: Attributes for ProcessFeedRate | 36 |
| Table 19: Attributes for Measurement | 39 |
| Table 20: Measurement Subtypes for CuttingTool | 41 |
| Table 21: Attributes for CuttingItems | 45 |
| Table 22: Attributes for CuttingItem | 47 |
| Table 23: Elements for CuttingItem | 48 |
| Table 24: Attributes for ItemLife | 50 |
| Table 25: Values for type of ItemLife | 51 |
| Table 26: Values for countDirection | 51 |
| Table 27: Measurement Subtypes for CuttingItem | 53 |

1 Purpose of This Document

- 2 This document, *MTConnect Standard: Part 4.1 - Cutting Tools* of the MTConnect Standard,
3 establishes the rules and terminology to be used by designers to describe the function
4 and operation of cutting tools used within manufacturing and to define the data that is pro-
5 vided by an *Agent* from a piece of equipment. This part of the Standard also defines the
6 structure for the XML document that is returned from an *Agent* in response to a probe
7 request.
- 8 The data associated with these cutting tools will be retrieved from multiple sources that
9 are responsible for providing their knowledge of an *MTConnect Asset*.

10 2 Terminology and Conventions

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

14 2.1 Glossary

15 CDATA

16 General meaning:

17 An abbreviation for Character Data.

18 CDATA is used to describe a value (text or data) published as part of an XML ele-
 19 ment.

20 For example, "This is some text" is the CDATA in the XML element:

21 <Message ...>This is some text</Message>

22 Appears in the documents in the following form: CDATA

23 NMOKEN

24 The data type for XML identifiers.

25 Note: The identifier must start with a letter, an underscore "_" or a colon. The next
 26 character must be a letter, a number, or one of the following ".", "-", "_", ":". The
 27 identifier must not have any spaces or special characters.

28 Appears in the documents in the following form: NMOKEN.

29 XML

30 Stands for eXtensible Markup Language.

31 XML defines a set of rules for encoding documents that both a human-readable and
 32 machine-readable.

33 XML is the language used for all code examples in the MTConnect Standard.

34 Refer to <http://www.w3.org/XML> for more information about XML.

35 Agent

36 Refers to an MTConnect Agent.

37 Software that collects data published from one or more piece(s) of equipment, orga-
 38 nizes that data in a structured manner, and responds to requests for data from client

39 software systems by providing a structured response in the form of a *Response Document*
40 that is constructed using the *semantic data models* defined in the Standard.
41 Appears in the documents in the following form: *Agent*.

42 **Asset**

43 General meaning:

44 Typically referred to as an *MTConnect Asset*.

45 An *MTConnect Asset* is something that is used in the manufacturing process, but is
46 not permanently associated with a single piece of equipment, can be removed from
47 the piece of equipment without compromising its function, and can be associated
48 with other pieces of equipment during its lifecycle.

49 Used to identify a storage area in an Agent:

50 See description of *buffer*.

51 Used as an Information Model:

52 Used to describe an *Information Model* that contains the rules and terminology that
53 describe information that may be included in electronic documents representing *MT-*
54 *Connect Assets*.

55 The *Asset Information Models* defines the structure for the *Assets Response Document*.
56

57 Individual *Information Models* describe the structure of the *Asset Documents* rep-
58 resent each type of *MTConnect Asset*. Appears in the documents in the following
59 form: *Asset Information Models* or (asset type) *Information Model*.

60 Used when referring to an MTConnect Asset:

61 Refers to the information related to an *MTConnect Asset* or a group of *MTConnect*
62 *Assets*.

63 Appears in the documents in the following form: *Asset* or *Assets*.

64 Used as an XML container or element:

- 65 • When used as an XML container that consists of one or more types of *Asset*
66 XML elements.

67 Appears in the documents in the following form: *Assets*.

- 68 • When used as an abstract XML element. It is replaced in the XML document
69 by types of *Asset* elements representing individual *Asset* entities.

70 Appears in the documents in the following form: *Asset*.

71 Used to describe information stored in an Agent:

72 Identifies an electronic document published by a data source and stored in the *assets*
73 *buffer* of an *Agent*.

74 Appears in the documents in the following form: *Asset Document*.

75 Used as an XML representation of an *MTConnect Response Document*:

76 Identifies an electronic document encoded in XML and published by an *Agent* in
77 response to a *Request* for information from a client software application relating to
78 *MTConnect Assets*.

79 Appears in the documents in the following form: `MTConnectAssets`.

80 Used as an *MTConnect Request*:

81 Represents a specific type of communications request between a client software ap-
82 plication and an *Agent* regarding *MTConnect Assets*.

83 Appears in the documents in the following form: *Asset Request*.

84 Used as part of an *HTTP Request*:

85 Used in the path portion of an *HTTP Request Line*, by a client software applica-
86 tion, to initiate an *Asset Request* to an *Agent* to publish an `MTConnectAssets`
87 document.

88 Appears in the documents in the following form: `asset`.

89 ***Asset Document***

90 An electronic document published by an *Agent* in response to a *Request* for infor-
91 mation from a client software application relating to Assets.

92 ***Attribute***

93 A term that is used to provide additional information or properties for an element.

94 Appears in the documents in the following form: `attribute`.

95 ***buffer***

96 General meaning:

97 A section of an *Agent* that provides storage for information published from pieces
98 of equipment.

99 Used relative to *Streaming Data*:

100 A section of an *Agent* that provides storage for information relating to individual
101 pieces of *Streaming Data*.

102 Appears in the documents in the following form: `buffer`.

103 Used relative to *MTConnect Assets*:

104 A section of an *Agent* that provides storage for *Asset Documents*.

105 Appears in the documents in the following form: `assets buffer`.

106 ***Data Entity***

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

110 Appears in the documents in the following form: *Data Entity*.

111 ***Document***

112 General meaning:

113 A piece of written, printed, or electronic matter that provides information.

114 Used to represent an *MTConnect Document*:

115 Refers to printed or electronic document(s) that represent a *Part(s)* of the MTCon-
116 nect Standard.

117 Appears in the documents in the following form: *MTConnect Document*.

118 Used to represent a specific representation of an *MTConnect Document*:

119 Refers to electronic document(s) associated with an *Agent* that are encoded using
120 XML; *Response Documents* or *Asset Documents*.

121 Appears in the documents in the following form: *MTConnect XML Document*.

122 Used to describe types of information stored in an *Agent*:

123 In an implementation, the electronic documents that are published from a data source
124 and stored by an *Agent*.

125 Appears in the documents in the following form: *Asset Document*.

126 Used to describe information published by an *Agent*:

127 A document published by an *Agent* based upon one of the *semantic data models*
128 defined in the MTConnect Standard in response to a request from a client.

129 Appears in the documents in the following form: *Response Document*.

130 ***Equipment Metadata***

131 See *Metadata*

132 ***HTTP Request***

133 In the MTConnect Standard, a communications command issued by a client soft-
134 ware application to an *Agent* requesting information defined in the *HTTP Request*
135 *Line*.

136 Appears in the documents in the following form: *HTTP Request*.

137 ***HTTP Request Line***

138 In the MTConnect Standard, the first line of an *HTTP Request* describing a specific
139 *Response Document* to be published by an *Agent*.

140 Appears in the documents in the following form: *HTTP Request Line*.

141 ***Information Model***

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

144 For example, an information model is used to define the structure for each *MTCon-*
145 *nnect Response Document*; the definition of each piece of information within those
146 documents and the relationship between pieces of information.

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

148 ***MTConnect Document***

149 See *Document*.

150 ***MTConnect Request***

151 A communication request for information issued from a client software application
152 to an *Agent*.

153 Appears in the documents in the following form: *MTConnect Request*.

154 ***MTConnect XML Document***

155 See *Document*.

156 ***Request***

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

159 Appears in the documents in the following form: *Request*.

160 ***Response Document***

161 See *Document*.

162 ***semantic data model***

163 A methodology for defining the structure and meaning for data in a specific logical
164 way.

165 It provides the rules for encoding electronic information such that it can be inter-
166 preted by a software system.

167 Appears in the documents in the following form: *semantic data model*.

168 ***Streaming Data***

169 The values published by a piece of equipment for the *Data Entities* defined by the
170 *Equipment Metadata*.

171 Appears in the documents in the following form: *Streaming Data*.

172 ***Valid Data Value***

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

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

176 ***XML Schema***

177 In the MTConnect Standard, an instantiation of a schema defining a specific docu-
178 ment encoded in XML.

179 **2.2 Acronyms**

180 ***AMT***

181 The Association for Manufacturing Technology

182 **2.3 MTConnect References**

183 [MTConnect Part 1.0] *MTConnect Standard Part 1.0 - Overview and Fundamentals*. Ver-
184 sion 1.5.0.

185 [MTConnect Part 2.0] *MTConnect Standard: Part 2.0 - Devices Information Model*. Ver-
186 sion 1.5.0.

187 [MTConnect Part 3.0] *MTConnect Standard: Part 3.0 - Streams Information Model*. Ver-
188 sion 1.5.0.

189 [MTConnect Part 4.1] *MTConnect Standard: Part 4.1 - Cutting Tools*. Version 1.5.0.

190 3 Cutting Tool and Cutting Tool Archetype

191 There are two *Information Models* used to represent a cutting tool, *CuttingToolArchetype*
192 and *CuttingTool*. The *CuttingToolArchetype* represent the static cutting tool
193 geometries and nominal values as one would expect from a tool catalog and the *Cut-
194 tingTool* represents the use or application of the tool on the shop floor with actual
195 measured values and process data. In Version 1.3.0 of the MTConnect Standard it was de-
196 cided to separate out these two concerns since not all pieces of equipment will have access
197 to both sets of information. In this way, a generic definition of the cutting tool can coexist
198 with a specific assembly *Information Model* with minimal redundancy of data.

199 3.1 XML Schema Structure for CuttingTool and CuttingToolArchetype

200 The *Figure 1* shows the XML schema that applies to both the *CuttingTool Information
201 Model* and the *CuttingToolArchetype Information Model*.

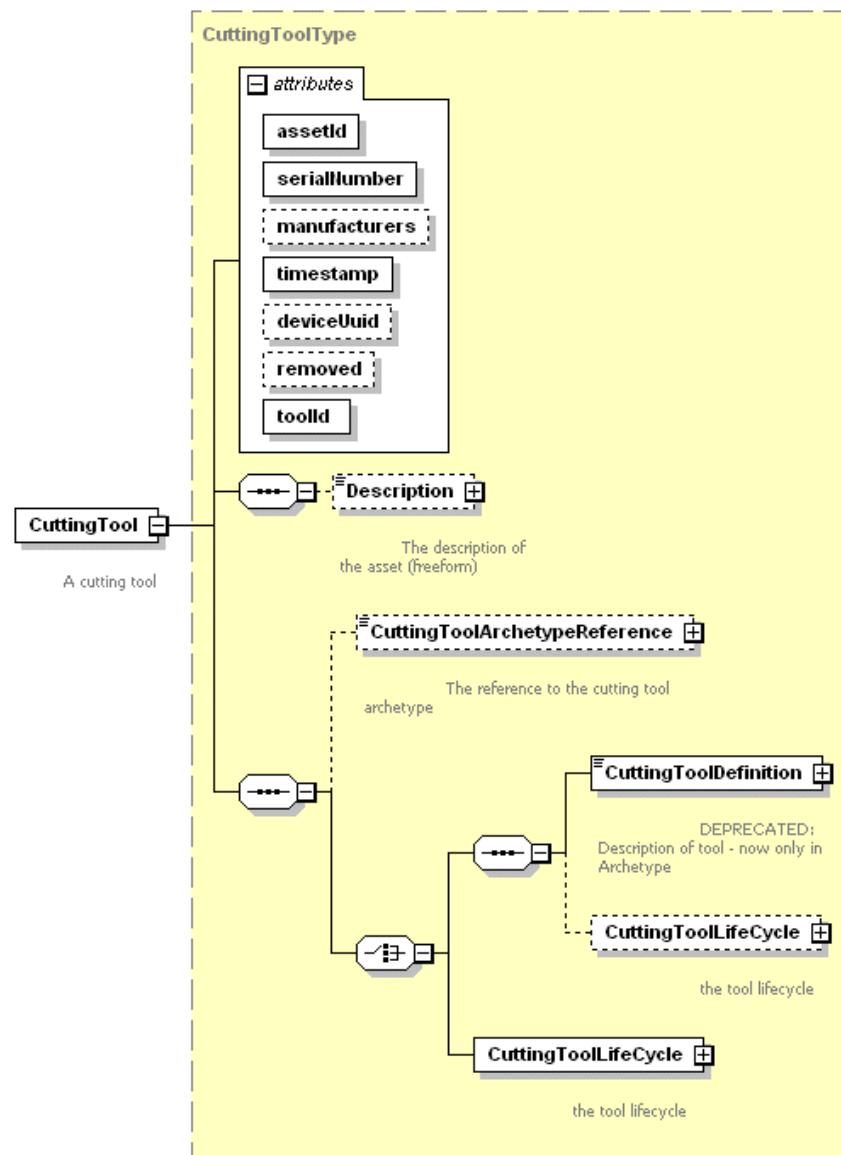


Figure 1: Cutting Tool Schema

202 Note: The use of the XML element `CuttingToolDefinition` has been **DEP-**
 203 **RECATED** in the `CuttingTool` schema, but remains in the `Cutting-`
 204 `ToolArchetype` schema.

205 The following sections contain the definitions of `CuttingTool` and `CuttingToolArchetype`
 206 and describe their unique components. The following are the common entities for both el-
 207 ements.

208 3.2 Common Attributes for CuttingTool and CuttingToolArchetype

Table 1: Attributes for CuttingTool and CuttingToolArchetype

| Attribute | Description | Occurrence |
|---------------------------|--|------------|
| <code>timestamp</code> | The time this <i>MTConnect Asset</i> was last modified. Always given in UTC. The <code>timestamp</code> MUST be provided in UTC (Universal Time Coordinate, also known as GMT). This is the time the <i>Asset</i> data was last modified. <code>timestamp</code> is a required attribute. | 1 |
| <code>assetId</code> | The unique identifier of the instance of this tool. This will be the same as the <code>toolId</code> and <code>serialNumber</code> in most cases. The <code>assetId</code> SHOULD be the combination of the <code>toolId</code> and <code>serialNumber</code> as in <code>toolId</code> , <code>serialNumber</code> or an equivalent implementation dependent identification scheme. <code>assetId</code> is a required attribute. <code>assetId</code> is a permanent identifier that will be associated with an <i>MTConnect Asset</i> for its entire life. | 1 |
| <code>serialNumber</code> | The unique identifier for this assembly. This is defined as an XML string type and is implementation dependent. <code>serialNumber</code> is a required attribute. | 1 |

| Continuation of Table 1 | | |
|-------------------------|--|------------|
| Attribute | Description | Occurrence |
| toolId | The identifier for a class of Cutting Tools. This is defined as an XML string type and is implementation dependent. toolId is a required attribute. | 1 |
| deviceUuid | The piece of equipments 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. | 1 |
| manufacturers | An optional attribute referring to the manufacturer(s) of this Cutting Tool, for this element, this will reference the Tool Item and Adaptive Items specifically. The Cutting Items manufacturers' will be an attribute of the CuttingItem elements. The representation will be a comma (,) delimited list of manufacturer names. This can be any series of numbers and letters as defined by the XML type string. | 0..1 |
| removed | This is an indicator that the Cutting Tool has been removed from the piece of equipment. removed is a required attribute. If the <i>MTConnect Asset</i> 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 |

209 3.3 Common Elements for CuttingTool and CuttingToolArchetype

Table 2: Common Elements for CuttingTool and CuttingToolArchetype

| Element | Description | Occurrence |
|-------------|--|------------|
| Description | An 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 |

210 3.3.1 Description Element for CuttingTool and CuttingToolArchetype

211 Description **MAY** contain mixed content, meaning that an additional XML element
 212 or plain text may be provided as part of the content of the description tag. Currently
 213 Description contains no attributes.

214 4 CuttingToolArchetype Information Model

215 The CuttingToolArchetype *Information Model* will have the identical structure as
 216 the CuttingTool *Information Model* illustrated in *Figure 1*, except for a few entities.
 217 The CuttingTool will no longer carry the CuttingToolDefinition, this **MUST**
 218 only appear in the CuttingToolArchetype. The CuttingToolArchetype **MUST**
 219 **NOT** have measured values and **MUST NOT** have any of the following items: Cutter-
 220 Status, ToolLife values, Location, or a ReconditionCount.

221 MTConnect Standard will adopt the ISO 13399 structure when formulating the vocabulary
 222 for Cutting Tool geometries and structure to be represented in the CuttingToolArchetype.
 223 The nominal values provided in the CuttingToolLifeCycle section are only con-
 224 cerned with two aspects of the Cutting Tool, the Cutting Tool and the Cutting Item. The
 225 Tool Item, Adaptive Item, and Assembly Item will only be covered in the Cutting-
 226 ToolDefinition section of this document since this section contains the full ISO
 227 13399 information about a Cutting Tool.



Figure 2: Cutting Tool Parts

228 The *Figure 2* illustrates the parts of a Cutting Tool. The Cutting Tool is the aggregate of
 229 all the components and the Cutting Item is the part of the tool that removes the material
 230 from the workpiece. These are the primary focus of the MTConnect Standard.

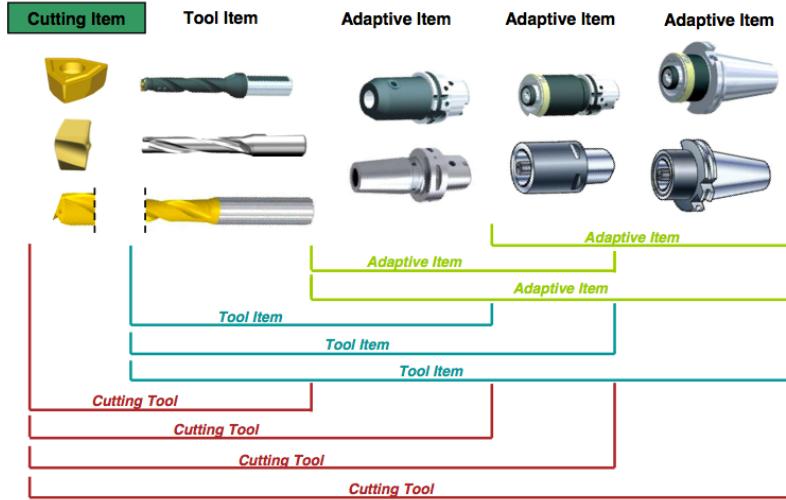


Figure 3: Cutting Tool Composition

231 *Figure 3* provides another view of the composition of a Cutting Tool. The Adaptive Items
 232 and Tool Items will be used for measurements, but will not be modeled as separate entities.
 233 When we are referencing the Cutting Tool we are referring to the entirety of the assembly
 234 and when we provide data regarding the Cutting Item we are referencing each individual
 235 item as illustrated on the left of the previous diagram.

236 *Figure 4* and *Figure 5* further illustrates the components of the Cutting Tool. As we
 237 compose the Tool Item, Cutting Item, Adaptive Item, we get a Cutting Tool. The Tool Item,
 238 Adaptive Item, and Assembly Item will only be in the CuttingToolDefinition
 239 section that will contain the full ISO 13399 information.

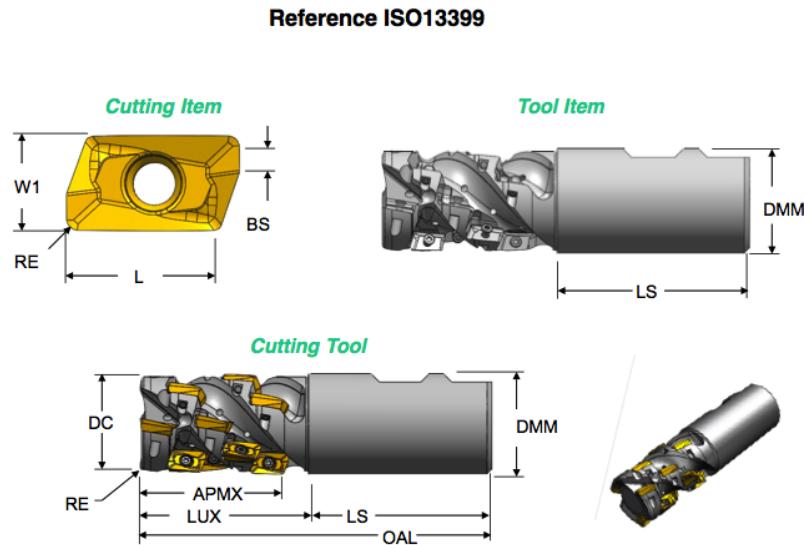


Figure 4: Cutting Tool, Tool Item, and Cutting Item

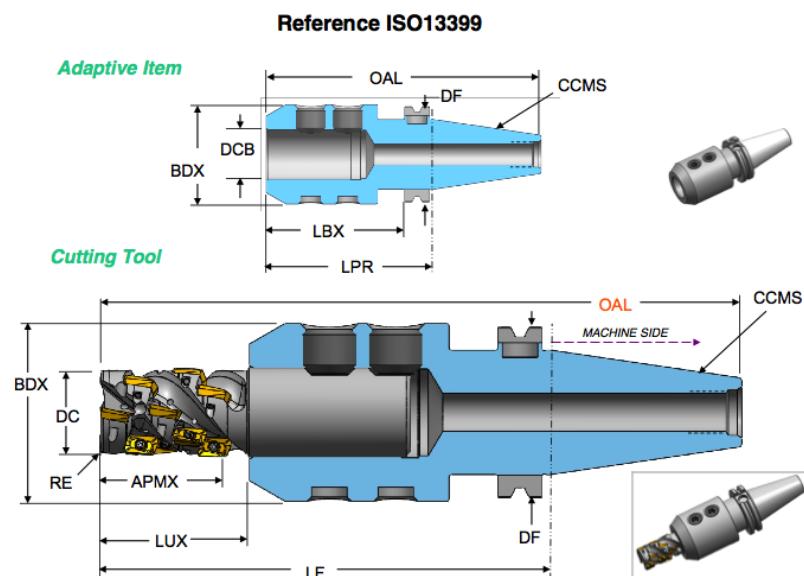


Figure 5: Cutting Tool, Tool Item, and Cutting Item 2

240 *Figure 4* and *Figure 5* use the ISO 13399 codes for each of the measurements. These
 241 codes will be translated into the MTConnect Standard vocabulary as illustrated below.
 242 The measurements will have a maximum, minimum, and nominal value representing the
 243 tolerance of allowable values for this dimension. See below for a full discussion.

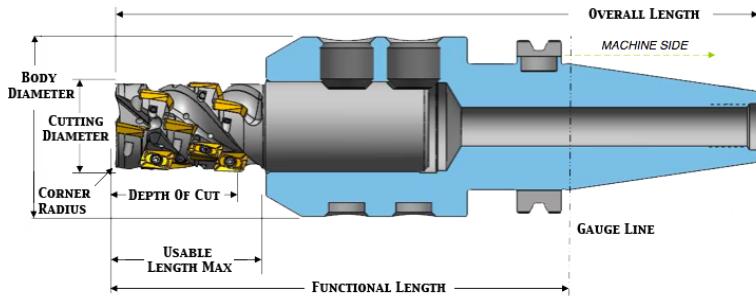


Figure 6: Cutting Tool Measurements

244 The MTConnect Standard will not define the entire geometry of the Cutting Tool, but will
 245 provide the information necessary to use the tool in the manufacturing process. Addi-
 246 tional information can be added to the definition of the Cutting Tool by means of schema
 247 extensions.

248 Additional diagrams will reference these dimensions by their codes that will be defined in
 249 the measurement tables. The codes are consistent with the codes used in ISO 13399 and
 250 have been standardized. MTConnect Standard will use the full text name for clarity in the
 251 XML document.

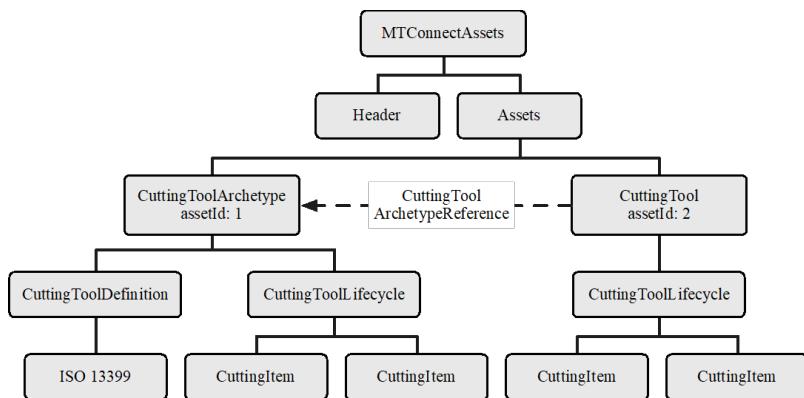


Figure 7: Cutting Tool Asset Structure

252 The structure of the MTConnectAssets header is defined in *MTConnect Standard Part*
 253 *1.0 - Overview and Fundamentals* of the Standard. A finite number of *MTConnect Assets*
 254 will be stored in the *Agent*. This finite number is implementation specific and will depend
 255 on memory and storage constraints. The standard will not prescribe the number or capacity
 256 requirements for an implementation.

257 4.1 Attributes for CuttingToolArchetype

258 Refer to *Section 3.2 - Common Attributes for CuttingTool and CuttingToolArchetype* for a
 259 full description of the attributes for CuttingToolArchetype *Information Model*.

260 4.2 Elements for CuttingToolArchetype

261 The elements associated with CuttingToolArchetype are given in *Table 3*. Each
 262 element will be described in more detail below and any possible values will be presented
 263 with full definitions. The elements **MUST** be provided in the following order as prescribed
 264 by XML. At least one of CuttingToolDefinition or CuttingToolLifeCycle
 265 **MUST** be supplied.

Table 3: Elements for CuttingToolArchetype

| Element | Description | Occurrence |
|-----------------------|--|------------|
| Description | An 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 |
| CuttingToolDefinition | Reference to an ISO 13399. | 0..1 |
| CuttingToolLifeCycle | Data regarding the use of this tool. The archetype will only contain nominal values. | 0..1 |

266 4.2.1 CuttingToolDefinition Element for CuttingToolArchetype

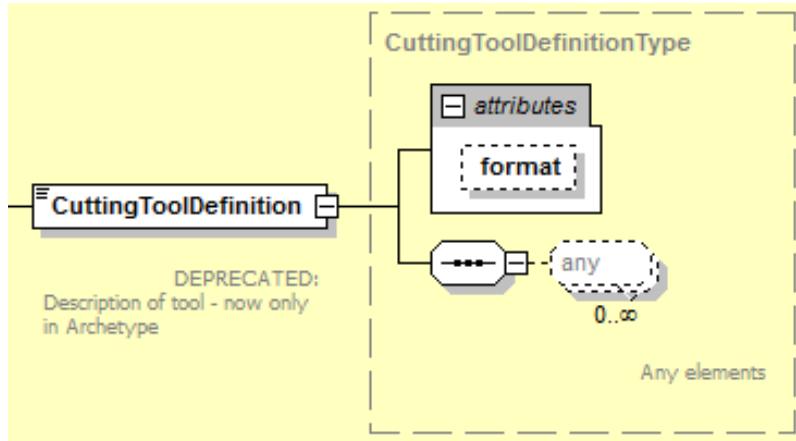


Figure 8: CuttingToolDefinition Schema

267 The CuttingToolDefinition contains the detailed structure of the Cutting Tool.
 268 The information contained in this element will be static during its lifecycle. Currently we
 269 are referring to the external ISO 13399 standard to provide the complete definition and
 270 composition of the Cutting Tool as defined in *Section 6.1 - CuttingToolLifeCycle*.

271 4.2.1.1 Attributes for CuttingToolDefinition

Table 4: Attributes for CuttingToolDefinition

| Attribute | Description | Occurrence |
|-----------|---|------------|
| format | <p>Identifies the expected representation of the enclosed data.</p> <p>format is an optional attribute.</p> <p>Valid values of format are – XML, EXPRESS, TEXT, or UNDEFINED.</p> <p>If format is not specified, the assumed format is XML.</p> | 0..1 |

272 4.2.1.1.1 format Attribute for CuttingToolDefinition

273 The format attribute describes the expected representation of the enclosed data. If no
 274 value is given, the assumed format will be XML.

Table 5: Values for format attribute of CuttingToolDefinition

| Value | Description |
|-----------|--|
| XML | The default value for the definition. The content will be an XML document. |
| EXPRESS | The document will conform to the ISO 10303 Part 21 standard. |
| TEXT | The document will be a text representation of the tool data. |
| UNDEFINED | The document will be provided in an undefined format. |

275 **4.2.1.2 Elements for CuttingToolDefinition**

276 The only acceptable Cutting Tool definition at present is defined by the ISO 13399 stan-
277 dard. Additional formats **MAY** be considered in the future.

278 **4.2.1.3 ISO13399 Standard**

279 The ISO 13399 data **MUST** be presented in either XML (ISO 10303-28) or EXPRESS
280 format (ISO 10303-21). An XML schema will be preferred as this will allow for easier
281 integration with the MTConnect Standard XML tools. EXPRESS will also be supported,
282 but software tools will need to be provided or made available for handling this data repre-
283 sentation.

284 There will be the root element of the ISO13399 document when XML is used. When
285 EXPRESS is used the XML element will be replaced by the text representation.

286 **4.2.2 CuttingToolLifeCycle Element for CuttingToolArchetype**

287 Refer to *Section 6 - Common Entity CuttingToolLifeCycle* for a complete description of
288 CuttingToolLifeCycle element.

289 5 CuttingTool Information model

290 The CuttingTool *Information Model* illustrated in *Figure 1* has the identical struc-
 291 ture as the CuttingToolArchetype *Information Model* except for the XML ele-
 292 ment CuttingToolDefinition that has been **DEPRECATED** in the Cutting-
 293 Tool schema.

294 5.1 Attributes for CuttingTool

295 Refer to *Section 3.2 - Common Attributes for CuttingTool and CuttingToolArchetype* for a
 296 full description of the *Attributes for CuttingTool Information Model*.

297 5.2 Elements for CuttingTool

298 The elements associated with CuttingTool are given below. The elements **MUST** be
 299 provided in the order shown in *Table 6* as prescribed by XML.

Table 6: Elements for CuttingTool

| Element | Description | Occurrence |
|-----------------------|--|------------|
| Description | An 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 |
| CuttingToolDefinition | DEPRECATED for CuttingTool in Version 1.3.0. Reference to an ISO 13399. | 0..1 |

| Continuation of Table 6 | | |
|-------------------------------|--|------------|
| Element | Description | Occurrence |
| CuttingToolLifeCycle | Data regarding the use of this tool. | 0..1 |
| CuttingToolArchetypeReference | The content of this XML element is the assetId of the CuttingToolArchetype document. It MAY also contain a source attribute that gives the URL of the archetype data as well. | 0..1 |

300 5.2.1 CuttingToolLifeCycle Elements for CuttingTool Only

301 The following CuttingToolLifeCycle elements are used only in the Cutting-
 302 Tool *Information Model* and are not part of the CuttingToolArchetype *Information Model*. Refer to *Section 6 - Common Entity CuttingToolLifeCycle* for a complete
 304 description of the remaining elements for CuttingToolLifeCycle that are common
 305 in both *Information Models*. Refer also to the CuttingToolLifeCycle schema illus-
 306 trated in *Figure 14*.

307 5.2.1.1 CutterStatus Element for CuttingToolLifeCycle

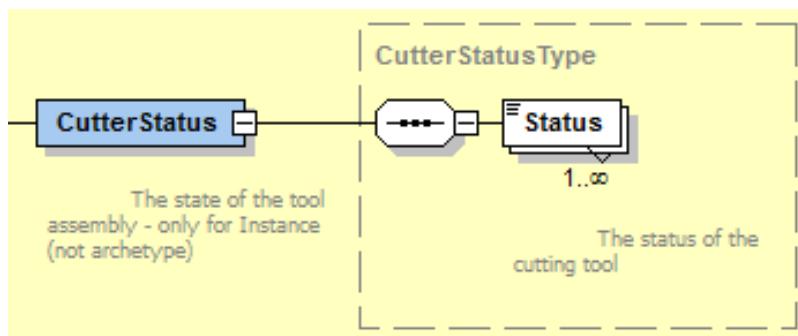


Figure 9: CutterStatus Schema

308 The elements of the CutterStatus element can be a combined set of Status ele-
 309 ments. The *MTConnect Standard* allows any set of statuses to be combined, but only
 310 certain combinations make sense. A CuttingTool **SHOULD** not be both NEW and

311 USED at the same time. There are no rules in the schema to enforce this, but this is left to
 312 the implementer. The following combinations **MUST NOT** occur:

- 313 • NEW **MUST NOT** be used with USED, RECONDITIONED, or EXPIRED.
- 314 • UNKNOWN **MUST NOT** be used with any other status.
- 315 • ALLOCATED and UNALLOCATED **MUST NOT** be used together.
- 316 • AVAILABLE and UNAVAILABLE **MUST NOT** be used together.
- 317 • If the tool is EXPIRED, BROKEN, or NOT_REGISTERED it **MUST NOT** be AVAIL-
 318 ABLE.
- 319 • All other combinations are allowed.

Table 7: Elements for CutterStatus

| Element | Description | Occurrence |
|---------|--|------------|
| Status | The status of the Cutting Tool. There can be multiple Status elements. | 1..* |

320 **5.2.1.1.1 Status Element for CutterStatus**

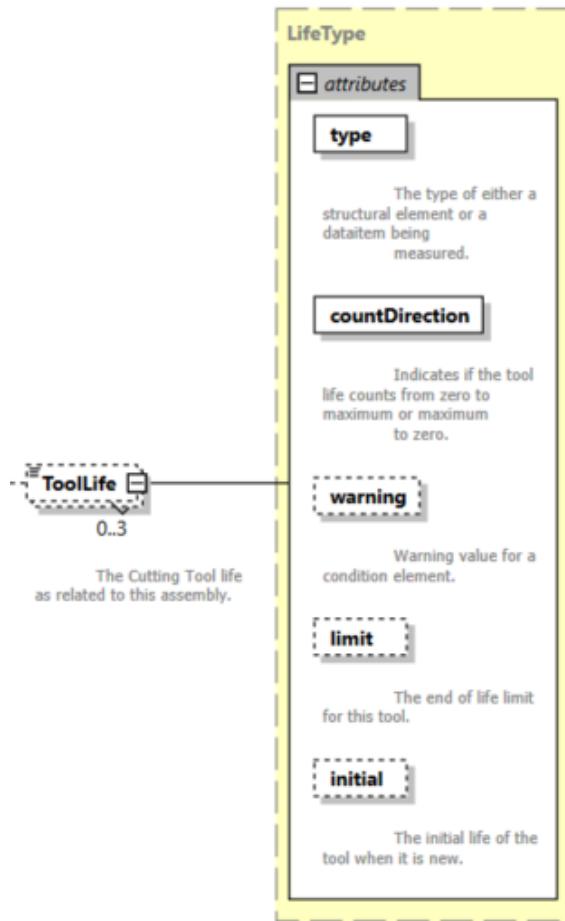
321 One of the values for the status of the CuttingTool.

Table 8: Values for Status Element of CutterStatus

| Value | Description |
|-------------|---|
| NEW | A new tool that has not been used or first use. Marks the start of the tool history. |
| AVAILABLE | Indicates the tool is available for use. If this is not present, the tool is currently not ready to be used. |
| UNAVAILABLE | Indicates the tool is unavailable for use in metal removal. If this is not present, the tool is currently not ready to be used. |

| Continuation of Table 8 | |
|-------------------------|--|
| Value | Description |
| ALLOCATED | Indicates if this tool is has been committed to a piece of equipment for use and is not available for use in any other piece of equipment. If this is not present, this tool has not been allocated for this piece of equipment and can be used by another piece of equipment. |
| UNALLOCATED | Indicates this Cutting Tool has not been committed to a process and can be allocated. |
| MEASURED | The tool has been measured. |
| RECONDITIONED | The Cutting Tool has been reconditioned. See ReconditionCount for the number of times this cutter has been reconditioned. |
| USED | The Cutting Tool is in process and has remaining tool life. |
| EXPIRED | The Cutting Tool has reached the end of its useful life. |
| BROKEN | Premature tool failure. |
| NOT_REGISTERED | This Cutting Tool cannot be used until it is entered into the system. |
| UNKNOWN | The Cutting Tool is an indeterminate state. This is the default value. |

322 5.2.1.2 ToolLife Element for CuttingToolLifeCycle

**Figure 10:** ToolLife Schema

323 The value is the current value for the **ToolLife**. The value **MUST** be a number. **ToolLife**
 324 is an option element which can have three types, either minutes for time based, part
 325 count for parts based, or wear based using a distance measure. One **ToolLife** element
 326 can appear for each type, but there cannot be two entries of the same type. Additional
 327 types can be added in the future.

328 **5.2.1.2.1 Attributes for ToolLife**

329 ToolLife has the following attributes that can be used to indicate the behavior of the
 330 tool life management mechanism.

Table 9: Attributes for ToolLife

| Attribute | Description | Occurrence |
|----------------|--|------------|
| type | The type of tool life being accumulated. MINUTES, PART_COUNT, or WEAR. type is a required attribute. | 1 |
| countDirection | Indicates if the tool life counts from zero to maximum or maximum to zero. The value MUST be one of UP or DOWN. countDirection is a required attribute. | 1 |
| warning | The point at which a tool life warning will be raised. warning is an optional attribute. | 0..1 |
| limit | The end of life limit for this tool. If the countDirection is DOWN, the point at which this tool should be expired, usually zero. If the countDirection is UP, this is the upper limit for which this tool should be expired. limit is an optional attribute. | 0..1 |
| initial | The initial life of the tool when it is new. initial is an optional attribute. | 0..1 |

331 **5.2.1.2.2 type Attribute for ToolLife**

332 The value of type must be one of the following:

Table 10: Values for type of ToolLife

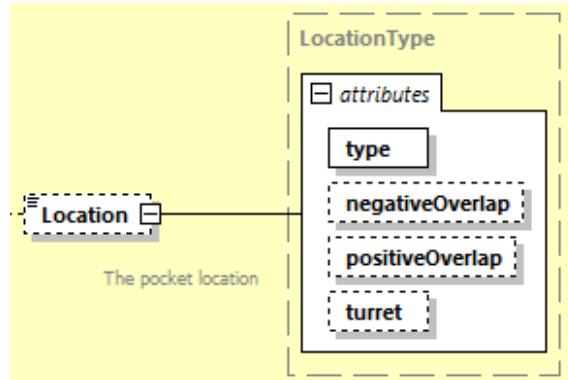
| Value | Description |
|------------|--|
| MINUTES | The tool life measured in minutes. All units for minimum, maximum, and nominal MUST be provided in minutes. |
| PART_COUNT | The tool life measured in parts. All units for minimum, maximum, and nominal MUST be provided as the number of parts. |
| WEAR | The tool life measured in tool wear. Wear MUST be provided in millimeters as an offset to nominal. All units for minimum, maximum, and nominal MUST be given as millimeter offsets as well. The standard will only consider dimensional wear at this time. |

333 **5.2.1.2.3 countDirection Attribute for ToolLife**

334 The value of countDirection must be one of the following:

Table 11: Values for countDirection

| Value | Description |
|-------|---|
| UP | The tool life counts up from zero to the maximum. |
| DOWN | The tool life counts down from the maximum to zero. |

335 **5.2.1.3 Location Element for CuttingToolLifeCycle****Figure 11:** Location Schema

336 Location element identifies the specific location where a tool resides in a piece of equip-

337 ment tool storage or in a tool crib. This can be any series of numbers and letters as defined
 338 by the XML type NMTOKEN. When a POT or STATION type is used, the value **MUST**
 339 be a numeric value. If a negativeOverlap or the positiveOverlap is provided,
 340 the tool reserves additional locations on either side, otherwise if they are not given, no
 341 additional locations are required for this tool. If the pot occupies the first or last location,
 342 a rollover to the beginning or the end of the index-able values may occur. For example, if
 343 there are 64 pots and the tool is in pot 64 with a positiveOverlap of 1, the first pot
 344 **MAY** be occupied as well.

345 **5.2.1.3.1 Attributes for Location**

Table 12: Attributes for Location

| Attribute | Description | Occurrence |
|-----------------|--|------------|
| type | The type of location being identified. type MUST be one of POT, STATION, or CRIB. type is a required attribute. | 1 |
| positiveOverlap | The number of locations at higher index value from this location. positiveOverlap is a optional attribute. | 0..1 |
| negativeOverlap | The number of location at lower index values from this location. negativeOverlap is an optional attribute. | 0..1 |

346 **5.2.1.3.2 type Attribute for Location**

347 The type of location being identified.

Table 13: Values for type of Location

| Value | Description |
|---------|--|
| POT | The number of the pot in the tool handling system. |
| STATION | The tool location in a horizontal turning machine. |
| CRIB | The location with regard to a tool crib. |

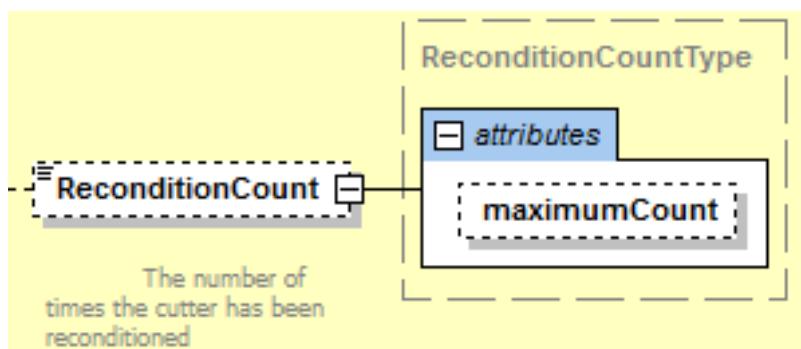
348 **5.2.1.3.3 positiveOverlap Attribute for Location**

349 The number of locations at higher index values that the CuttingTool occupies due to
 350 interference. The value **MUST** be an integer. If not provided it is assumed to be 0.

351 **5.2.1.3.4 negativeOverlap Attribute for Location**

352 The number of locations at lower index values that the CuttingTool occupies due to
 353 interference. The value **MUST** be an integer. If not provided it is not assumed to be 0.

354 The tool number assigned in the part program and is used for cross referencing this tool
 355 information with the process parameters. The value **MUST** be an integer.

356 **5.2.1.4 ReconditionCount Element for CuttingToolLifeCycle****Figure 12:** ReconditionCount Schema

357 This element **MUST** contain an integer value as the CDATA that represents the number of
 358 times the cutter has been reconditioned.

359 **5.2.1.4.1 Attributes for ReconditionCount****Table 14:** Attributes for ReconditionCount

| Attribute | Description | Occurrence |
|--------------|--|------------|
| maximumCount | The maximum number of times this tool may be reconditioned. maximumCount is a optional attribute. | 0..1 |

360 5.2.2 CuttingToolArchetypeReference Element for Cutting Tool

361



Figure 13: CuttingToolArcheTypeReference Schema

362 This optional element references another *MTConnect Asset* document providing the static
 363 geometries and nominal values for all the measurements. This reduces the amount of data
 364 duplication as well as providing a mechanism for asset definitions to be provided before
 365 complete measurement has occurred.

366 5.2.2.1 source Attribute for CuttingToolArcheTypeReference

Table 15: Attributes for CuttingToolArchetypeReference

| Attribute | Description | Occurrence |
|-----------|--|------------|
| source | <p>The URL of the <i>CuttingToolArchetype Information Model</i>.</p> <p>This MUST be a fully qualified URL as in http://example.com/asset/A213155</p> | 0..1 |

367 **6 Common Entity CuttingToolLifeCycle**

368 **6.1 CuttingToolLifeCycle**

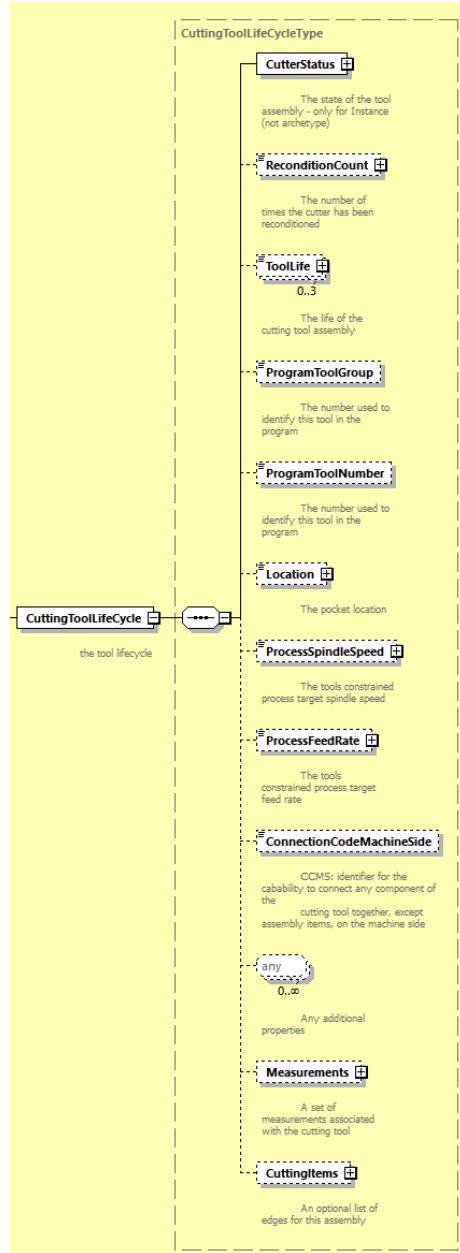
369 The life cycle refers to the data pertaining to the application or the use of the tool. This
370 data is provided by various pieces of equipment (i.e. machine tool, presetter) and statistical
371 process control applications. Life cycle data will not remain static, but will change
372 periodically when a tool is used or measured. The life cycle has three conceptual parts;
373 CuttingTool and CuttingItem identity, properties, and measurements. A measurement
374 is defined as a constrained value that is reported in defined units and as a W3C
375 floating point format.

376 The CuttingToolLifeCycle contains data for the entire tool assembly. The specific
377 CuttingItems that are part of the CuttingToolLifeCycle are contained in the
378 CuttingItems element. Each Cutting Item has similar properties as the assembly;
379 identity, properties, and Measurements.

380 The units for all Measurements have been predefined in the *MTConnect Standard* and
381 will be consistent with *MTConnect Standard: Part 2.0 - Devices Information Model* and
382 *MTConnect Standard: Part 3.0 - Streams Information Model*. This means that all lengths
383 and distances will be given in millimeters and all angular measures will be given in degrees.
384 Quantities like ProcessSpindleSpeed will be given in RPM, the same as the
385 ROTARY_VELOCITY in *MTConnect Standard: Part 3.0 - Streams Information Model*.

386 **6.1.1 XML Schema Structure for CuttingToolLifeCycle**

387 The CuttingToolLifeCycle schema shown in *Figure 14* is used in both the CuttingToolArchetype and CuttingTool *Information Models*. The only difference
388 is that the elements CutterStatus, ToolLife, Location, and Recondition-
389 Count are used only in the CuttingTool *Information Model*.

**Figure 14:** CuttingToolLifeCycle Schema

391 6.2 Elements for CuttingToolLifeCycle

392 The elements associated with this Cutting Tool are given in *Table 16*. The elements **MUST**
 393 be provided in the following order as prescribed by XML.

Table 16: Elements for CuttingToolLifeCycle

| Element | Description | Occurrence |
|------------------|---|------------|
| CutterStatus | <p>The status of this assembly.</p> <p>CutterStatus can be one of the following values: NEW, AVAILABLE, UNAVAILABLE, ALLOCATED, UNALLOCATED, MEASURED, RECONDITIONED, NOT_REGISTERED, USED, EXPIRED, BROKEN, or UNKNOWN.</p> <p>MUST only be used in the <i>CuttingTool Information Model</i>.</p> | 1 |
| ReconditionCount | <p>The number of times this cutter has been reconditioned.</p> <p>MUST only be used in the <i>CuttingTool Information Model</i>.</p> | 0..1 |
| ToolLife | <p>The Cutting Tool life as related to this assembly.</p> <p>MUST only be used in the <i>CuttingTool Information Model</i>.</p> | 0..1 |
| Location | <p>The Pot or Spindle this tool currently resides in.</p> <p>MUST only be used in the <i>CuttingTool Information Model</i>.</p> | 0..1 |

| Continuation of Table 16 | | |
|---------------------------|---|------------|
| Element | Description | Occurrence |
| ProgramToolGroup | The tool group this tool is assigned in the part program. | 0..1 |
| ProgramToolNumber | The number of the tool as referenced in the part program. | 0..1 |
| ProcessSpindleSpeed | The constrained process spindle speed for this tool. | 0..1 |
| ProcessFeedRate | The constrained process feed rate for this tool in mm/s. | 0..1 |
| ConnectionCodeMachineSide | Identifier for the capability to connect any component of the Cutting Tool together, except Assembly Items, on the machine side. Code: CCMS | 0..1 |
| Measurements | A collection of measurements for the tool assembly. | 0..1 |
| CuttingItems | An optional set of individual Cutting Items. | 0..1 |
| xs:any | Any additional properties not in the current document model. MUST be in separate XML namespace. | 0..n |

394 6.2.1 ProgramToolGroup Element for CuttingToolLifeCycle

395 The optional identifier for the group of Cutting Tools when multiple tools can be used
 396 interchangeably. This is defined as an XML string type and is implementation dependent.

397 6.2.2 ProgramToolNumber Element for CuttingToolLifeCycle

398 The tool number assigned in the part program and is used for cross referencing this tool
 399 information with the process parameters. The value **MUST** be an integer.

400 6.2.3 ProcessSpindleSpeed Element for CuttingToolLifeCycle

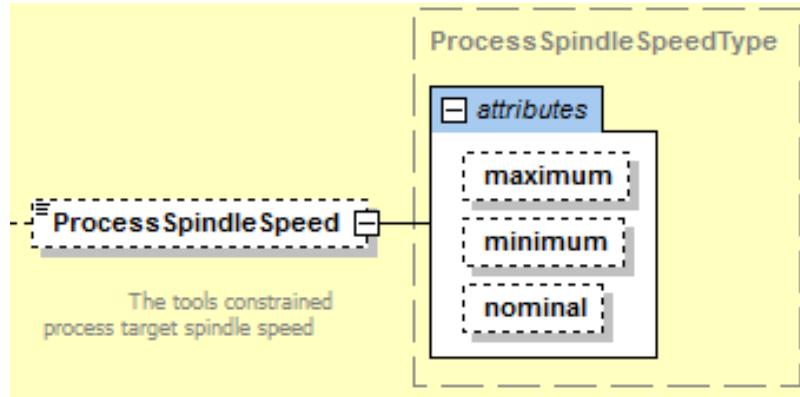


Figure 15: ProcessSpindleSpeed Schema

401 The **ProcessSpindleSpeed** **MUST** be specified in revolutions/minute (RPM). The
 402 CDATA **MAY** contain the nominal process target spindle speed if available. The maximum
 403 and minimum speeds **MAY** be provided as attributes. If **ProcessSpindleSpeed** is
 404 provided, at least one value of maximum, nominal, or minimum **MUST** be specified.

405 6.2.3.1 Attributes for ProcessSpindleSpeed

Table 17: Attributes for ProcessSpindleSpeed

| Attribute | Description | Occurrence |
|-----------|--|------------|
| maximum | The upper bound for the tool's target spindle speed. maximum is an optional attribute. | 0..1 |
| minimum | The lower bound for the tools spindle speed. minimum is a optional attribute. | 0..1 |
| nominal | The nominal speed the tool is designed to operate at. nominal is an optional attribute. | 0..1 |

406 6.2.4 ProcessFeedRate Element for CuttingToolLifeCycle

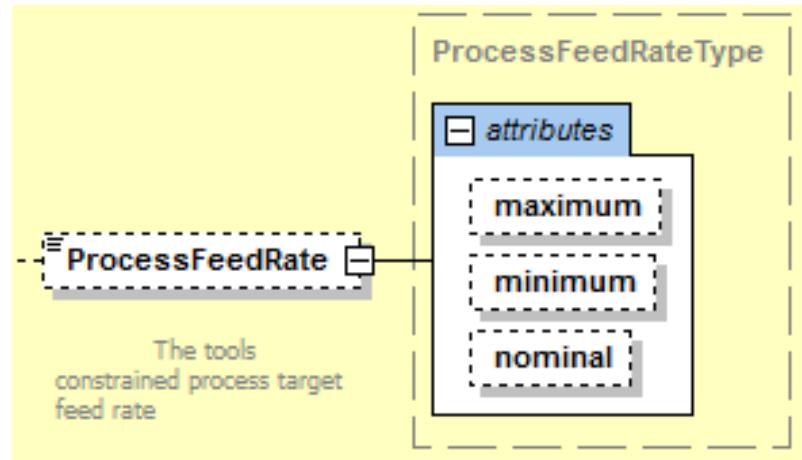


Figure 16: ProcessFeedRate Schema

407 The ProcessFeedRate **MUST** be specified in millimeters/second (mm/s). The CDATA
 408 **MAY** contain the nominal process target feed rate if available. The maximum and mini-
 409 mum rates **MAY** be provided as attributes. If ProcessFeedRate is provided, at least
 410 one value of maximum, nominal, or minimum **MUST** be specified.

411 6.2.4.1 Attributes for ProcessFeedRate

Table 18: Attributes for ProcessFeedRate

| Attribute | Description | Occurrence |
|-----------|---|------------|
| maximum | The upper bound for the tool's process target feedrate. maximum is an optional attribute. | 0..1 |
| minimum | The lower bound for the tools feedrate. minimum is a optional attribute. | 0..1 |
| nominal | The nominal feedrate the tool is designed to operate at. nominal is an optional attribute. | 0..1 |

412 **6.2.5 ConnectionCodeMachineSide Element for CuttingToolLifeCy-**
 413 **cle**

414 This is an optional identifier for implementation specific connection component of the
 415 Cutting Tool on the machine side. Code: CCMS. The CDATA **MAY** be any valid string
 416 according to the referenced connection code standards.

417 **6.2.6 xs:any Element for CuttingToolLifeCycle**

418 Utilizing the new capability in *XML Schema* Version 1.1, there are extension points where
 419 an additional element can be added to the document without being part of a substitution
 420 group. The new elements have the restriction that they **MUST NOT** be part of the *MT-*
 421 *Connect namespace* and **MUST NOT** be one of the predefined elements mentioned above.

422 This allows one to add additional properties to the *CuttingTool* without having to
 423 change the definition of the *CuttingTool* or modify the standard. The new capabilities
 424 were introduced in Version 1.3 of the *MTConnect Standard* and necessitate using Version
 425 1.1 of *XML Schema* to make use of this form of extensible properties.

426 **6.2.7 Measurements Element for CuttingToolLifeCycle**

427 The Measurements element is a collection of one or more constrained scalar values
 428 associated with this Cutting Tool. The XML element **MUST** be a type extension of the
 429 base types CommonMeasurement or AssemblyMeasurement. The following sec-
 430 tion defines the abstract Measurement type used in both CuttingToolLifeCycle
 431 and CuttingItem. This subsequent sections describe the AssemblyMeasurement
 432 types followed by the CuttingItemMeasurement types.

433 A Measurement is specific to the tool management policy at a particular shop. The tool
 434 zero reference point or gauge line will be different depending on the particular implemen-
 435 tation and will be assumed to be consistent within the shop. *MTConnect Standard* does
 436 not standardize the manufacturing process or the definition of the zero point.

437 6.2.8 Measurement

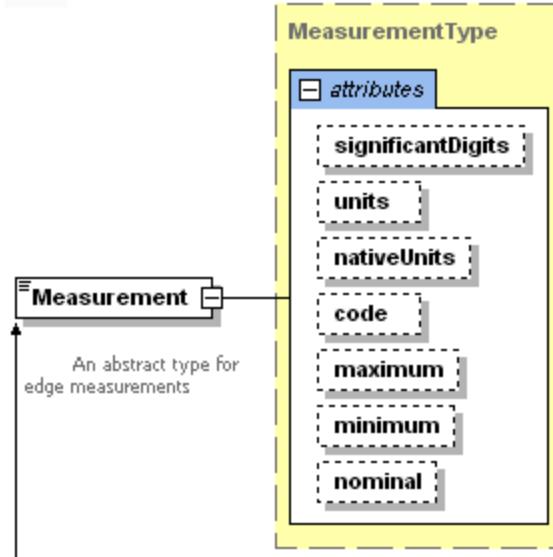


Figure 17: Measurement Schema

438 A Measurement **MUST** be a scalar floating-point value that **MAY** be constrained to a
 439 maximum and minimum value. Since the *CuttingToolLifeCycle*'s main responsi-
 440 bility is to track aspects of the tool that change over its use in the shop, *MTConnect* repre-
 441 sents the current value of the Measurement **MUST** be in the CDATA (text between the
 442 start and end element) as the most current valid value.

443 The minimum and maximum **MAY** be supplied if they are known or relevant to the
 444 Measurement. A nominal value **MAY** be provided to show the reference value for
 445 this Measurement.

446 There are three abstract subtypes of Measurement: CommonMeasurement, Assem-
 447 blyMeasurement, and CuttingItemMeasurement. These abstract types **MUST**
 448 **NOT** appear in an *MTConnectAssets* document, but are used in the schema as a way
 449 to separate which measurements **MAY** appear in the different sections of the document.
 450 Only subtypes that have extended these types **MAY** appear in the *MTConnectAssets*
 451 XML.

452 Measurements in the *CuttingToolLifeCycle* section **MUST** refer to the en-
 453 tire assembly and not to an individual *CuttingItem*. *CuttingItem* measurements
 454 **MUST** be located in the measurements associated with the individual *CuttingItem*.

455 Measurements **MAY** provide an optional units attribute to reinforce the given units.
 456 The units **MUST** always be given in the predefined *MTConnect* units. If units are

457 provided, they are only for documentation purposes. `nativeUnits` **MAY** optionally be
 458 provided to indicate the original units provided for the measurements.

459 **6.2.8.1 Attributes for Measurement**

Table 19: Attributes for Measurement

| Attribute | Description | Occurrence |
|--------------------------------|---|------------|
| <code>code</code> | A shop specific code for this measurement. ISO 13399 codes MAY be used for these codes as well. <code>code</code> is a optional attribute. | 0..1 |
| <code>maximum</code> | The maximum value for this measurement. Exceeding this value would indicate the tool is not usable. <code>maximum</code> is a optional attribute. | 0..1 |
| <code>minimum</code> | The minimum value for this measurement. Exceeding this value would indicate the tool is not usable. <code>minimum</code> is a optional attribute. | 0..1 |
| <code>nominal</code> | The as advertised value for this measurement. <code>nominal</code> is a optional attribute. | 0..1 |
| <code>significantDigits</code> | The number of significant digits in the reported value. This is used by applications to determine accuracy of values. This MAY be specified for all numeric values. <code>significantDigits</code> is a optional attribute. | 0..1 |

| Continuation of Table 19 | | |
|--------------------------|--|------------|
| Attribute | Description | Occurrence |
| units | The units for the measurements. MTConnect Standard defines all the units for each measurement, so this is mainly for documentation sake. See <i>MTConnect Standard: Part 2.0 - Devices Information Model</i> 7.2.2.5 for the full list of units. units is a optional attribute. | 0..1 |
| nativeUnits | The units the measurement was originally recorded in. This is only necessary if they differ from units. See <i>MTConnect Standard: Part 2.0 - Devices Information Model</i> Section 7.2.2.6 for the full list of units. nativeUnits is a optional attribute. | 0..1 |

460 6.2.8.2 Measurement Subtypes for CuttingToolLifeCycle

461 These Measurements for CuttingTool are specific to the entire assembly and **MUST**
 462 **NOT** be used for the Measurement pertaining to a CuttingItem. *Figure 18* and *Fig-
 463 ure 19* will be used to reference the assembly specific Measurements.

464 The Code in *Table 20* will refer to the acronyms in the diagrams. We will be referring to
 465 many diagrams to disambiguate all measurements of the CuttingTool and Cuttin-
 466 gItem.

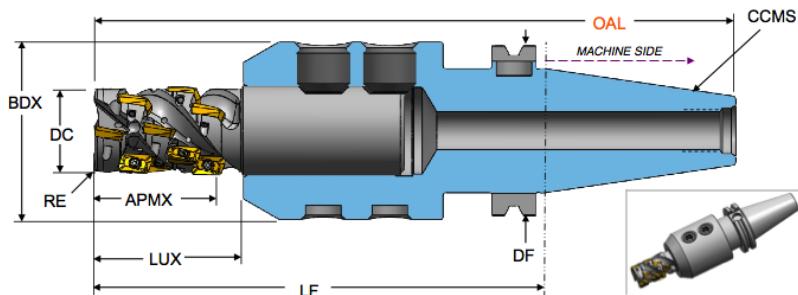
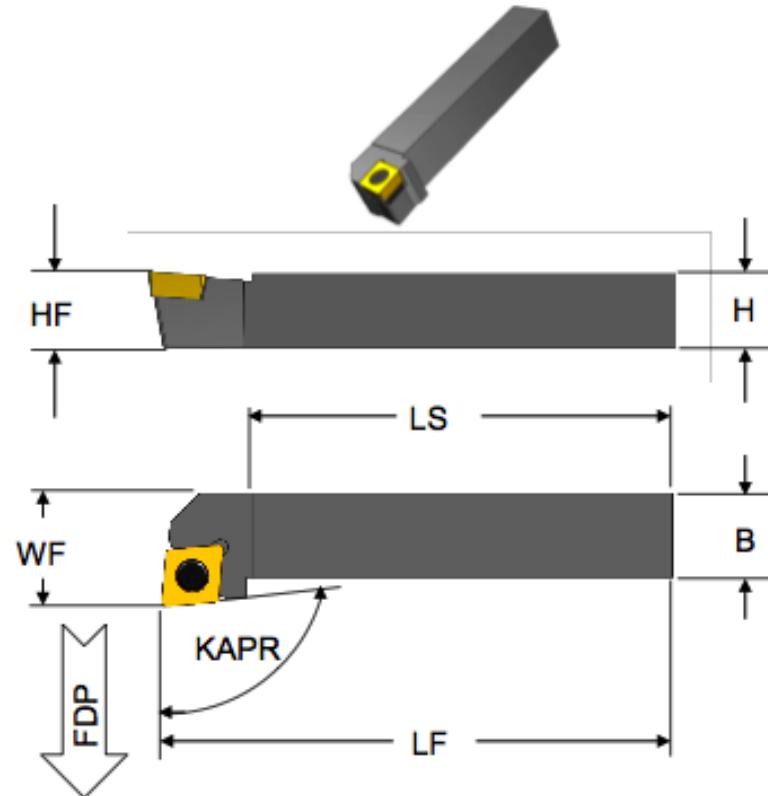


Figure 18: Cutting Tool Measurement Diagram 1

**Figure 19:** Cutting Tool Measurement Diagram 2**Table 20:** Measurement Subtypes for CuttingTool

| Measurement Subtype | Code | Description | Units |
|---------------------|------|--|------------|
| BodyDiameterMax | BDX | The largest diameter of the body of a Tool Item. | MILLIMETER |

| Continuation of Table 20 | | | |
|--------------------------|------|--|------------|
| Measurement Subtype | Code | Description | Units |
| BodyLengthMax | LBX | The distance measured along the X axis from that point of the item closest to the workpiece, including the Cutting Item for a Tool Item but excluding a protruding locking mechanism for an Adaptive Item, to either the front of the flange on a flanged body or the beginning of the connection interface feature on the machine side for cylindrical or prismatic shanks. | MILLIMETER |
| DepthOfCutMax | APMX | The maximum engagement of the cutting edge or edges with the workpiece measured perpendicular to the feed motion. | MILLIMETER |
| CuttingDiameterMax | DC | The maximum diameter of a circle on which the defined point Pk of each of the master inserts is located on a Tool Item. The normal of the machined peripheral surface points towards the axis of the Cutting Tool. | MILLIMETER |
| FlangeDiameterMax | DF | The dimension between two parallel tangents on the outside edge of a flange. | MILLIMETER |
| OverallToolLength | OAL | The largest length dimension of the Cutting Tool including the master insert where applicable. | MILLIMETER |

| Continuation of Table 20 | | | |
|--------------------------|------|---|------------|
| Measurement Subtype | Code | Description | Units |
| ShankDiameter | DMM | The dimension of the diameter of a cylindrical portion of a Tool Item or an Adaptive Item that can participate in a connection. | MILLIMETER |
| ShankHeight | H | The dimension of the height of the shank. | MILLIMETER |
| ShankLength | LS | The dimension of the length of the shank. | MILLIMETER |
| UsableLengthMax | LUX | Maximum length of a Cutting Tool that can be used in a particular cutting operation including the non-cutting portions of the tool. | MILLIMETER |
| ProtrudingLength | LPR | The dimension from the yz-plane to the furthest point of the Tool Item or Adaptive Item measured in the -X direction. | MILLIMETER |
| Weight | WT | The total weight of the Cutting Tool in grams. The force exerted by the mass of the Cutting Tool. | GRAM |

| Continuation of Table 20 | | | |
|--------------------------|------|---|------------|
| Measurement Subtype | Code | Description | Units |
| FunctionalLength | LF | The distance from the gauge plane or from the end of the shank to the furthest point on the tool, if a gauge plane does not exist, to the cutting reference point determined by the main function of the tool. The CuttingTool functional length will be the length of the entire tool, not a single Cutting Item. Each CuttingItem can have an independent FunctionalLength represented in its measurements. | MILLIMETER |

467 6.2.9 CuttingItems Element for CuttingToolLifeCycle

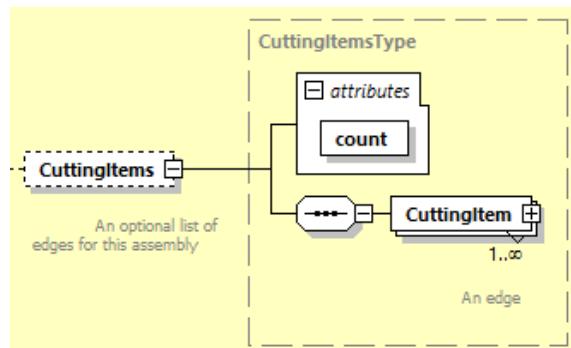


Figure 20: CuttingItems Schema

468 An optional collection of CuttingItems that **SHOULD** be provided for each indepen-
 469 dent edge or insert. If the CuttingItems are not present; it indicates there is no specific
 470 information with respect to each of the CuttingItems. This does not imply there are no
 471 CuttingItems – there **MUST** be at least one CuttingItem – but there is no specific
 472 information.

473 **6.2.9.1 Attributes for CuttingItems****Table 21:** Attributes for CuttingItems

| Attribute | Description | Occurrence |
|-----------|---|------------|
| count | The number of Cutting Item. count is a required attribute. | 1 |

474 **6.2.10 CuttingItem**

475 A CuttingItem is the portion of the tool that physically removes the material from the
 476 workpiece by shear deformation. The Cutting Item can be either a single piece of mate-
 477 rial attached to the CuttingItem or it can be one or more separate pieces of material
 478 attached to the CuttingItem using a permanent or removable attachment. A Cut-
 479 tingItem can be comprised of one or more cutting edges. CuttingItems include:
 480 replaceable inserts, brazed tips and the cutting portions of solid CuttingTools.

481 MTConnect Standard considers CuttingItems as part of the CuttingTool. A Cut-
 482 tingItems **MUST NOT** exist in MTConnect unless it is attached to a CuttingTool.
 483 Some of the measurements, such as FunctionalLength, **MUST** be made with refer-
 484 ence to the entire CuttingTool to be meaningful.

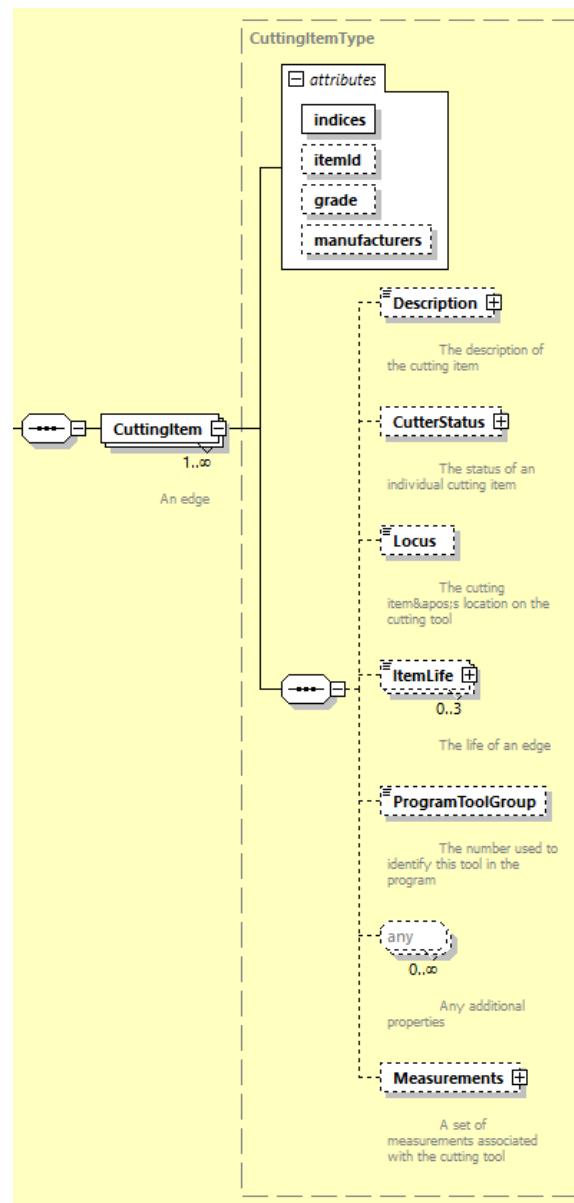


Figure 21: CuttingItem Schema

485 **6.2.10.1 Attributes for CuttingItem****Table 22:** Attributes for CuttingItem

| Attribute | Description | Occurrence |
|---------------|--|------------|
| indices | The number or numbers representing the individual Cutting Item or items on the tool. indices is a required attribute. | 1 |
| itemId | The manufacturer identifier of this Cutting Item. itemId is an optional attribute. | 0..1 |
| manufacturers | The manufacturers of the Cutting Item or Tool. manufacturers is an optional attribute. | 0..1 |
| grade | The material composition for this Cutting Item. grade is an optional attribute. | 0..1 |

486 **6.2.10.1.1 indices Attribute for CuttingItem**

487 An identifier that indicates the CuttingItem or CuttingItems these data are as-
 488 sociated with. The value **MUST** be a single number ("1") or a comma separated set of
 489 individual elements ("1,2,3,4"), or as a inclusive range of values as in ("1-10") or any
 490 combination of ranges and numbers as in "1-4,6-10,22". There **MUST NOT** be spaces or
 491 non-integer values in the text representation.

492 Indices **SHOULD** start numbering with the inserts or CuttingItem furthest from the
 493 gauge line and increasing in value as the items get closer to the gauge line. Items at the
 494 same distance **MAY** be arbitrarily numbered.

495 **6.2.10.1.2 itemId Attribute for CuttingItem**

496 The manufactures' identifier for this CuttingItem that **MAY** be its catalog or reference
 497 number. The value **MUST** be an XML NMTOKEN value of numbers and letters.

498 **6.2.10.1.3 manufacturers Attribute for CuttingItem**

499 This optional element references the manufacturers of this tool. At this level the manufac-

500 turers will reference the CuttingItem specifically. The representation will be a comma
 501 (,) delimited list of manufacturer names. This can be any series of numbers and letters as
 502 defined by the XML type string.

503 **6.2.10.1.4 grade Attribute for CuttingItem**

504 This provides an implementation specific designation for the material composition of this
 505 CuttingItem.

506 **6.2.10.2 Elements for CuttingItem**

Table 23: Elements for CuttingItem

| Element | Description | Occurrence |
|--------------|--|------------|
| Description | A free-form description of the Cutting Item. | 0..1 |
| Locus | A free form description of the location on the Cutting Tool. | 0..1 |
| ItemLife | The life of this Cutting Item. | 0..3 |
| Measurements | A collection of measurements relating to this Cutting Item. | 0..1 |

507 **6.2.10.2.1 Description Element for CuttingItem**

508 An optional free form text description of this CuttingItem.

509 **6.2.10.2.2 Locus Element for CuttingItem**

510 Locus represents the location of the CuttingItem with respect to the Cutting Tool.
 511 For clarity, the words FLUTE, INSERT, and CARTRIDGE **SHOULD** be used to assist in
 512 noting the location of a CuttingItem. The Locus **MAY** be any free form text, but
 513 **SHOULD** adhere to the following rules:

- 514 • The location numbering **SHOULD** start at the furthest CuttingItem (#1) and
 515 work it's way back to the Cutting Item closest to the gauge line.
- 516 • Flutes **SHOULD** be identified as such using the word FLUTE:. For example: FLUTE:

517 1, INSERT: 2 - would indicate the first flute and the second furthest insert from the
 518 end of the tool on that flute.

- 519 • Other designations such as CARTRIDGE **MAY** be included, but should be identified
 520 using upper case and followed by a colon (:).

521 **6.2.10.2.3 ItemLife Element for CuttingItem**

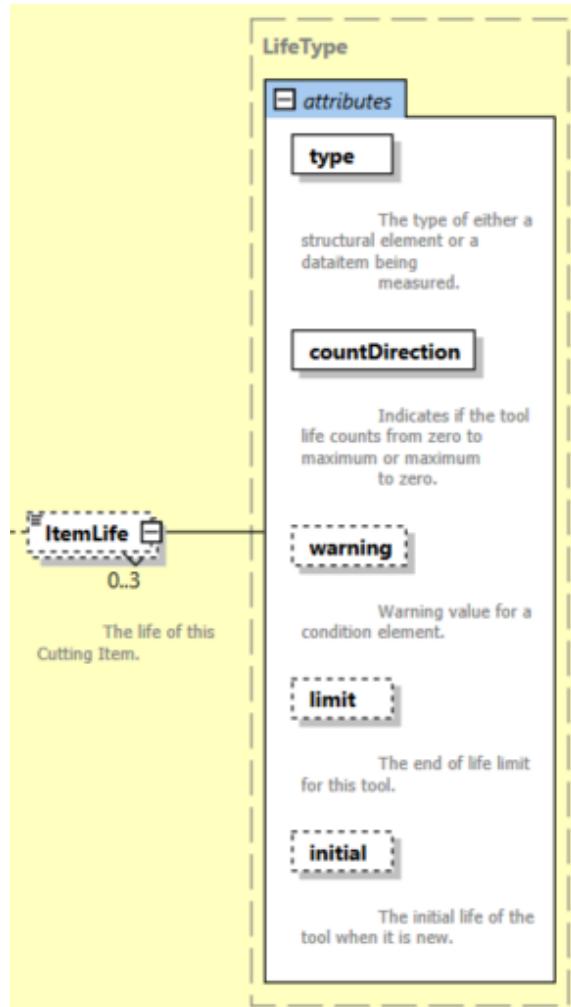


Figure 22: ItemLife Schema

522 The value is the current value for the ToolLife. The value **MUST** be a number. Tool-
 523 Life is an option element which can have three types, either minutes for time based, part
 524 count for parts based, or wear based using a distance measure. One tool life can appear for
 525 each type, but there cannot be two entries of the same type. Additional types can be added
 526 in the future.

527 **6.2.10.2.4 Attributes for ItemLife**

528 These is an optional attribute that can be used to further classify the operation type.

Table 24: Attributes for ItemLife

| Attribute | Description | Occurrence |
|----------------|--|------------|
| type | The type of tool life being accumulated. <i>Valid Data Values:</i> MINUTES, PART_COUNT, or WEAR. type is a required attribute. | 1 |
| countDirection | Indicates if the tool life counts from zero to maximum or maximum to zero. The value MUST be one of UP or DOWN. countDirection is a required attribute. | 1 |
| warning | The point at which a tool life warning will be raised. warning is an optional attribute. | 0..1 |
| limit | The end of life limit for this tool. If the countDirection is DOWN, the point at which this tool should be expired, usually zero. If the countDirection is UP, this is the upper limit for which this tool should be expired. limit is an optional attribute. | 0..1 |
| initial | The initial life of the tool when it is new. initial is an optional attribute. | 0..1 |

529 **6.2.10.2.5 type Attribute for ItemLife**

530 The value of type must be one of the following:

Table 25: Values for type of ItemLife

| Value | Description |
|------------|---|
| MINUTES | The tool life measured in minutes. All units for minimum, maximum, and nominal MUST be provided in minutes. |
| PART_COUNT | The tool life measured in parts. All units for minimum, maximum, and nominal MUST be provided as the number of parts. |
| WEAR | The tool life measured in tool wear. Wear MUST be provided in millimeters as an offset to nominal. All units for minimum, maximum, and nominal MUST be given as millimeter offsets as well. |

531 **6.2.10.2.6 countDirection Attribute for ItemLife**

532 The value of type must be one of the following:

Table 26: Values for countDirection

| Value | Description |
|-------|---|
| UP | The tool life counts up from zero to the maximum. |
| DOWN | The tool life counts down from the maximum to zero. |

533 **6.2.10.3 Measurement Subtypes for CuttingItem**534 These Measurements for CuttingItem are specific to an individual glscuttingitem
535 and **MUST NOT** be used for the Measurements pertaining to an assembly. The *Fig-*
536 *ure 23 , Figure 24 , Figure 25* and *Figure 26* will be used to for reference for the Cut-
537 *tingItem specific Measurements .*538 The Code in *Table 27* will refer to the acronym in the diagram. We will be referring to
539 many diagrams to disambiguate all Measurements of the CuttingTools and Cut-
540 tingItems. We will present a few here; please refer to Appendix B for additional
541 reference material.

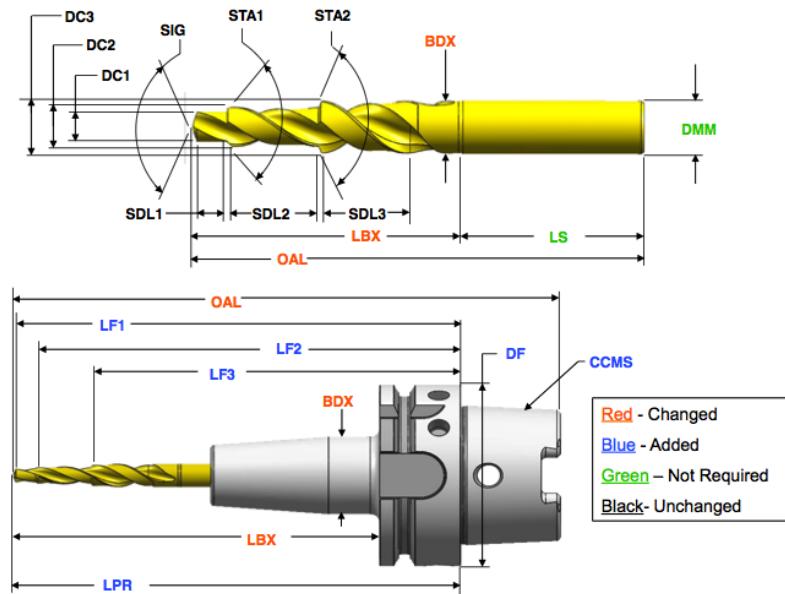


Figure 23: Cutting Tool

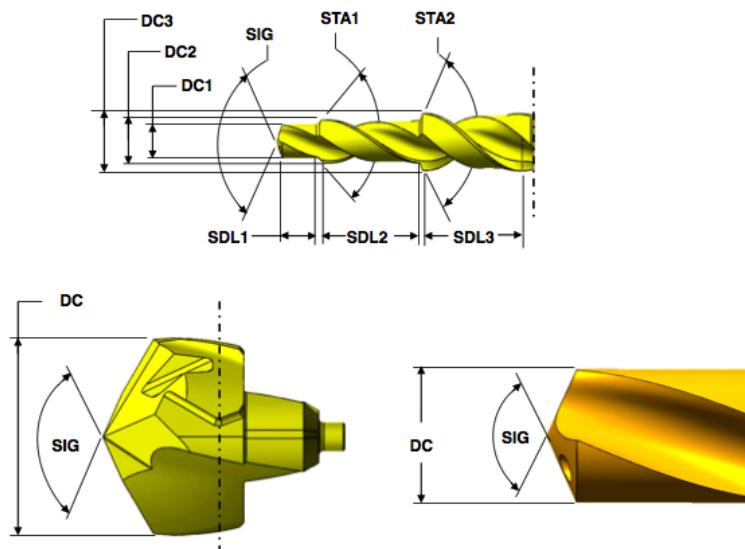
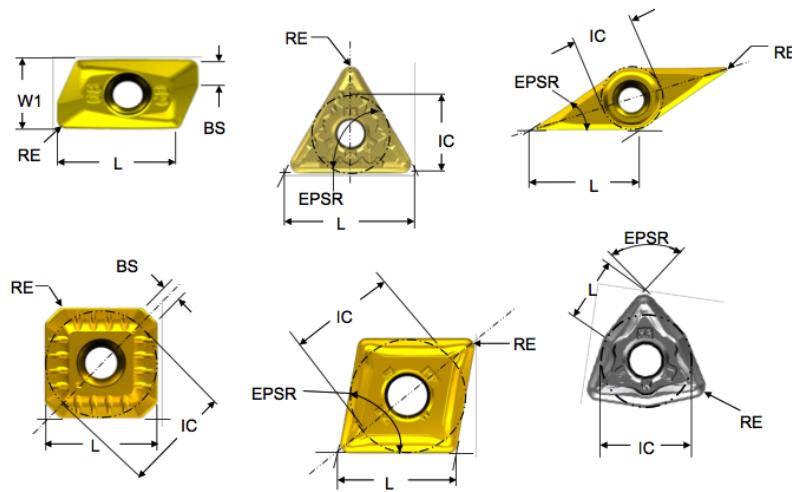
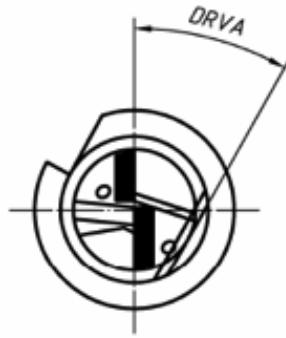


Figure 24: Cutting Item

**Figure 25:** Cutting Item Measurement Diagram 3**Figure 26:** Cutting Item Drive Angle

542 The Cutting Item Measurements in *Table 27* will refer the *Figure 23* , *Figure 24* ,
 543 *Figure 25* and *Figure 26* .

Table 27: Measurement Subtypes for CuttingItem

| Measurement Subtype | Code | Description | Units |
|-----------------------|------|---|------------|
| CuttingReferencePoint | CRP | The theoretical sharp point of the Cutting Tool from which the major functional dimensions are taken. | MILLIMETER |

| Continuation of Table 27 | | | |
|--------------------------|------|--|------------|
| Measurement Subtype | Code | Description | Units |
| CuttingEdgeLength | L | The theoretical length of the cutting edge of a Cutting Item over sharp corners. | MILLIMETER |
| DriveAngle | DRVA | Angle between the driving mechanism locator on a Tool Item and the main cutting edge. | DEGREE |
| FlangeDiameter | DF | The dimension between two parallel tangents on the outside edge of a flange. | MILLIMETER |
| FunctionalWidth | WF | The distance between the cutting reference point and the rear backing surface of a turning tool or the axis of a boring bar. | MILLIMETER |
| InscribedCircleDiameter | IC | The diameter of a circle to which all edges of a equilateral and round regular insert are tangential. | MILLIMETER |
| PointAngle | SIG | The angle between the major cutting edge and the same cutting edge rotated by 180 degrees about the tool axis. | DEGREE |
| ToolCuttingEdgeAngle | KAPR | The angle between the tool cutting edge plane and the tool feed plane measured in a plane parallel the xy-plane. | DEGREE |

| Continuation of Table 27 | | | |
|--------------------------|------|--|------------|
| Measurement Subtype | Code | Description | Units |
| ToolLeadAngle | PSIR | The angle between the tool cutting edge plane and a plane perpendicular to the tool feed plane measured in a plane parallel the xy-plane. | DEGREE |
| ToolOrientation | N/A | The angle of the tool with respect to the workpiece for a given process. The value is application specific. | DEGREE |
| WiperEdgeLength | BS | The measure of the length of a wiper edge of a Cutting Item. | MILLIMETER |
| StepDiameterLength | SDLx | The length of a portion of a stepped tool that is related to a corresponding cutting diameter measured from the cutting reference point of that cutting diameter to the point on the next cutting edge at which the diameter starts to change. | MILLIMETER |
| StepIncludedAngle | STAx | The angle between a major edge on a step of a stepped tool and the same cutting edge rotated 180 degrees about its tool axis. | DEGREE |

| Continuation of Table 27 | | | |
|--------------------------|------|--|------------|
| Measurement Subtype | Code | Description | Units |
| CuttingDiameter | DCx | The diameter of a circle on which the defined point Pk located on this Cutting Tool. The normal of the machined peripheral surface points towards the axis of the Cutting Tool. | MILLIMETER |
| CuttingHeight | HF | The distance from the basal plane of the Tool Item to the cutting point. | MILLIMETER |
| CornerRadius | RE | The nominal radius of a rounded corner measured in the X Y-plane. | MILLIMETER |
| Weight | WT | The total weight of the Cutting Tool in grams. The force exerted by the mass of the Cutting Tool. | GRAM |
| FunctionalLength | LFx | The distance from the gauge plane or from the end of the shank of the Cutting Tool, if a gauge plane does not exist, to the cutting reference point determined by the main function of the tool. This measurement will be with reference to the Cutting Tool and MUST NOT exist without a Cutting Tool. | MILLIMETER |
| ChamferFlatLength | BCH | The flat length of a chamfer. | MILLIMETER |
| ChamferWidth | CHW | The width of the chamfer. | MILLIMETER |

| Continuation of Table 27 | | | |
|--------------------------|------|---|------------|
| Measurement Subtype | Code | Description | Units |
| InsertWidth | W1 | W1 is used for the insert width when an inscribed circle diameter is not practical. | MILLIMETER |

544 Appendices

545 A Bibliography

- 546 Engineering Industries Association. *EIA Standard - EIA-274-D*, Interchangeable Variable,
547 Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically
548 Controlled Machines. Washington, D.C. 1979.
- 549 ISO TC 184/SC4/WG3 N1089. *ISO/DIS 10303-238*: Industrial automation systems and
550 integration Product data representation and exchange Part 238: Application Protocols: Ap-
551 plication interpreted model for computerized numerical controllers. Geneva, Switzerland,
552 2004.
- 553 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
554 tems and integration – Physical device control – Data model for computerized numerical
555 controllers – Part 10: General process data. Geneva, Switzerland, 2004.
- 556 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
557 tems and integration – Physical device control – Data model for computerized numerical
558 controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.
- 559 International Organization for Standardization. *ISO 6983/1* – Numerical Control of ma-
560 chines – Program format and definition of address words – Part 1: Data format for posi-
561 tioning, line and contouring control systems. Geneva, Switzerland, 1982.
- 562 Electronic Industries Association. *ANSI/EIA-494-B-1992*, 32 Bit Binary CL (BCL) and
563 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines.
564 Washington, D.C. 1992.
- 565 National Aerospace Standard. *Uniform Cutting Tests* - NAS Series: Metal Cutting Equip-
566 ment Specifications. Washington, D.C. 1969.
- 567 International Organization for Standardization. *ISO 10303-11*: 1994, Industrial automa-
568 tion systems and integration Product data representation and exchange Part 11: Descrip-
569 tion methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.
- 570 International Organization for Standardization. *ISO 10303-21*: 1996, Industrial automa-
571 tion systems and integration – Product data representation and exchange – Part 21: Imple-
572 mentation methods: Clear text encoding of the exchange structure. Geneva, Switzerland,
573 1996.
- 574 H.L. Horton, F.D. Jones, and E. Oberg. *Machinery's Handbook*. Industrial Press, Inc.

575 New York, 1984.

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

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

581 ASME/ANSI B5.54: *Methods for Performance Evaluation of Computer Numerically Con-*
582 *trolled Machining Centers.* 2005.

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

585 International Organization for Standardization. *ISO 13399: Cutting tool data representa-*
586 *tion and exchange.* Geneva, Switzerland, 2000.

587 B Additional Illustrations

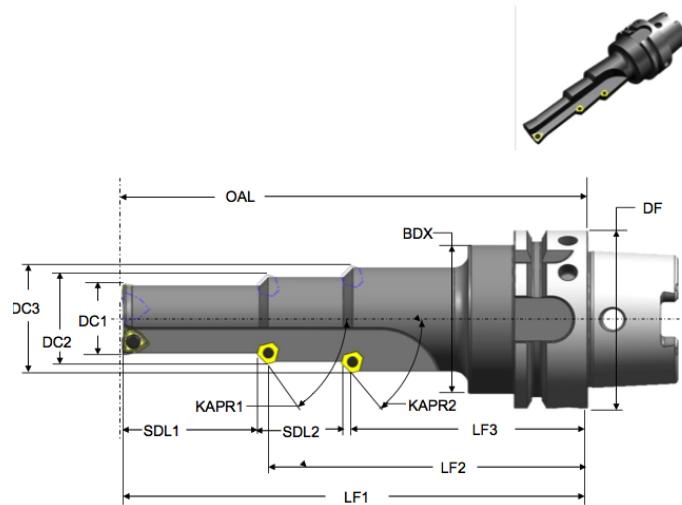


Figure 27: Cutting Tool Measurement Diagram 1
(Cutting Tool, Cutting Item, and Assembly Item – ISO 13399)

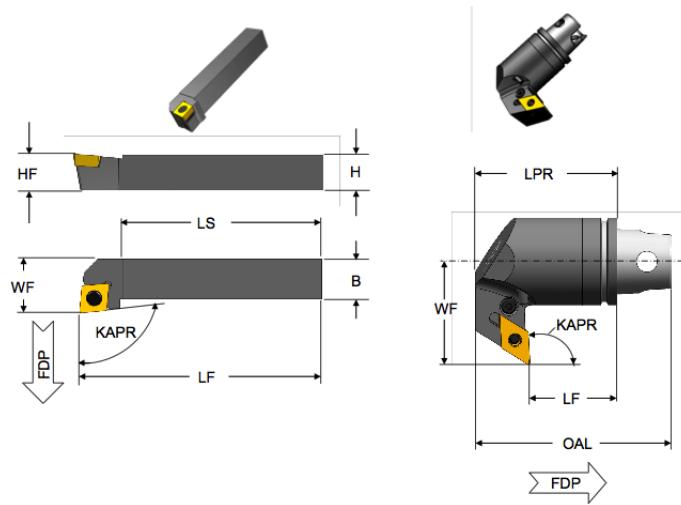


Figure 28: Cutting Tool Measurement Diagram 2
(Cutting Tool, Cutting Item, and Assembly Item – ISO 13399)

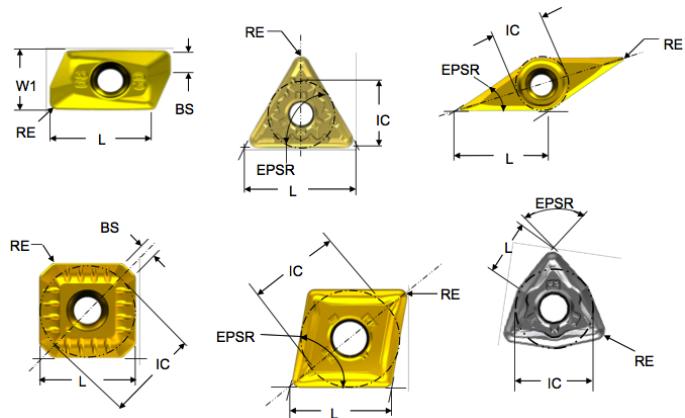


Figure 29: Cutting Tool Measurement Diagram 3
(Cutting Item – ISO 13399)

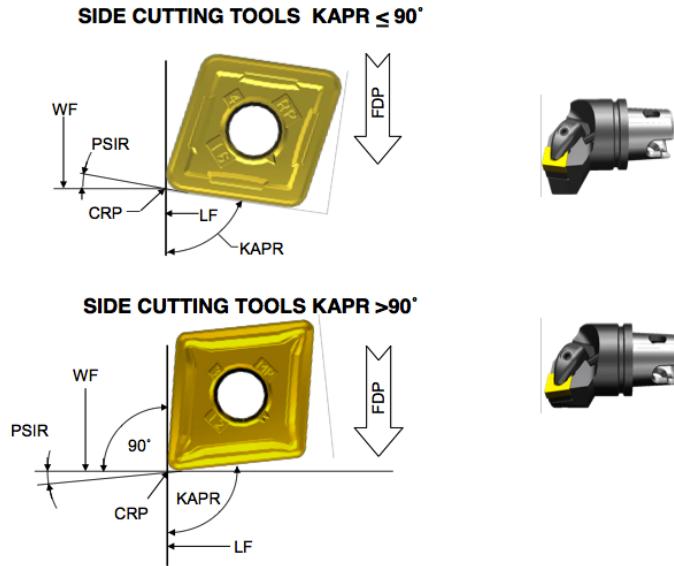


Figure 30: Cutting Tool Measurement Diagram 4
(Cutting Item – ISO 13399)

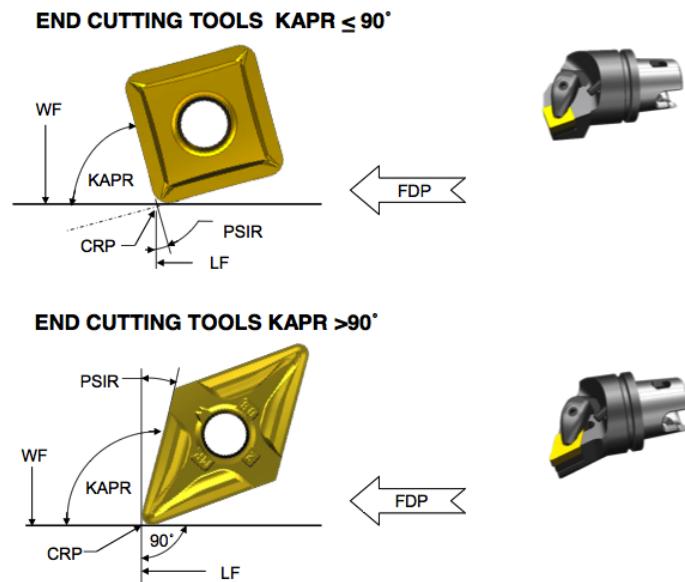


Figure 31: Cutting Tool Measurement Diagram 5
(Cutting Item – ISO 13399)

BCH = CHAMFER FLAT LENGTH
CHW = CHAMFER WIDTH

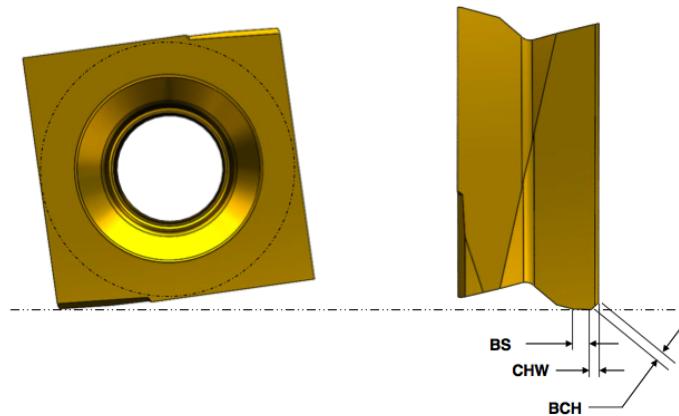


Figure 32: Cutting Tool Measurement Diagram 6
(Cutting Item – ISO 13399)

588 C Cutting Tool Example

589 C.1 Shell Mill

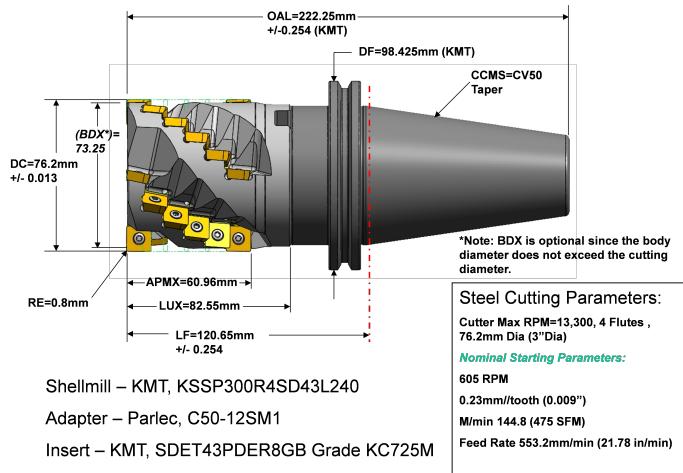


Figure 33: Shell Mill Side View

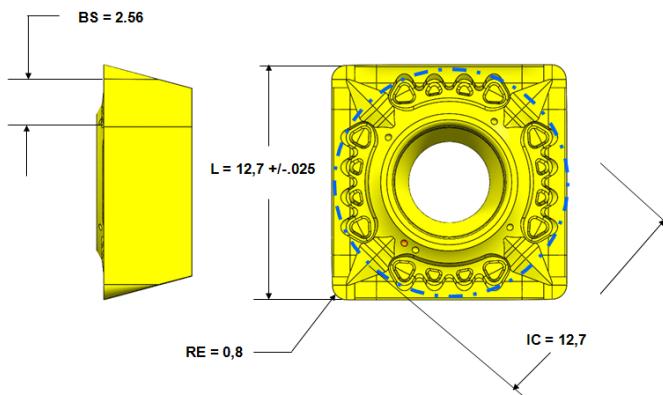


Figure 34: Indexable Insert Measurements

Example 1: Example for Indexable Insert Measurements

```

590 1 <?xml version="1.0" encoding="UTF-8"?>
591 2 <MTConnectAssets
592 3 xmlns:m="urn:mtconnect.org:MTConnectAssets:1.2"
593 4 xmlns="urn:mtconnect.org:MTConnectAssets:1.2"
594 5 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
595 6 xsi:schemaLocation="urn:mtconnect.org:MTConnectAssets:1.2
596 7 http://mtconnect.org/schemas/MTConnectAssets\_1.2.xsd">
597 8   <Header creationTime="2011-05-11T13:55:22"
598 9     assetBufferSize="1024" sender="localhost"
```

```

599 10    assetCount="2" version="1.2" instanceId="1234"/>
600 11    <Assets>
601 12      <CuttingTool serialNumber="1" toolId="KSSP300R4SD43L240">
602 13        timestamp="2011-05-11T13:55:22" assetId="KSSP300R4SD43L240.1"
603 14        manufacturers="KMT, Parlec">
604 15          <CuttingToolLifeCycle>
605 16            <CutterStatus><Status>NEW</Status></CutterStatus>
606 17            <ProcessSpindleSpeed maximum="13300"
607 18              nominal="605">10000</ProcessSpindleSpeed>
608 19            <ProcessFeedRate
609 20              nominal="9.22">9.22</ProcessSpindleSpeed>
610 21            <ConnectionCodeMachineSide>CV50
611 22            </ConnectionCodeMachineSide>
612 23            <Measurements>
613 24              <BodyDiameterMax code="BDX">73.25
614 25              </BodyDiameterMax>
615 26              <OverallToolLength nominal="222.25"
616 27                minimum="221.996" maximum="222.504"
617 28                  code="OAL">222.25</OverallToolLength>
618 29              <UsableLengthMax code="LUX" nominal="82.55">82.55
619 30              </UsableLengthMax>
620 31              <CuttingDiameterMax code="DC" nominal="76.2"
621 32                maximum="76.213" minimum="76.187">76.2
622 33              </CuttingDiameterMax>
623 34              <BodyLengthMax code="LF" nominal="120.65"
624 35                maximum="120.904" minimum="120.404">120.65
625 36              </BodyLengthMax>
626 37              <DepthOfCutMax code="APMX"
627 38                nominal="60.96">60.95</DepthOfCutMax>
628 39              <FlangeDiameterMax code="DF"
629 40                nominal="98.425">98.425</FlangeDiameterMax>
630 41            </Measurements>
631 42            <CuttingItems count="24">
632 43              <CuttingItem indices="1-24" itemId="SDET43PDER8GB"
633 44                manufacturers="KMT" grade="KC725M">
634 45                <Measurements>
635 46                  <CuttingEdgeLength code="L" nominal="12.7"
636 47                    minimum="12.675" maximum="12.725">12.7
637 48                  </CuttingEdgeLength>
638 49                  <WiperEdgeLength code="BS" nominal=
639 50                    "2.56">2.56</WiperEdgeLength>
640 51                  <InscribedCircleDiameter code="IC"
641 52                    nominal="12.7">12.7
642 53                  </InscribedCircleDiameter>
643 54                  <CornerRadius code="RE" nominal="0.8">
644 55                    0.8</CornerRadius>
645 56                  </Measurements>
646 57                  </CuttingItem>
647 58            </CuttingItems>
648 59            </CuttingToolLifeCycle>
649 60          </CuttingTool>

```

```
650 61    </Assets>
651 62  </MTConnectAssets>
```

652 C.2 Step Drill

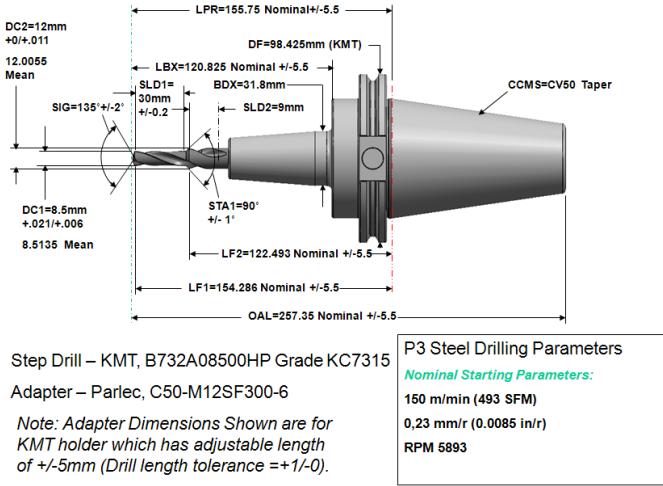


Figure 35: Step Mill Side View

Example 2: Example for Step Mill Side View

```

653 1 <?xml version="1.0" encoding="UTF-8"?>
654 2 <MTConnectAssets xmlns:m="urn:mtconnect.org:MTConnectAssets:1.2"
655 3 xmlns="urn:mtconnect.org:MTConnectAssets:1.2"
656 4 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
657 5 xsi:schemaLocation="urn:mtconnect.org:MTConnectAssets:1.2
658 6 http://mtconnect.org/schemas/MTConnectAssets\1.2.xsd">
659 7   <Header creationTime="2011-05-
660 8     _11T13:55:22" assetBufferSize="1024"
661 9     sender="localhost" assetCount="2" version="1.2" instanceId="1234"/>
662 10  <Assets>
663 11    <CuttingTool serialNumber="1" toolId="B732A08500HP"
664 12      timestamp="2011-05-11T13:55:22" assetId="B732A08500HP_"
665 13      manufacturers="KMT, Parlec">
666 14      <Description>
667 15        Step Drill - KMT, B732A08500HP Grade KC7315
668 16        Adapter - Parlec, C50-M12SF300-6
669 17      </Description>
670 18      <CuttingToolLifeCycle>
671 19        <CutterStatus><Status>NEW</Status></CutterStatus>
672 20        <ProcessSpindleSpeed nominal="5893">5893</ProcessSpindleSpeed>
673 21        <ProcessFeedRate nominal="2.5">2.5</ProcessFeedRate>
674 22        <ConnectionCodeMachineSide>CV50 Taper</ConnectionCodeMachineSide>
675 23      <Measurements>
676 24        <BodyDiameterMax code="BDX">31.8</BodyDiameterMax>
677 25        <BodyLengthMax code="LBX" nominal="120.825" maximum="126.325"
678 26          minimum="115.325">120.825</BodyLengthMax>
679 27        <ProtrudingLength code="LPR" nominal="155.75" maximum="161.25"
680 28          minimum="150.26">155.75</ProtrudingLength>

```

```

681 29      <FlangeDiameterMax code="DF"
682 30          nominal="98.425">98.425</FlangeDiameterMax>
683 31      <OverallToolLength nominal="257.35" minimum="251.85"
684 32          maximum="262.85" code="OAL">257.35</OverallToolLength>
685 33      </Measurements>
686 34      <CuttingItems count="2">
687 35          <CuttingItem indices="1" manufacturers="KMT" grade="KC7315">>
688 36              <Measurements>
689 37                  <CuttingDiameter code="DC1" nominal="8.5" maximum="8.521"
690 38                      minimum="8.506">8.5135</CuttingDiameter>
691 39                  <StepIncludedAngle code="STA1" nominal="90" maximum="91"
692 40                      minimum="89">90</StepIncludedAngle>
693 41                  <FunctionalLength code="LF1" nominal="154.286"
694 42                      minimum="148.786"
695 43                      maximum="159.786">154.286</FunctionalLength>
696 44                  <StepDiameterLength code="SDL1"
697 45                      nominal="9">9</StepDiameterLength>
698 46                  <PointAngle code="SIG" nominal="135" minimum="133"
699 47                      maximum="137">135</PointAngle>
700 48              </Measurements>
701 49          </CuttingItem>
702 50          <CuttingItem indices="2" manufacturers="KMT" grade="KC7315">>
703 51              <Measurements>
704 52                  <CuttingDiameter code="DC2" nominal="12" maximum="12.011"
705 53                      minimum="12">12</CuttingDiameter>
706 54                  <FunctionalLength code="LF2" nominal="122.493"
707 55                      maximum="127.993"
708 56                      minimum="116.993">122.493</FunctionalLength>
709 57                  <StepDiameterLength code="SDL2"
710 58                      nominal="9">9</StepDiameterLength>
711 59              </Measurements>
712 60          </CuttingItem>
713 61      </CuttingItems>
714 62      </CuttingToolLifeCycle>
715 63  </CuttingTool>
716 64  </Assets>
717 65 </MTConnectAssets>

```

718 C.3 Shell Mill with Individual Loci

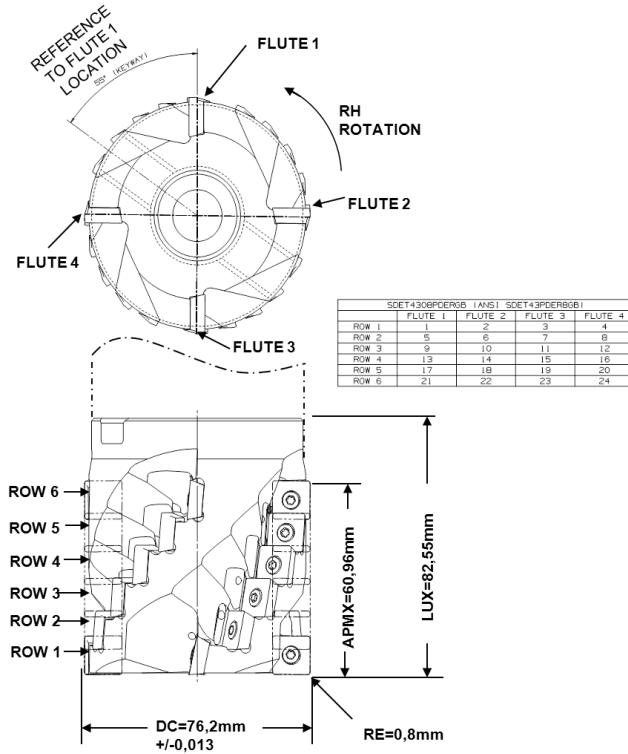


Figure 36: Shell Mill with Explicate Loci

Example 3: Example for Shell Mill with Explicate Loci

```

719 1 <?xml version="1.0" encoding="UTF-8"?>
720 2 <MTConnectAssets xmlns:m="urn:mtconnect.org:MTConnectAssets:1.2"
721 3 xmlns="urn:mtconnect.org:MTConnectAssets:1.2"
722 4 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
723 5 xsi:schemaLocation="urn:mtconnect.org:MTConnectAssets:1.2
724 6 http://mtconnect.org/schemas/MTConnectAssets\_1.2.xsd">
725 7 <Header creationTime="2011-05-11T13:55:22" assetBufferSize="1024"
726 8 sender="localhost" assetCount="2" version="1.2" instanceId="1234"/>
727 9 <Assets>
728 10 <CuttingTool serialNumber="1" toolID="KSSP300R4SD43L240"
729 11 timestamp="2011-05-11T13:55:22" assetId="KSSP300R4SD43L240.1"
730 12 manufacturers="KMT, Parlec">
731 13 <Description>Keyway: 55 degrees</Description>
732 14 <CuttingToolLifeCycle>
733 15 <CutterStatus><Status>NEW</Status></CutterStatus>
734 16 <Measurements>
735 17 <UsableLengthMax code="LUX"
736 18 nominal="82.55">82.55</UsableLengthMax>
737 19 <CuttingDiameterMax code="DC" nominal="76.2" maximum="76.213"
```

```
738 20      minimum="76.187">76.2</CuttingDiameterMax>
739 21      <DepthOfCutMax code="APMX" nominal="60.96">60.95</DepthOfCutMax>
740 22  </Measurements>
741 23  <CuttingItems count="24">
742 24      <CuttingItem indices="1" itemId="SDET43PDER8GB"
743 25      manufacturers="KMT">
744 26          <Locus>FLUTE: 1, ROW: 1</Locus>
745 27          <Measurements>
746 28              <DriveAngle code="DRVA" nominal="55">55</DriveAngle>
747 29          </Measurements>
748 30      </CuttingItem>
749 31      <CuttingItem indices="2-24" itemId="SDET43PDER8GB"
750 32      manufacturers="KMT">
751 33          <Locus>FLUTE: 2-4, ROW: 1; FLUTE: 1-4, ROW 2-6</Locus>
752 34          </CuttingItem>
753 35      </CuttingItems>
754 36      </CuttingToolLifeCycle>
755 37  </CuttingTool>
756 38  </Assets>
757 39 </MTConnectAssets>
```

758 C.4 Drill with Individual Loci

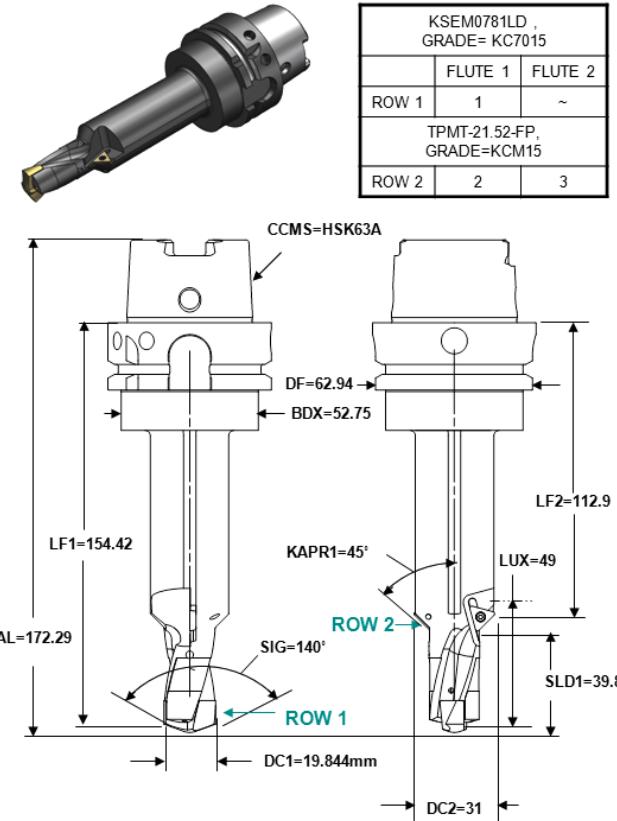


Figure 37: Step Drill with Explicate Loci

Example 4: Example for Step Drill with Explicate Loci

```

759 1 <?xml version="1.0" encoding="UTF-8"?>
760 2 <MTConnectAssets xmlns:m="urn:mtconnect.org:MTConnectAssets:1.2"
761 3   xmlns="urn:mtconnect.org:MTConnectAssets:1.2"
762 4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
763 5   xsi:schemaLocation="urn:mtconnect.org:MTConnectAssets:1.2
764 6   http://mtconnect.org/schemas/MTConnectAssets\1.2.xsd">
765 7   <Header creationTime="2011-05-11T13:55:22" assetBufferSize="1024"
766 8     sender="localhost" assetCount="2" version="1.2" instanceId="1234"/>
767 9   <Assets>
768 10    <CuttingTool serialNumber="1" toolId="KSEM0781LD"
769 11      timestamp="2011-05-11T13:55:22" assetId="KSEM0781LD.1" manufacturers="KMT">
770 12      <CuttingToolLifeCycle>
771 13        <CutterStatus><Status>NEW</Status></CutterStatus>
772 14        <ConnectionCodeMachineSide>HSK63A</ConnectionCodeMachineSide>
773 15        <Measurements>
774 16          <BodyDiameterMax code="BDX">52.75</BodyDiameterMax>
775 17          <OverallToolLength nominal="172.29"

```

```

776 18      code="OAL">172.29</OverallToolLength>
777 19      <UsableLengthMax code="LUX" nominal="49">49</UsableLengthMax>
778 20      <FlangeDiameterMax code="DF"
779 21          nominal="62.94">62.94</FlangeDiameterMax>
780 22      </Measurements>
781 23      <CuttingItems count="3">
782 24          <CuttingItem indices="1" itemId="KSEM0781LD" manufacturers="KMT"
783 25              grade="KC7015">
784 26              <Locus>FLUTE: 1, ROW: 1</Locus>
785 27              <Measurements>
786 28                  <FunctionalLength code="LF1" nominal="154.42">154.42</FunctionalLength>
787 29                  <CuttingDiameter code="DC1" nominal="19.844">19.844</CuttingDiameter>
788 30                  <PointAngle code="SIG" nominal="140">140</PointAngle>
789 31                  <ToolCuttingEdgeAngle code="KAPR1" nominal="45">45</ToolCuttingEdgeAngle>
790 32                  <StepDiameterLength code="SLD1" nominal="39.8">39.8</StepDiameterLength>
791 33          </Measurements>
792 34      </CuttingItem>
793 35          <CuttingItem indices="2-3" itemId="TPMT-21.52-FP"
794 36              manufacturers="KMT" grade="KCM15">
795 37              <Locus>FLUTE: 1-2, ROW: 2</Locus>
796 38              <Measurements>
797 39                  <FunctionalLength code="LF2" nominal="112.9">119.2</FunctionalLength>
798 40                  <CuttingDiameter code="DC2" nominal="31">31</CuttingDiameter>
799 41          </Measurements>
800 42      </CuttingItem>
801 43      </CuttingItems>
802 44  </CuttingToolLifeCycle>
803 45  </CuttingTool>
804 46  </Assets>
805 47  </MTConnectAssets>

```

806 C.5 Shell Mill with Different Inserts on First Row

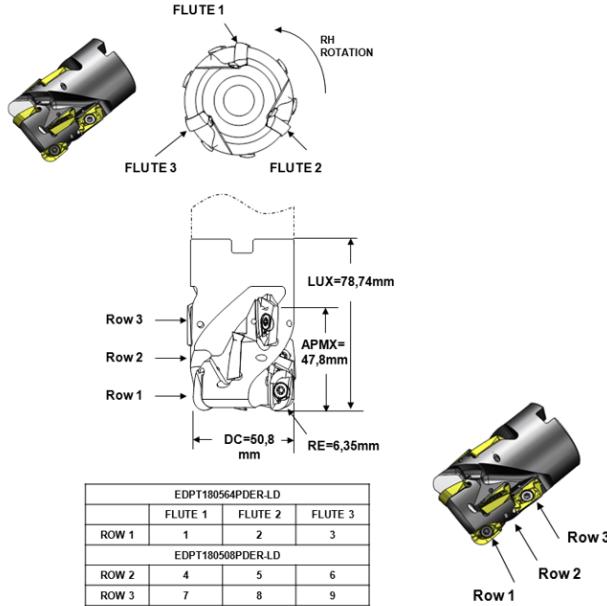


Figure 38: Shell Mill with Different Inserts on First Row

Example 5: Example for Shell Mill with Different Inserts on First Row

```

807 1 <?xml version="1.0" encoding="UTF-8"?>
808 2 <MTConnectAssets xmlns:m="urn:mtconnect.org:MTConnectAssets:1.2"
809 3   xmlns="urn:mtconnect.org:MTConnectAssets:1.2"
810 4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
811 5   xsi:schemaLocation="urn:mtconnect.org:MTConnectAssets:1.2
812 6   http://mtconnect.org/schemas/MTConnectAssets\1.2.xsd">
813 7     <Header creationTime="2011-05-11T13:55:22" assetBufferSize="1024"
814 8       sender="localhost" assetCount="2" version="1.2" instanceId="1234"/>
815 9     <Assets>
816 10       <CuttingTool serialNumber="1" toolId="XXX" timestamp="2011-05-11T13:55:22"
817 11         assetId="XXX.1" manufacturers="KMT">
818 12           <CuttingToolLifeCycle>
819 13             <CutterStatus><Status>NEW</Status></CutterStatus>
820 14             <Measurements>
821 15               <DepthOfCutMax code="APMX" nominal="47.8">47.8</DepthOfCutMax>
822 16               <CuttingDiameterMax code="DC"
823 17                 nominal="50.8">50.8</CuttingDiameterMax>
824 18               <UsableLengthMax code="LUX"
825 19                 nominal="78.74">78.74</UsableLengthMax>
826 20             </Measurements>
827 21             <CuttingItems count="9">
828 22               <CuttingItem indices="1-3" itemId="EDPT180564PDER-LD"
829 23                 manufacturers="KMT">
830 24                 <Locus>FLUTE: 1-3, ROW: 1</Locus>

```

```
831 25      <Measurements>
832 26          <CornerRadius code="RE" nominal="6.25">6.35</CornerRadius>
833 27      </Measurements>
834 28  </CuttingItem>
835 29      <CuttingItem indices="4-9" itemId="EDPT180508PDER-LD"
836 30          manufacturers="KMT">
837 31          <Locus>FLANGE: 1-4, ROW: 2-3</Locus>
838 32      </CuttingItem>
839 33      </CuttingItems>
840 34  </CuttingToolLifeCycle>
841 35      </CuttingTool>
842 36  </Assets>
843 37 </MTConnectAssets>
```



MTConnect® Standard

Part 5 – Interfaces

Version 1.5.0

Prepared for: MTConnect Institute
Prepared on: December 2, 2019

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/>.

MTConnect Specification and Materials

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 | 7 |
| 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

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

5 This data model is based on an *Interaction Model* that defines the exchange of information
6 between pieces of equipment and is organized in the MTConnect Standard as the XML
7 element `Interfaces`.

8 *Interfaces* is modeled as an extension to the `MTConnectDevices` and `MTConnect-`
9 `Streams` XML documents. *Interfaces* leverages similar rules and terminology as
10 those used to describe a component in the `MTConnectDevices` XML document. *In-*
11 *terfaces* also uses similar methods for reporting data to those used in the `MTCon-`
12 `nnectStreams` XML document.

13 As defined in *MTConnect Standard: Part 2.0 - Devices Information Model*, *Interfaces*
14 is modeled as a *Top Level* component in the `MTConnectDevices` document (see *Fig-*
15 *ure 3*). Each individual *Interface* XML element is modeled as a *Lower Level* com-
16 ponent of *Interfaces*. The data associated with each *Interface* is modeled within each
17 *Lower Level* component.

18 Note: See *MTConnect Standard: Part 2.0 - Devices Information Model* and *MT-*
19 *Connect Standard: Part 3.0 - Streams Information Model* of the MTConnect
20 Standard for information on how *Interfaces* is structured in the XML doc-
21 uments which are returned from an *Agent* in response to a *probe*, *sample*, or
22 *current request*.

23 2 Terminology and Conventions

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

27 2.1 Glossary

28 CDATA

29 General meaning:

30 An abbreviation for Character Data.

31 CDATA is used to describe a value (text or data) published as part of an XML ele-
 32 ment.

33 For example, "This is some text" is the CDATA in the XML element:

34 <Message ...>This is some text</Message>

35 Appears in the documents in the following form: CDATA

36 Agent

37 Refers to an MTConnect Agent.

38 Software that collects data published from one or more piece(s) of equipment, orga-
 39 nizes that data in a structured manner, and responds to requests for data from client
 40 software systems by providing a structured response in the form of a *Response Doc-
 41 ument* that is constructed using the *semantic data models* defined in the Standard.

42 Appears in the documents in the following form: *Agent*.

43 Asset Document

44 An electronic document published by an *Agent* in response to a *Request* for infor-
 45 mation from a client software application relating to Assets.

46 Child Element

47 A portion of a data modeling structure that illustrates the relationship between an
 48 element and the higher-level *Parent Element* within which it is contained.

49 Appears in the documents in the following form: *Child Element*.

50 ***Controlled Vocabulary***

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

53 Appears in the documents in the following form: *Controlled Vocabulary*.

54 ***Data Entity***

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

58 Appears in the documents in the following form: *Data Entity*.

59 ***Devices Information Model***

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

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

63 ***Document***

64 General meaning:

65 A piece of written, printed, or electronic matter that provides information.

66 Used to represent an *MTConnect Document*:

67 Refers to printed or electronic document(s) that represent a *Part(s)* of the MTCon-
68 nect Standard.

69 Appears in the documents in the following form: *MTConnect Document*.

70 Used to represent a specific representation of an *MTConnect Document*:

71 Refers to electronic document(s) associated with an *Agent* that are encoded using
72 XML; *Response Documents* or *Asset Documents*.

73 Appears in the documents in the following form: *MTConnect XML Document*.

74 Used to describe types of information stored in an *Agent*:

75 In an implementation, the electronic documents that are published from a data source
76 and stored by an *Agent*.

77 Appears in the documents in the following form: *Asset Document*.

78 Used to describe information published by an *Agent*:

79 A document published by an *Agent* based upon one of the *semantic data models*
80 defined in the MTConnect Standard in response to a request from a client.

81 Appears in the documents in the following form: *Response Document*.

82 ***Element Name***

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

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

86 Used to describe the name for a specific XML element:

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

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

90 ***Equipment Metadata***

91 See *Metadata*

92 ***Information Model***

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

95 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
96 documents and the relationship between pieces of information.

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

99 ***Interaction Model***

100 The definition of information exchanged to support the interactions between pieces
101 of equipment collaborating to complete a task.

102 Appears in the documents in the following form: *Interaction Model*.

103 ***Interface***

104 General meaning:

105 The exchange of information between pieces of equipment and/or software systems.

106 Appears in the documents in the following form: interface.

107 Used as an *Interaction Model*:

108 An *Interaction Model* that describes a method for inter-operations between pieces
109 of equipment.

110 Appears in the documents in the following form: *Interface*.

111 Used as an XML container or element:

112 - When used as an XML container that consists of one or more types of *Interface*
113 XML elements.

114 Appears in the documents in the following form: *Interfaces*.

115 - When used as an abstract XML element. It is replaced in the XML document
116 by types of `Interface` elements.

117 Appears in the documents in the following form: `Interface`

118 ***Lower Level***

119 A nested element that is below a higher level element.

120 ***Metadata***

121 Data that provides information about other data.

122 For example, *Equipment Metadata* defines both the *Structural Elements* that rep-
123 resent the physical and logical parts and sub-parts of each piece of equipment, the
124 relationships between those parts and sub-parts, and the definitions of the *Data En-*
125 *tities* associated with that piece of equipment.

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

127 ***MTConnect Document***

128 See *Document*.

129 ***MTConnect XML Document***

130 See *Document*.

131 ***Parent Element***

132 An XML element used to organize *Lower Level* child elements that share a common
133 relationship to the *Parent Element*.

134 Appears in the documents in the following form: *Parent Element*.

135 ***Publish/Subscribe***

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

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

142 ***Request***

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

145 Appears in the documents in the following form: *Request*.

146 ***Requester***

147 An entity that initiates a *Request* for information in a communications exchange.
148 Appears in the documents in the following form: *Requester*.

149 ***Responder***

150 An entity that responds to a *Request* for information in a communications exchange.
151 Appears in the documents in the following form: *Responder*.

152 ***Response Document***

153 See *Document*.

154 ***semantic data model***

155 A methodology for defining the structure and meaning for data in a specific logical
156 way.
157 It provides the rules for encoding electronic information such that it can be inter-
158 preted by a software system.
159 Appears in the documents in the following form: *semantic data model*.

160 ***Streaming Data***

161 The values published by a piece of equipment for the *Data Entities* defined by the
162 *Equipment Metadata*.
163 Appears in the documents in the following form: *Streaming Data*.

164 ***Structural Element***

165 General meaning:
166 An XML element that organizes information that represents the physical and logical
167 parts and sub-parts of a piece of equipment.
168 Appears in the documents in the following form: *Structural Element*.
169 Used to indicate hierarchy of Components:
170 When used to describe a primary physical or logical construct within a piece of
171 equipment.
172 Appears in the documents in the following form: *Top Level Structural Element*.
173 When used to indicate a *Child Element* which provides additional detail describing
174 the physical or logical structure of a *Top Level Structural Element*.
175 Appears in the documents in the following form: *Lower Level Structural Element*.

176 *Top Level*

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

179 ***Valid Data Value***

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

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

183 2.2 Acronyms

184 *AMT*

185 The Association for Manufacturing Technology

186 2.3 MTConnect References

187 [MTConnect Part 1.0] *MTConnect Standard Part 1.0 - Overview and Fundamentals*. Version 1.5.0.
188

189 [MTConnect Part 2.0] *MTConnect Standard: Part 2.0 - Devices Information Model*. Ver-
190 sion 1.5.0.

[MTConnect Part 3.0] *MTConnect Standard: Part 3.0 - Streams Information Model*. Version 1.5.0.

193 [MTConnect Part 5.0] *MTConnect Standard: Part 5.0 - Interfaces*. Version 1.5.0.

194 3 Interfaces Overview

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

200 In the MTConnect Standard, *Interfaces* provides a means to replace this traditional method
 201 for interconnecting pieces of equipment with a structured *Interaction Model* that provides
 202 a rich set of information used to coordinate the actions between pieces of equipment. Im-
 203 plementers may utilize the information provided by this data model to (1) realize the inter-
 204 action between pieces of equipment and (2) to extend the functionality of the equipment
 205 to improve the overall performance of the manufacturing process.

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

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

- 215 • An *Agent* which provides:
 - 216 - The data required to implement the *Interaction Model*.
 - 217 - Any other data from a piece of equipment needed to implement the *Interface*
 - 218 – operating states of the equipment, position information, execution modes, process
 - 219 information, etc.
- 220 • A client software application that enables the piece of equipment to acquire and
- 221 interpret information from another piece of equipment.

222 3.1 Interfaces Architecture

223 MTConnect Standard is based on a communications method that provides no direct way
 224 for one piece of equipment to change the state of or cause an action to occur in another

225 piece of equipment. The *Interaction Model* used to implement *Interfaces* is based on a
 226 *Publish/Subscribe* type of communications as described in *MTConnect Standard Part 1.0*
 227 - *Overview and Fundamentals* and utilizes a *Request* and *Response* information exchange
 228 mechanism. For *Interfaces*, pieces of equipment must perform both the publish (*Agent*)
 229 and subscribe (client) functions.

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

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

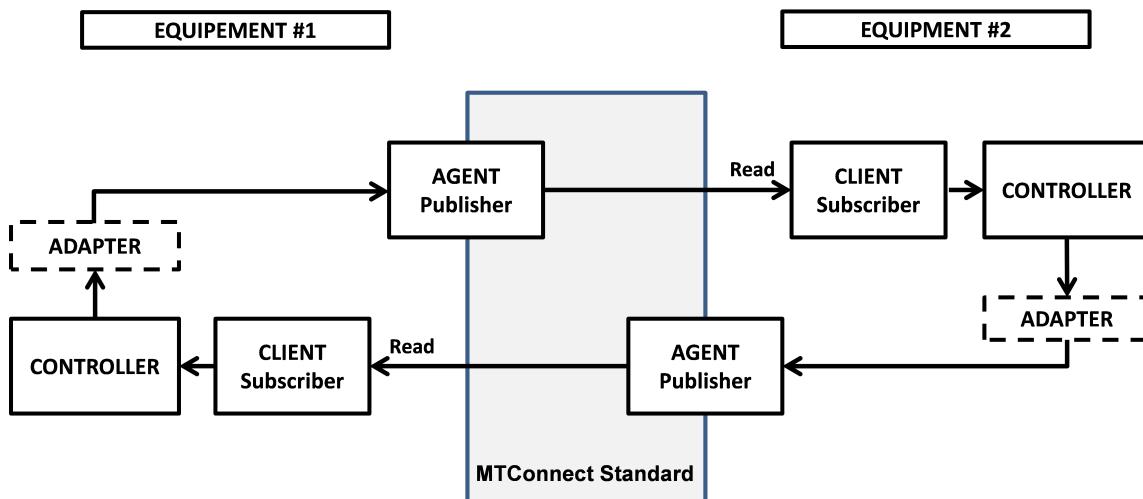


Figure 1: Data Flow Architecture for Interfaces

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

237 In the implementation of the *Interaction Model* for *Interfaces*, two pieces of equipment
 238 can exchange information in the following manner. One piece of equipment indicates a
 239 *Request* for service by publishing a type of *Request* using a data item provided through an
 240 *Agent* as defined in *Section 4 - Interfaces for Devices and Streams Information Models*.
 241 The client associated with the second piece of equipment, which is subscribing to data
 242 from the first machine, detects and interprets that *Request*. If the second machine chooses
 243 to take any action to fulfill this *Request*, it can indicate its acceptance by publishing a
 244 *Response* using a data item provided through its *Agent*. The client on the first piece of
 245 equipment continues to monitor information from the second piece of equipment until it
 246 detects an indication that the *Response* to the *Request* has been completed or has failed.

247 An example of this type of interaction between pieces of equipment can be represented

248 by a machine tool that wants the material to be loaded by a robot. In this example, the
 249 machine tool is the *Requester*, and the robot is the *Responder*. On the other hand, if the
 250 robot wants the machine tool to open a door, the robot becomes the *Requester* and the
 251 machine tool the *Responder*.

252 3.2 Request and Response Information Exchange

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

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

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

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

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

281 Note: The implementation details of how the *Responder* piece of equipment reacts to
 282 the *Request* and then completes the requested task are up to the implementer.

283 *Figure 2* provides an example of the *Request* and *Response* state machine:

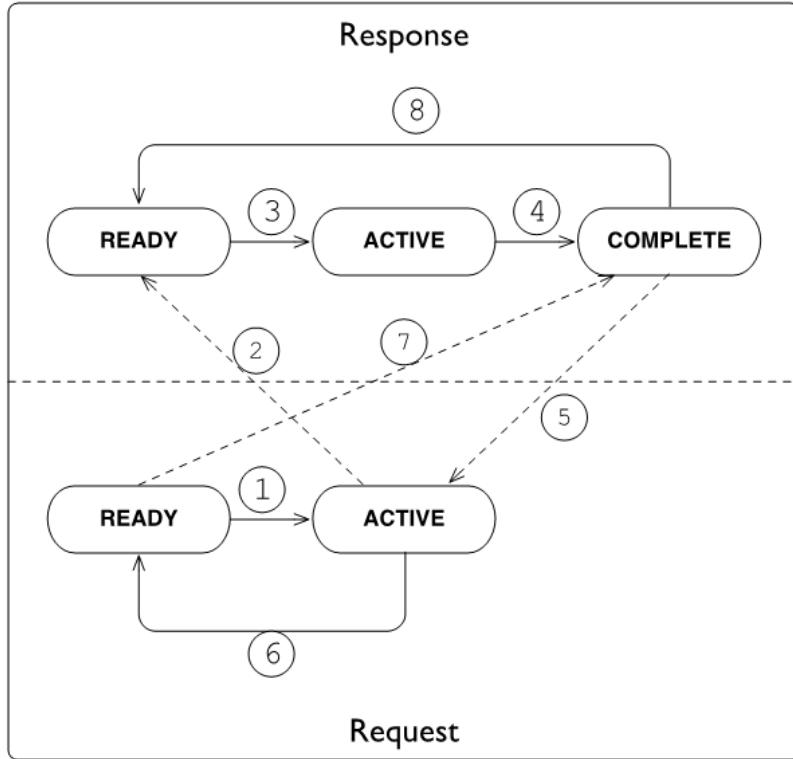


Figure 2: Request and Response Overview

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

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

Table 1: Sequence of interaction between pieces of equipment

| Step | Description |
|------|---|
| 1 | The <i>Request</i> transitions from READY to ACTIVE signaling that a service is needed. |
| 2 | The <i>Response</i> detects the transition of the <i>Request</i> . |
| 3 | The <i>Response</i> transitions from READY to ACTIVE indicating that it is performing the action. |
| 4 | Once the action has been performed, the <i>Response</i> transitions to COMPLETE. |

| Continuation of Table 1 | |
|-------------------------|---|
| Step | Description |
| 5 | The <i>Request</i> detects the action is COMPLETE. |
| 6 | The <i>Request</i> transitions back to READY acknowledging that the service has been performed. |
| 7 | The <i>Response</i> detects the <i>Request</i> has returned to READY. |
| 8 | In recognition of this acknowledgement, the <i>Response</i> transitions back to READY. |

289 After the final action has been completed, both pieces of equipment are back in the READY
 290 state indicating that they are able to perform another action.

291 4 Interfaces for Devices and Streams Information Models

292 The *Interaction Model* for implementing *Interfaces* is defined in the MTConnect Standard
 293 as an extension to the MTConnectDevices and MTConnectStreams XML docu-
 294 ments.

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

302 The XML tree in *Figure 3* illustrates where *Interfaces* is modeled in the *Devices Informa-*
 303 *tion Model* for a piece of equipment.

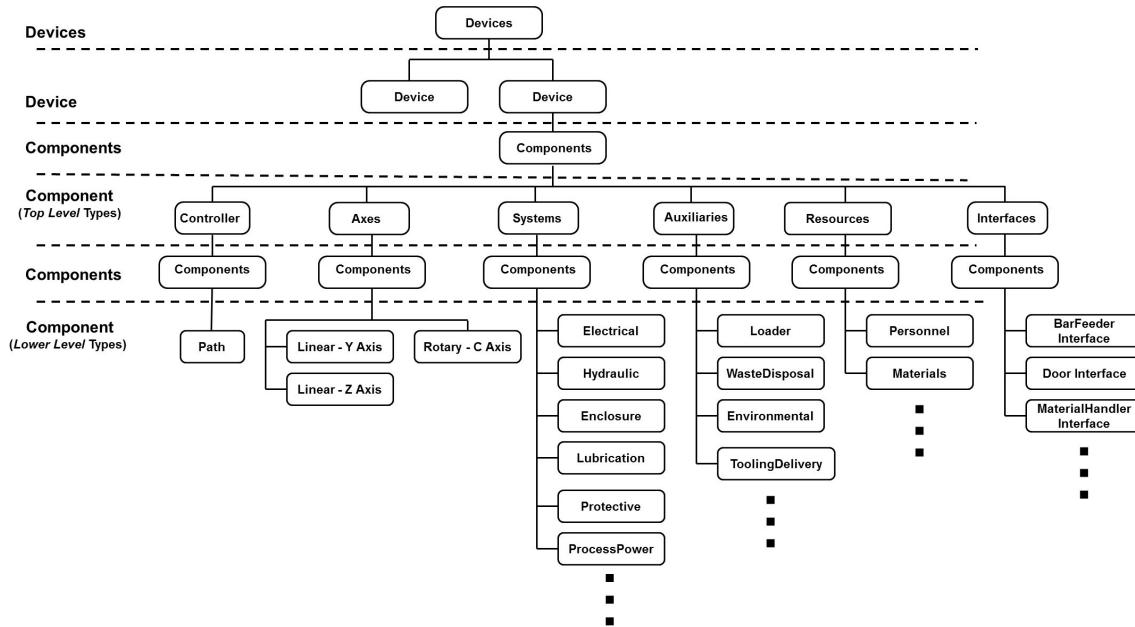


Figure 3: Interfaces as a Structural Element

304 4.1 Interfaces

305 *Interfaces* is an XML *Structural Element* in the MTConnectDevices XML document.
306 *Interfaces* is a container type XML element. *Interfaces* is used to group information de-
307 scribing *Lower Level Interface* XML elements, which each provide information for
308 an individual *Interface*.

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

311 4.2 Interface

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

315 Each *Interface* is also a container type element. As a container, the *Interface*
316 XML element is used to organize information required to implement the *Interaction Model*
317 for an *Interface*. It also provides structure for describing the *Lower Level Structural Ele-
318 ments* associated with the *Interface*. Each *Interface* contains *Data Entities* avail-
319 able from the piece of equipment that may be needed to coordinate activities with associ-
320 ated pieces of equipment.

321 The information provided by a piece of equipment for each *Interface* is returned in a Com-
322 ponentStream container of an MTConnectStreams document in the same manner
323 as all other types of components.

324 4.2.1 XML Schema Structure for Interface

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

326 The schema for an *Interface* element is the same as defined for Component elements
327 described in Section 4.4 in *MTConnect Standard: Part 2.0 - Devices Information Model*
328 of the MTConnect Standard. The *Figure 4* shows the attributes defined for *Interface*
329 and the elements that may be associated with *Interface*.

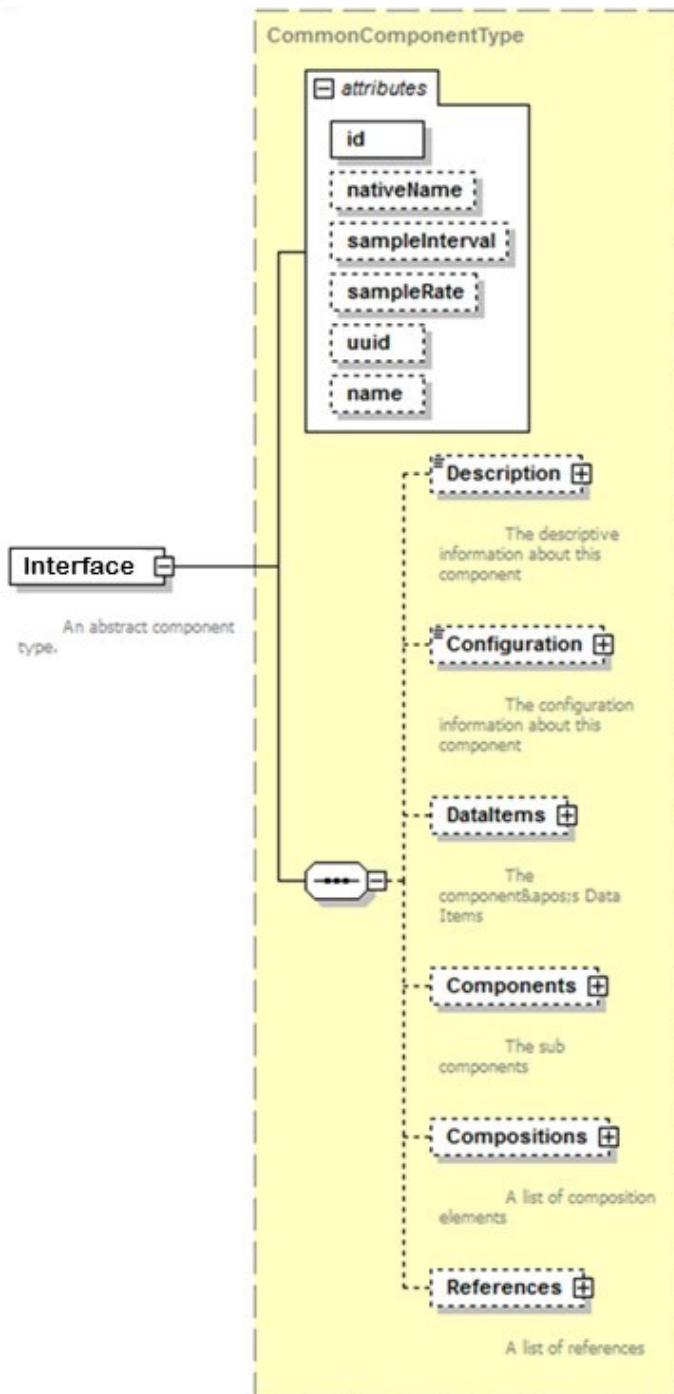


Figure 4: Interface Schema

330 Refer to *MTConnect Standard: Part 2.0 - Devices Information Model*, Section 4.4 for
 331 complete descriptions of the attributes and elements that are illustrated in the *Figure 4* for
 332 Interface.

333 4.2.2 Interface Types

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

Table 2: Interface types

| Interface | Description |
|--------------------|--|
| BarFeederInterface | <p>BarFeederInterface provides the set of information used to coordinate the operations between a Bar Feeder and another piece of equipment.</p> <p>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.</p> |

| Continuation of Table 2 | |
|--------------------------|--|
| Interface | Description |
| MaterialHandlerInterface | <p>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.</p> <p>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:</p> <ul style="list-style-type: none"> Loading/unloading material or tooling Part inspection Testing Cleaning Etc. <p>A robot is a common example of a material handler.</p> |
| DoorInterface | <p>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.</p> <p>The piece of equipment that is controlling the door MUST provide the data item DOOR_STATE as part of the set of information provided.</p> |

| Continuation of Table 2 | |
|-------------------------|--|
| Interface | Description |
| ChuckInterface | <p>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.</p> <p>The piece of equipment that is controlling the chuck MUST provide the data item CHUCK_STATE as part of the set of information provided.</p> |

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

339 In order to implement the *Interaction Model* for *Interfaces*, each piece of equipment as-
 340 sociated with an *Interface* **MUST** provide an `Interface` XML element for that type of
 341 *Interface*. A piece of equipment **MAY** support any number of unique *Interfaces*.

342 4.2.3 Data for Interface

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

347 Details on data items specific to the *Interaction Model* for each type of *Interface* are pro-
 348 vided in *Section 4.2.4 - Data Items for Interface*.

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

351 4.2.3.1 References for Interface

352 Some of the data items needed to support a specific *Interface* may already be defined else-
 353 where in the XML document for a piece of equipment. However, the implementer may
 354 not be able to directly associate this data with the *Interface* since the MTConnect Standard
 355 does not permit multiple occurrences of a piece of data to be configured in a XML docu-
 356 ment. *References* provides a mechanism for associating information defined elsewhere

357 in the *Information Model* for a piece of equipment with a specific *Interface*.

358 References is an XML container that organizes pointers to information defined else-
359 where in the XML document for a piece of equipment. References **MAY** contain one
360 or more Reference XML elements.

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

364 References is an economical syntax for providing interface specific information with-
365 out directly duplicating the occurrence of the data. It provides a mechanism to include all
366 necessary information required for interaction and deterministic information flow between
367 pieces of equipment.

368 For more information on the definition for References and Reference, see Section
369 4.7 and 4.8 of *MTConnect Standard: Part 2.0 - Devices Information Model*.

370 4.2.4 Data Items for Interface

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

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

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

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

381 4.2.4.1 INTERFACE_STATE for Interface

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

385 An INTERFACE_STATE data item **MUST** be defined for every Interface XML ele-

386 ment.

387 INTERFACE_STATE is reported in the MTConnectStreams XML document as In-
 388 terfaceState. InterfaceState reports one of two states – ENABLED or DIS-
 389 ABLED, which are provided in the CDATA for InterfaceState.

390 The *Table 3* shows both the INTERFACE_STATE data item as defined in the MTCon-
 391 nectDevices document and the corresponding *Element Name* that **MUST** be reported
 392 in the MTConnectStreams document.

Table 3: InterfaceState Event

| DataItem Type | Element Name | Description |
|-----------------|----------------|---|
| INTERFACE_STATE | InterfaceState | <p>The current functional or operational state of an <i>Interface</i> type element indicating whether the <i>Interface</i> is active or not currently functioning.</p> <p><i>Valid Data Values:</i></p> <ul style="list-style-type: none"> ENABLED: The <i>Interface</i> is currently operational and performing as expected. DISABLED: The <i>Interface</i> is currently not operational. <p>When the INTERFACE_STATE is DISABLED, the state of all data items that are specific for the <i>Interaction Model</i> associated with that <i>Interface</i> MUST be set to NOT_READY.</p> |

393 4.2.4.2 Specific Data Items for the Interaction Model for Interface

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

400 the associated piece of equipment to indicate its progress in performing its *Response* to the
 401 *Request* for service.

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

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

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

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

420 The *Table 4* also provides the corresponding transformed *Element Name* for each data item
 421 that **MAY** be returned by an *Agent* as an *Event* type XML *Data Entity* in the `MTConnectStreams` XML
 422 document. The *Controlled Vocabulary* for each of these data items
 423 are defined in *Section 4.2.4.3 - Event States for Interfaces*.

Table 4: Event Data Item types for Interface

| DataItem Type | Element Name | Description |
|-------------------|-----------------|---|
| MATERIAL_FEED | MaterialFeed | Service to advance material or feed product to a piece of equipment from a continuous or bulk source. |
| MATERIAL_CHANGE | MaterialChange | Service to change the type of material or product being loaded or fed to a piece of equipment. |
| MATERIAL_-RETRACT | MaterialRetract | Service to remove or retract material or product. |

| Continuation of Table 4 | | |
|-------------------------|----------------|--|
| DataItem Type | Element Name | Description |
| PART_CHANGE | PartChange | Service to change the part or product associated with a piece of equipment to a different part or product. |
| MATERIAL_LOAD | MaterialLoad | Service to load a piece of material or product. |
| MATERIAL_UNLOAD | MaterialUnload | Service to unload a piece of material or product. |
| OPEN_DOOR | OpenDoor | Service to open a door. |
| CLOSE_DOOR | CloseDoor | Service to close a door. |
| OPEN_CHUCK | OpenChuck | Service to open a chuck. |
| CLOSE_CHUCK | CloseChuck | Service to close a chuck. |

424 **4.2.4.3 Event States for Interfaces**

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

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

Table 5: Request States

| Request State | Description |
|---------------|---|
| NOT_READY | The <i>Requester</i> is not ready to make a <i>Request</i> . |
| READY | The <i>Requester</i> is prepared to make a <i>Request</i> , but no <i>Request</i> for service is required. The <i>Requester</i> will transition to ACTIVE when it needs a service to be performed. |
| ACTIVE | The <i>Requester</i> has initiated a <i>Request</i> for a service and the service has not yet been completed by the <i>Responder</i> . |

| Continuation of Table 5 | |
|-------------------------|---|
| Request State | Description |
| FAIL | <p>CONDITION 1:</p> <p>When the <i>Requester</i> has detected a failure condition, it indicates to the <i>Responder</i> to either not initiate an action or stop its action before it completes by changing its state to FAIL.</p> <p>CONDITION 2:</p> <p>If the <i>Responder</i> changes its state to FAIL, the <i>Requester</i> MUST change its state to FAIL.</p> <p>ACTIONS:</p> <p>After detecting a failure, the <i>Requester</i> SHOULD NOT change its state to any other value until the <i>Responder</i> has acknowledged the FAIL state by changing its state to FAIL.</p> <p>Once the FAIL state has been acknowledged by the <i>Responder</i>, the <i>Requester</i> may attempt to clear its FAIL state.</p> <p>As part of the attempt to clear the FAIL state, the <i>Requester</i> 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 <i>Requester</i> changes its Request state from FAIL to READY. If for some reason the <i>Requester</i> is not again prepared to perform a service, it transitions its state from FAIL to NOT_READY.</p> |

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

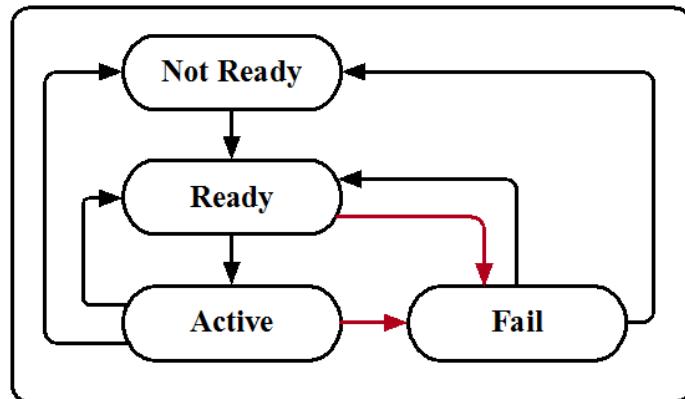


Figure 5: Request State Diagram

432 The *Response* portion of the *Interaction Model* for *Interfaces* has five states as defined in
 433 the *Table 6*.

Table 6: Response States

| Response State | Description |
|----------------|---|
| NOT_READY | The <i>Responder</i> is not ready to perform a service. |
| READY | <p>The <i>Responder</i> is prepared to react to a Request, but no Request for service has been detected.</p> <p>The <i>Responder</i> MUST transition to ACTIVE to inform the <i>Requester</i> that it has detected and accepted the Request and is in the process of performing the requested service.</p> <p>If the <i>Responder</i> is not ready to perform a Request, it MUST transition to a NOT_READY state.</p> |
| ACTIVE | <p>The <i>Responder</i> 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.</p> <p>In normal operation, the <i>Responder</i> MUST NOT change its state to ACTIVE unless the <i>Requester</i> state is ACTIVE.</p> |

| Continuation of Table 6 | |
|-------------------------|--|
| Response State | Description |
| FAIL | <p>CONDITION 1:</p> <p>The <i>Responder</i> has failed while executing the actions required to perform a service and the service has not yet been completed or the <i>Responder</i> has detected that the <i>Requester</i> has unexpectedly changed state.</p> <p>CONDITION 2:</p> <p>If the <i>Requester</i> changes its state to FAIL, the <i>Responder</i> MUST change its state to FAIL.</p> <p>ACTIONS:</p> <p>After entering a FAIL state, the <i>Responder</i> SHOULD NOT change its state to any other value until the <i>Requester</i> has acknowledged the FAIL state by changing its state to FAIL.</p> <p>Once the FAIL state has been acknowledged by the <i>Requester</i>, the <i>Responder</i> may attempt to clear its FAIL state.</p> <p>As part of the attempt to clear the FAIL state, the <i>Responder</i> 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 <i>Responder</i> changes its <i>Response</i> state from FAIL to READY. If for some reason the <i>Responder</i> is not again prepared to perform a service, it transitions its state from FAIL to NOT_READY.</p> |
| COMPLETE | <p>The <i>Responder</i> has completed the actions required to perform the service.</p> <p>The <i>Responder</i> MUST remain in the COMPLETE state until the <i>Requester</i> acknowledges that the service is complete by changing its state to READY.</p> <p>At that point, the <i>Responder</i> 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.</p> |

434 The state values described in the *Table 6* and *Table 6* **MUST** be provided in the CDATA for
 435 each of the *Interface* specific data items provided in the MTConnect Streams document.

436 *Figure 6* shows a graphical representation of the possible state transitions for a *Response*:

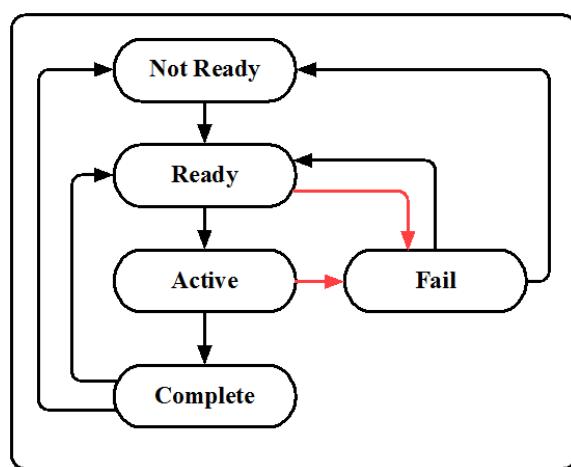


Figure 6: Response State Diagram

437 5 Operation and Error Recovery

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

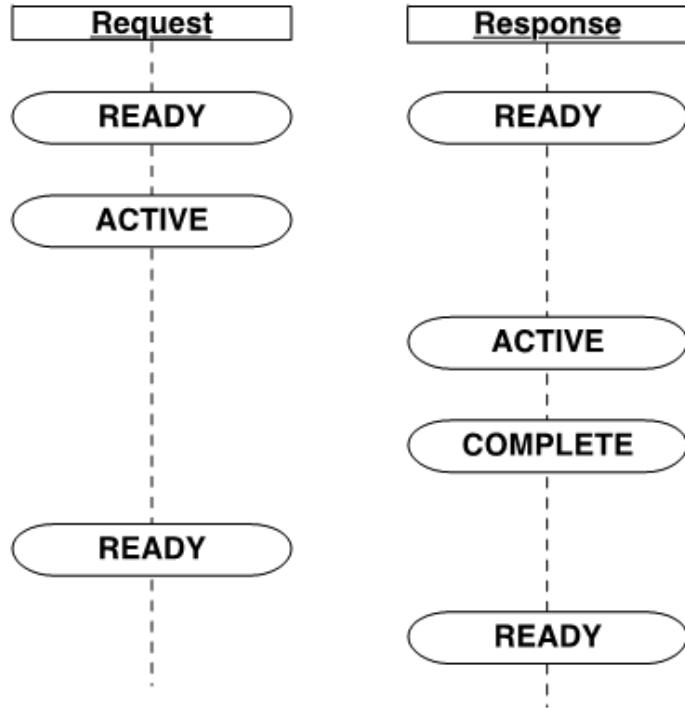


Figure 7: Success Scenario

442 5.1 Request/Response Failure Handling and Recovery

443 A significant feature of the *Request/Response Interaction Model* is the ability for either
 444 piece of equipment to detect a failure associated with either the *Request* or *Response* ac-
 445 tions. When either a failure or unexpected action occurs, the *Request* and the *Response*
 446 portion of the *Interaction Model* can announce a FAIL state upon detecting a problem. The
 447 following are graphical models describing multiple scenarios where either the *Requester*
 448 or *Responder* detects and reacts to a failure. In these examples, either the *Requester* or *Re-*
 449 *sponder* announces the detection of a failure by setting either the *Request* or the *Response*
 450 state to FAIL.

451 Once a failure is detected, the *Interaction Model* provides information from each piece of

452 equipment as they attempt to recover from a failure, reset all of their functions associated
 453 with the *Interface* to their original state, and return to normal operation.

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

457 Scenario #1 – Responder Fails Immediately

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

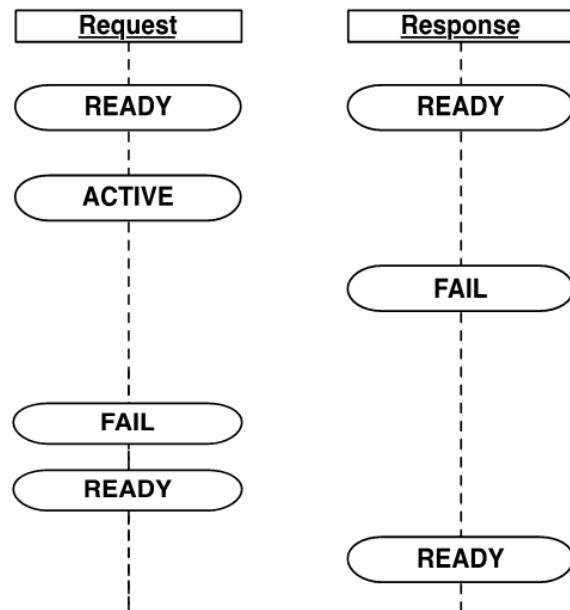


Figure 8: Responder - Immediate Failure

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

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

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

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

475 Scenario #2 – Responder Fails While Providing a Service

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

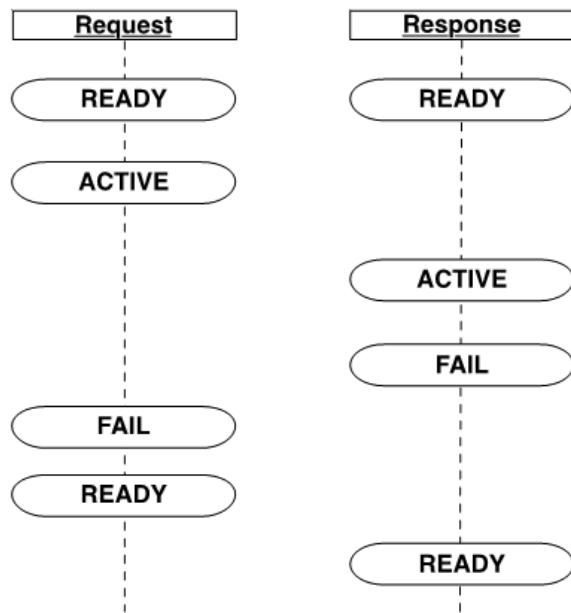


Figure 9: Responder Fails While Providing a Service

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

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

487 The *Responder*, as part of clearing a failure, resets any partial actions that were initiated
 488 and attempts to return to a condition where it is again ready to perform a service. If the
 489 recovery is successful, the *Responder* changes its *Response* state from FAIL to READY if

490 it is again prepared to perform a service. If for some reason the *Responder* is not again
 491 prepared to perform a service, it transitions its state from FAIL to NOT_READY.

492 Scenario #3 – Requester Failure During a Service Request

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

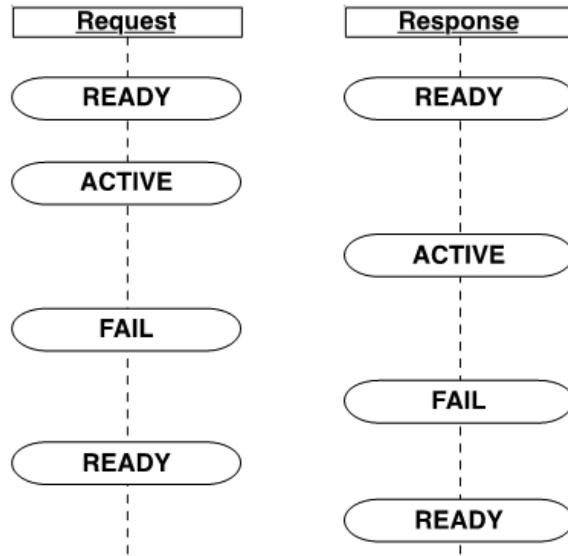


Figure 10: Requester Fails During a Service Request

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

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

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

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

509 In some cases, a *Requester* may transition to an unexpected state after it has initiated a

510 Request for service.

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

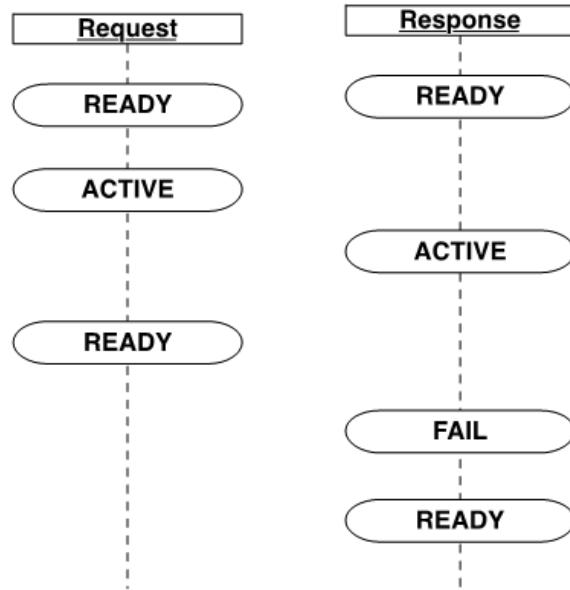


Figure 11: Requester Makes Unexpected State Change

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

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

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

525 Note: The same scenario exists if the *Requester* transitions its *Request* state to NOT_--
 526 READY. However, in this case, the *Requester* then transitions its *Request* state
 527 to READY after it resets all of its functions back to a condition where it is again
 528 prepared to make a *Request* for service.

529 Scenario #5 – Responder Changes to an Unexpected State While Providing a Service

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

532 As demonstrated in *Figure 12*, the *Responder* is performing the actions to provide a ser-
533 vice and the *Response* state is ACTIVE. During these actions, the *Responder* transitions its
534 state to NOT_READY before completing its actions. This should be regarded as a failure
535 of the *Responder*.

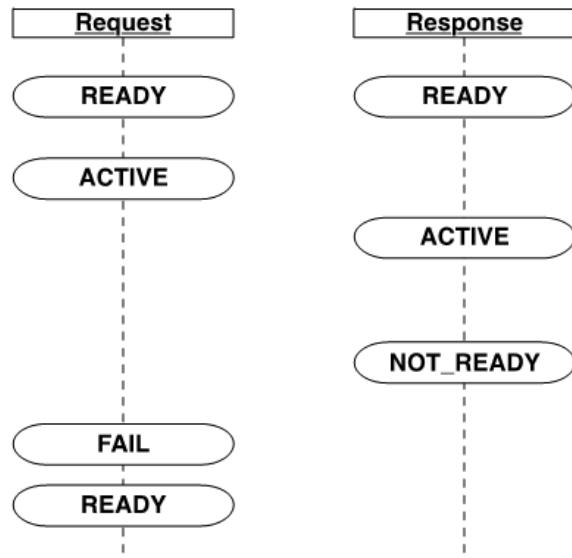


Figure 12: Responder Makes Unexpected State Change

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

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

543 Since the *Responder* has failed to an invalid state, the condition of the *Responder* is un-
544 known. Where possible, the *Responder* should try to reset to an initial state.

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

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

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

552 In this scenario, a failure occurs in the communications connection between the *Responder*
 553 and *Requester*. This failure may result from the InterfaceState from either piece of
 554 equipment returning a value of UNAVAILABLE or one of the pieces of equipment does
 555 not provide a heartbeat within the desired amount of time (See *MTConnect Standard Part*
 556 *1.0 - Overview and Fundamentals* for details on heartbeat).

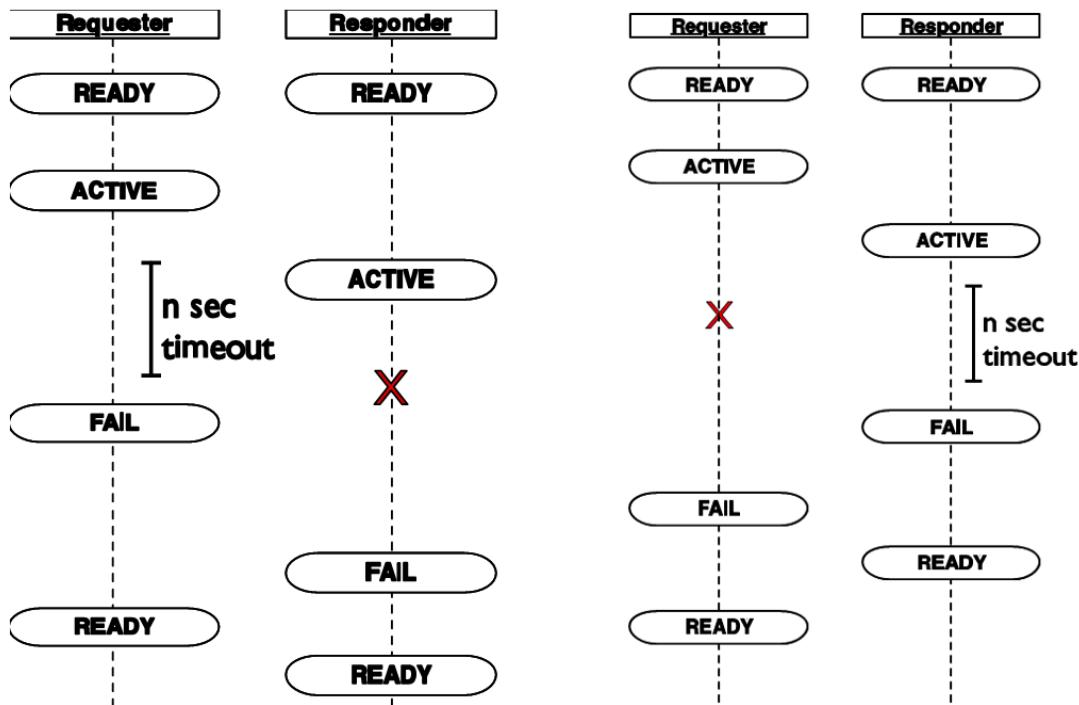


Figure 13: Requester/Responder Communication Failures

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

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

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

566 a service, it transitions its state from FAIL to NOT_READY.

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

572 Appendices

573 A Bibliography

- 574 Engineering Industries Association. *EIA Standard - EIA-274-D*, Interchangeable Variable,
575 Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically
576 Controlled Machines. Washington, D.C. 1979.
- 577 ISO TC 184/SC4/WG3 N1089. *ISO/DIS 10303-238*: Industrial automation systems and
578 integration Product data representation and exchange Part 238: Application Protocols: Ap-
579 plication interpreted model for computerized numerical controllers. Geneva, Switzerland,
580 2004.
- 581 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
582 tems and integration – Physical device control – Data model for computerized numerical
583 controllers – Part 10: General process data. Geneva, Switzerland, 2004.
- 584 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
585 tems and integration – Physical device control – Data model for computerized numerical
586 controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.
- 587 International Organization for Standardization. *ISO 6983/1* – Numerical Control of ma-
588 chines – Program format and definition of address words – Part 1: Data format for posi-
589 tioning, line and contouring control systems. Geneva, Switzerland, 1982.
- 590 Electronic Industries Association. *ANSI/EIA-494-B-1992*, 32 Bit Binary CL (BCL) and
591 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines.
592 Washington, D.C. 1992.
- 593 National Aerospace Standard. *Uniform Cutting Tests* - NAS Series: Metal Cutting Equip-
594 ment Specifications. Washington, D.C. 1969.
- 595 International Organization for Standardization. *ISO 10303-11*: 1994, Industrial automa-
596 tion systems and integration Product data representation and exchange Part 11: Descrip-
597 tion methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.
- 598 International Organization for Standardization. *ISO 10303-21*: 1996, Industrial automa-
599 tion systems and integration – Product data representation and exchange – Part 21: Imple-
600 mentation methods: Clear text encoding of the exchange structure. Geneva, Switzerland,
601 1996.
- 602 H.L. Horton, F.D. Jones, and E. Oberg. *Machinery's Handbook*. Industrial Press, Inc.

- 603 New York, 1984.
- 604 International Organization for Standardization. *ISO 841-2001: Industrial automation sys-*
605 *tems and integration - Numerical control of machines - Coordinate systems and motion*
606 *nomenclature*. Geneva, Switzerland, 2001.
- 607 ASME B5.57: *Methods for Performance Evaluation of Computer Numerically Controlled*
608 *Lathes and Turning Centers*, 1998.
- 609 ASME/ANSI B5.54: *Methods for Performance Evaluation of Computer Numerically Con-*
610 *trolled Machining Centers*. 2005.
- 611 OPC Foundation. *OPC Unified Architecture Specification, Part 1: Concepts Version 1.00*.
612 July 28, 2006.
- 613 IEEE STD 1451.0-2007, *Standard for a Smart Transducer Interface for Sensors and Ac-*
614 *tuators – Common Functions, Communication Protocols, and Transducer Electronic Data*
615 *Sheet (TEDS) Formats*, IEEE Instrumentation and Measurement Society, TC-9, The In-
616 *stitute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH99684,*
617 *October 5, 2007*.
- 618 IEEE STD 1451.4-1994, *Standard for a Smart Transducer Interface for Sensors and Ac-*
619 *tuators – Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet*
620 *(TEDS) Formats*, IEEE Instrumentation and Measurement Society, TC-9, The Institute of
621 *Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH95225, December*
622 *15, 2004*.