



MTConnect® American National Standard

Version 1.7.0

CONTENTS

Part 1 - Overview and Fundamentals v1.7.0

Part 2 - Devices v1.7.0

Part 3 - Streams v1.7.0

Part 4 - Assets v1.7.0

Part 4.1 - Cutting Tools v1.7.0

Part 4.2 - File Asset Information Model v1.7.0

Part 4.3 - Raw Material Asset Information Model v1.7.0

Part 4.4 - QIF Asset Information Model v1.7.0

Part 5 - Interfaces v1.7.0



MTConnect[®] Standard

Part 1.0 – Overview and Fundamentals

Version 1.8.0

Prepared for: MTConnect Institute
Prepared on: September 6, 2021

MTConnect[®] is a registered trademark of AMT - The Association for Manufacturing Technology. Use of *MTConnect* is limited to use as specified on <http://www.mtconnect.org/>.

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	31
4	MTConnect Standard	32
4.1	MTConnect Documents Organization	32
4.2	MTConnect Document Versioning	33
4.2.1	Document Releases	34
4.3	MTConnect Document Naming Conventions	35
4.3.1	Document Title	35
4.3.2	Electronic Document File Naming	35
4.4	Document Conventions	36
4.4.1	Use of MUST, SHOULD, and MAY	36
4.4.2	Text Conventions	36
4.4.3	Code Line Syntax and Conventions	37
4.4.4	Semantic Data Model Content	38
4.4.5	Referenced Standards and Specifications	38
4.4.6	Deprecation and Deprecation Warnings	39
4.4.6.1	Deprecation	39
4.4.6.2	Deprecation Warning	39
4.5	Backwards Compatibility	40
5	MTConnect Fundamentals	41
5.1	Agent	41
5.1.1	Instance of an Agent	43
5.1.2	Storage of Equipment Metadata for a Piece of Equipment	43
5.1.3	Storage of Streaming Data	44
5.1.3.1	Management of Streaming Data Storage	44
5.1.3.2	Sequence Numbers	45
5.1.3.3	Buffer Data Structure	48
5.1.3.4	Time Stamp	49
5.1.3.5	Recording Occurrences of Streaming Data	50
5.1.3.6	Maintaining Last Value for Data Entities	50
5.1.3.7	Unavailability of Data	51
5.1.3.8	Persistence and Recovery	51
5.1.3.9	Heartbeat	52
5.1.3.10	Data Sets	52

5.1.4	Storage of Documents for MTConnect Assets	52
5.2	Response Documents	54
5.2.1	XML Documents	55
5.3	Semantic Data Models	56
5.4	Request/Response Information Exchange	57
5.5	Accessing Information from an Agent	58
5.5.1	Accessing Equipment Metadata from an Agent	58
5.5.2	Accessing Streaming Data from the Buffer of an Agent	58
5.5.3	Accessing MTConnect Assets Information from an Agent	60
6	XML Representation of Response Documents	61
6.1	Fundamentals of Using XML to Encode Response Documents	62
6.2	XML Declaration	63
6.3	Root Element	63
6.3.1	MTConnectDevices Root Element	63
6.3.1.1	MTConnectDevices Elements	64
6.3.2	MTConnectStreams Root Element	65
6.3.2.1	MTConnectStreams Elements	66
6.3.3	MTConnectAssets Root Element	66
6.3.3.1	MTConnectAssets Elements	67
6.3.4	MTConnectError Root Element	67
6.3.4.1	MTConnectError Elements	68
6.4	Schema and Namespace Declaration	69
6.5	Document Header	69
6.5.1	Header for MTConnectDevices	70
6.5.1.1	XML Schema Structure for Header for MTConnectDe- vices	70
6.5.1.2	Attributes for Header for MTConnectDevices	70
6.5.2	Header for MTConnectStreams	75
6.5.2.1	XML Schema Structure for Header for MTConnectStreams	75
6.5.2.2	Attributes for MTConnectStreams Header	75
6.5.3	Header for MTConnectAssets	80
6.5.3.1	XML Schema Structure for Header for MTConnectAssets	81
6.5.3.2	Attributes for Header for MTConnectAssets	81
6.5.4	Header for MTConnectError	85
6.5.4.1	XML Schema Structure for Header for MTConnectError	85
6.5.4.2	Attributes for Header for MTConnectError	86
6.6	Document Body	90
6.7	Extensibility	91
7	Protocol and Messaging	94
8	HTTP Messaging Supported by an Agent	96

8.1	REST Interface	96
8.2	HTTP Request	96
8.2.1	authority Portion of an HTTP Request Line	97
8.2.2	path Portion of an HTTP Request Line	98
8.2.3	query Portion of an HTTP Request Line	98
8.3	MTConnect Request/Response Information Exchange Implemented with HTTP	98
8.3.1	Probe Request Implemented Using HTTP	99
8.3.1.1	Path Portion of the HTTP Request Line for a Probe Request	99
8.3.1.2	Query Portion of the HTTP Request Line for a Probe Request	99
8.3.1.3	Response to a Probe Request	100
8.3.1.4	HTTP Status Codes for a Probe Request	100
8.3.2	Current Request Implemented Using HTTP	102
8.3.2.1	Path Portion of the HTTP Request Line for a Current Request	102
8.3.2.2	Query Portion of the HTTP Request Line for a Current Request	102
8.3.2.3	Response to a Current Request	104
8.3.2.4	HTTP Status Codes for a Current Request	105
8.3.3	Sample Request Implemented Using HTTP	107
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	108
8.3.3.3	Response to a Sample Request	114
8.3.3.4	HTTP Status Codes for a Sample Request	114
8.3.4	Asset Request Implemented Using HTTP	116
8.3.4.1	Path Portion of the HTTP Request Line for an Asset Request	117
8.3.4.2	Query Portion of the HTTP Request Line for an Asset Request	117
8.3.4.3	Response to an Asset Request	118
8.3.4.4	HTTP Status Codes for a Asset Request	119
8.3.5	HTTP Errors	120
8.3.6	Streaming Data	121
8.3.6.1	Heartbeat	122
8.3.7	References	123
9	Error Information Model	124
9.1	MTConnectError Response Document	124
9.1.1	Structural Element for MTConnectError	124

9.1.2	Error Data Entity	126
9.1.2.1	XML Schema Structure for Error	126
9.1.2.2	Attributes for Error	127
9.1.2.3	Values for errorCode	127
9.1.2.4	CDATA for Error	129
9.1.3	Examples for MTConnectError	129

Appendices	131
-------------------	------------

A	Bibliography	131
B	Fundamentals of Using XML to Encode Response Documents	133
C	Schema and Namespace Declaration Information	136

Table of Figures

Figure 1: Basic MTConnect Implementation Structure	4
Figure 2: MTConnect Architecture Model	41
Figure 3: Data Storage in Buffer	44
Figure 4: First In First Out Buffer Management	44
Figure 5: instanceId and sequence	46
Figure 6: Identifying the range of data with firstSequence and lastSequence .	46
Figure 7: Identifying the range of data with from and count	47
Figure 8: Identifying the range of data with nextSequence and lastSequence .	48
Figure 9: Data Storage Concept	49
Figure 10:First In First Out Asset Buffer Management	53
Figure 11:Relationship between assetId and stored Asset documents	53
Figure 12:Example Buffer	59
Figure 13:MTConnectDevices Structure	64
Figure 14:MTConnectStreams Structure	65
Figure 15:MTConnectAssets Structure	66
Figure 16:MTConnectError Structure	68
Figure 17:Header Schema Diagram for MTConnectDevices	70
Figure 18:Header Schema Diagram for MTConnectStreams	75
Figure 19:Header Schema Diagram for MTConnectAssets	81
Figure 20:Header Schema Diagram for MTConnectError	86
Figure 21:Errors Schema Diagram	125
Figure 22:Error Schema Diagram	127

List of Tables

Table 1: Elements for MTConnectDevices	64
Table 2: Elements for MTConnectStreams	66
Table 3: Elements for MTConnectAssets	67
Table 4: Elements for MTConnectError	69
Table 5: MTConnectDevices Header	71
Table 6: MTConnectStreams Header	76
Table 7: MTConnectAssets Header	82
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	99
Table 11: HTTP Status Codes for a Probe Request	100
Table 12: Path of the HTTP Request Line for a Current Request	102
Table 13: Query Parameters of the HTTP Request Line for a Current Request	103
Table 14: HTTP Status Codes for a Current Request	105
Table 15: Path of the HTTP Request Line for a Sample Request	108
Table 16: Query Parameters of the HTTP Request Line for a Sample Request	108
Table 17: HTTP Status Codes for a Sample Request	115
Table 18: Path of the HTTP Request Line for an Asset Request	117
Table 19: Query Parameters of the HTTP Request Line for an Asset Request	118
Table 20: HTTP Status Codes for an Asset Request	119
Table 21: MTConnect Errors Element	125
Table 22: Attributes for Error	127
Table 23: Values for errorCode	128

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.

To assist in implementation, the MTConnect Standard is built upon the most prevalent standards in the manufacturing and software industries. This maximizes the number of software tools available for implementation and provides the highest level of interoperability with other standards, software applications, and equipment used throughout manufacturing operations.

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

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

Note: The term "document" is used with different meanings in the MTConnect Standard:

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

The following will be used throughout the MTConnect Standard to distinguish between these different meanings for the term "document":

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

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

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

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

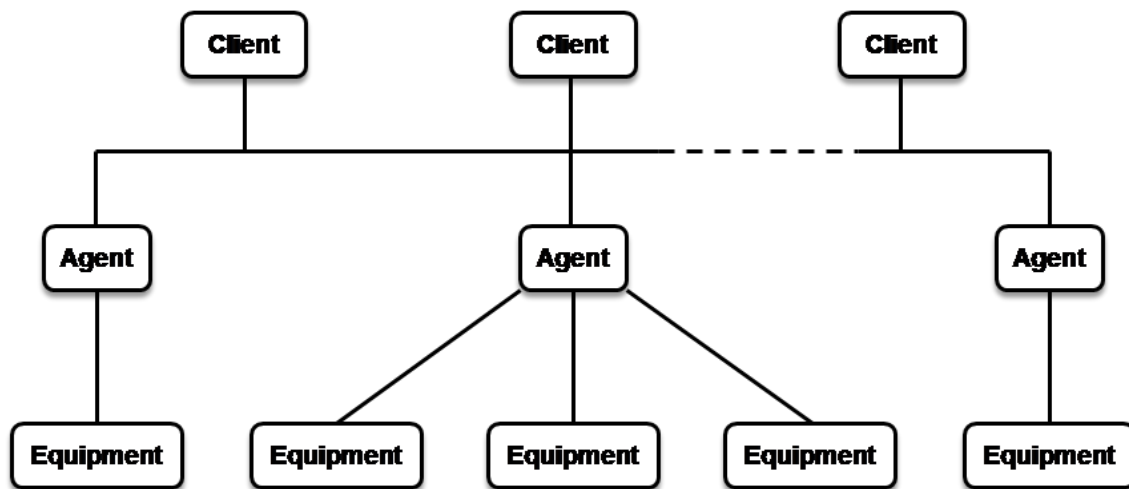


Figure 1: Basic MTConnect Implementation Structure

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

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

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

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

Client Software Application: Software that requests data from *Agents* and processes 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 *nect 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 ele-
143 ment.

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 The World Wide Web Consortium (W3C) is an international community that devel-
191 ops open standards to ensure the long-term growth of the Web.

192 See <https://www.w3.org/>.

193 XML

194 Stands for eXtensible Markup Language.

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

197 XML is the language used for all code examples in the MTConnect Standard.

198 Refer to <http://www.w3.org/XML> for more information about XML.

199 XPath

200 General meaning:

201 XPath is a command structure that describes a way for a software system to locate
202 information in an XML document.

203 XPath uses an addressing syntax based on a path through the document's logical
204 structure.

205 See <http://www.w3.org/TR/xpath> for more information on XPath.

206 Appears in the documents in the following form: XPath.

207 ***Abstract Element***

208 An element that defines a set of common characteristics that are shared by a group
209 of elements.

210 An abstract element cannot appear in a document. In a specific implementation of
211 a schema, an abstract element is replaced by a derived element that is itself not an
212 abstract element. The characteristics for the derived element are inherited from the
213 abstract element.

214 Appears in the documents in the following form: abstract.

215 ***Adapter***

216 An optional piece of hardware or software that transforms information provided by
217 a piece of equipment into a form that can be received by an *Agent*.

218 Appears in the documents in the following form: adapter.

219 ***Agent***

220 Refers to an MTConnect Agent.

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

225 Appears in the documents in the following form: *Agent*.

226 ***alarm limits***

227 A set of limits used to trigger warning or alarm indicators.

228 ***Application Programming Interface***

229 A set of methods to provide communications between software applications.

230 The API defined in the MTConnect Standard describes the methods for providing
231 the *Request/Response* Information Exchange between an *Agent* and client software
232 applications.

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

235 ***Archetype***

236 General Description of an *MTConnect Asset*:

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

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

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

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

243 Appears in the documents in the following form: Archetype

244 ***Asset***

245 item, thing or entity that has potential or actual value to an organization *Ref:ISO*
246 *55000:2014(en)*

247 Note 1 to entry: Value can be tangible or intangible, financial or non-financial,
248 and includes consideration of risks and liabilities. It can be positive or negative
249 at different stages of the asset life.

250 Note 2 to entry: Physical assets usually refer to equipment, inventory and prop-
251 erties owned by the organization. Physical assets are the opposite of intangible
252 assets, which are non-physical assets such as leases, brands, digital assets, use
253 rights, licences, intellectual property rights, reputation or agreements.

254 Note 3 to entry: A grouping of assets referred to as an asset system could also
255 be considered as an asset.

256

257 ***Asset Document***

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

260 ***Attachment***

261 The connection by which one thing is associated with another.

262 ***Attribute***

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

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

265 **Base Functional Structure**

266 A consistent set of functionalities defined by the MTConnect Standard. This func-
 267 tionality includes the protocol(s) used to communicate data to a client software ap-
 268 plication, the *semantic data models* defining how that data is organized into *Re-*
 269 *sponse Documents*, and the encoding of those *Response Documents*.
 270 Appears in the documents in the following form: *Base Functional Structure*.

271 **buffer**

272 General meaning:

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

275 Used relative to *Streaming Data*:

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

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

279 Used relative to *MTConnect Assets*:

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

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

282 **Child Element**

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

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

286 **Client**

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

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

291 **Component**

292 General meaning:

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

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

296 Used in *Information Models*:

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

- 309 • When used as an XML container to organize *Lower Level* Component ele-
300 ments.
- 301 Appears in the documents in the following form: *Components*.
- 302 • When used as an abstract XML element. *Component* is replaced in a data
303 model by a type of *Component* element. *Component* is also an XML con-
304 tainer used to organize *Lower Level* Component elements, *Data Entities*, or
305 both.
- 306 Appears in the documents in the following form: *Component*.

307 ***Composition***

308 General meaning:

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

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

312 Used in *Information Models*:

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

- 315 • When used as an XML container to organize *Composition* elements.
- 316 Appears in the documents in the following form: *Compositions*
- 317 • When used as an abstract XML element. *Composition* is replaced in a data
318 model by a type of *Composition* element.
- 319 Appears in the documents in the following form: *Composition*.

320 ***Condition***

321 An indicator of the ability of a piece of equipment or *Component* to function to
322 specification.

323 ***control limits***

324 A set of limits used to indicate whether a process variable is stable and in control.

325 ***Controlled Vocabulary***

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

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

329 ***current***

330 occurring in or existing at the present time.

331 ***Current Request***

332 A *Current Request* is a *Request* to an *Agent* to produce an *MTConnectStreams Re-*
 333 *sponse Document* containing the *Observations Information Model* for a snapshot of
 334 the latest *observations* at the moment of the *Request* or at a given *sequence number*.

335 ***data dictionary***

336 Listing of standardized terms and definitions used in *MTConnect Information Mod-*
 337 *els*.

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

339 ***Data Entity***

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

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

344 ***Data Item***

345 General meaning:

346 Descriptive information or properties and characteristics associated with a *Data En-*
 347 *tity*.

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

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

- 350 • When used as an XML container to organize `DataItem` elements.
- 351 Appears in the documents in the following form: `DataItems`.
- 352 • When used to represent a specific *Data Entity*, the form `DataItem` is an XML
- 353 element.
- 354 Appears in the documents in the following form: `DataItem`.

355 ***Data Set***

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

357 ***Data Source***

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

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

360 ***Data Streaming***

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

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

364 ***Deprecated***

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

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

369 ***Deprecation Warning***

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

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

373 ***Devices Information Model***

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

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

377 ***Document***

378 A piece of written, printed, or electronic matter that provides information or evi-
379 dence that serves as an official record.

380 ***Document Body***

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

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

385 ***Document Header***

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

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

391 ***electric current***

392 The rate of flow of electric charge.

393 ***Element***

394 Refers to an XML element.

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

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

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

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

402 ***Element Name***

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

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

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

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

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

410 ***engineering units***

411 A quantity, dimension, or magnitude used in engineering adopted as a standard in
412 terms of which the magnitude of other quantities of the same kind can be expressed
413 or calculated.

414 ***Equipment***

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

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

419 ***Equipment Metadata***

420 See *Metadata*

421 ***Error Information Model***

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

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

427 ***Extensible***

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

430 ***Fault State***

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

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

434 ***Force***

435 A push or pull on a mass which results in an acceleration.

436 ***heartbeat***

437 General meaning:

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

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

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

443 The form `heartbeat` is used as a parameter in the query portion of an *HTTP*
444 *Request Line*.

445 Appears in the documents in the following form: `heartbeat`.

446 ***Higher Level***

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

448 ***HTTP Error Message***

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

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

453 ***HTTP Header***

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

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

457 ***HTTP Message***

458 An *HTTP Message* consists of requests from client to server and responses from
459 server to client. *Ref:IETF:RFC-2616*

460 ***HTTP Method***

461 In the MTConnect Standard, a portion of a command in an *HTTP Request* that indi-
462 cates the desired action to be performed on the identified resource; often referred to
463 as verbs.

464 ***HTTP Request***

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

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

469 ***HTTP Request Line***

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

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

473 ***HTTP Response***

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

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

478 ***HTTP Server***

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

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

483 ***HTTP Status Code***

484 In the MTConnect Standard, a numeric code contained in an *HTTP Response* that
485 defines a status category associated with the *Response* either as a success status or a
486 category of an HTTP error.

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

488 ***id***

489 General meaning:

490 An identifier used to distinguish a piece of information.

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

492 Used as an XML attribute:

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

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

496 ***Implementation***

497 A specific instantiation of the MTConnect Standard.

498 ***Information Model***

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

501 For example, an information model is used to define the structure for each *MTCon-*
502 *nect Response Document*; the definition of each piece of information within those
503 documents and the relationship between pieces of information.

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

505 ***instance***

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

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

509 ***Interaction Model***

510 Defines how information is exchanged across an *Interface* between independent sys-
511 tems.

512 ***Interface***

513 The means by which communication is achieved between independent systems.

514 ***key***

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

516 ***key-value pair***

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

520 ***Lower Level***

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

522 ***lower limit***

523 The lower conformance boundary for a variable.

524 Note: immediate concern or action may be required.

525 ***lower warning***

526 The lower boundary indicating increased concern and supervision may be required.

527 ***maximum***

528 A numeric upper constraint.

529 ***Message***

530 A communication in writing, in speech, or by signals.

531 ***Metadata***

532 Data that provides information about other data.

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

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

538 ***minimum***

539 A numeric lower constraint.

540 ***MTConnect Agent***

541 See definition for *Agent*.

542 ***MTConnect Asset***

543 An *MTConnect Asset* is an *Asset* used by the manufacturing process to perform tasks.

545 Note 1 to entry: An *MTConnect Asset* relies upon an *MTConnect Device* to provide *observations* and information about itself and the *MTConnect Device* revises the information to reflect changes to the *MTConnect Asset* during their interaction. Examples of *MTConnect Assets* are Cutting Tools, Part Information, Manufacturing Processes, Fixtures, and Files.

550 Note 2 to entry: A singular `assetId` uniquely identifies an *MTConnect Asset*
 551 throughout its lifecycle and is used to track and relate the *MTConnect Asset* to
 552 other *MTConnect Devices* and entities.

553 Note 3 to entry: *MTConnect Assets* are temporally associated with a device and
 554 can be removed from the device without damage or alteration to its primary
 555 functions.

556

557 ***MTConnect Device***

558 An *MTConnect Device* is a piece of equipment or a manufacturing system that pro-
 559 duces *observations* about itself and/or publishes data using the *MTConnect Infor-*
 560 *mation Model*.

561 ***MTConnect Document***

562 Printed or electronic document(s) that represent a Part(s) of the MTConnect Stan-
 563 dard.

564 ***MTConnect Event***

565 An *MTConnect Event* is an *observation* of either a state or discrete value of the
 566 *Component*. *Component* states **SHOULD** have a controlled vocabulary.

567 ***MTConnect Information Model***

568 See *Information Model*

569 ***MTConnect Interface***

570 An *Interaction Model* for interoperability between pieces of equipment.

571 ***MTConnect Request***

572 A communication request for information issued from a client software application
 573 to an *Agent*.

574 Appears in the documents in the following form: *MTConnect Request*.

575 ***MTConnect XML Document***

576 See *Response Document*.

577 ***MTConnectAssets Response Document***

578 A *Response Document* published by an *MTConnect Agent* in response to an *Asset*
 579 *Request*.

580 ***MTConnectDevices Response Document***

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

583 ***MTConnectErrors Response Document***

584 An electronic document published by an *Agent* whenever it encounters an error
585 while interpreting a *Request* for information from a client software application or
586 when an *Agent* experiences an error while publishing the *Response* to a *Request* for
587 information.

588 Appears in the documents in the following form: *MTConnectErrors Response Doc-*
589 *ument*.

590 ***MTConnectStreams Response Document***

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

593 ***nominal***

594 The ideal or desired value for a variable.

595 ***observable***

596 A quality, property, or characteristic that can be observed.

597 ***observation***

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

599 ***Observations Information Model***

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

602 ***observe***

603 The act of measuring or determining the value of a property at a point in time.

604 ***organize***

605 The act of containing and owning one or more elements.

606 ***organizer***

607 An element that contains and owns one or more elements.

608 ***parameter***

609 General Meaning:

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

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

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

616 Appears in the documents in the following form: *parameter*.

617 ***Parent Element***

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

620 Appears in the documents in the following form: *Parent Element*.

621 ***Part***

622 *Part* is defined as a discrete item that has both defined and measurable physical
623 characteristics including mass, material and features and is created by applying one
624 or more manufacturing process steps to a workpiece.

625 ***Persistence***

626 A method for retaining or restoring information.

627 ***Probe***

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

630 ***Probe Request***

631 A *Probe Request* is a *Request* to an *Agent* to produce an *MTConnectDevices Re-*
632 *sponse Document* containing the *Devices Information Model*.

633 ***Protocol***

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

636 ***Publish/Subscribe***

637 In the MTConnect Standard, a communications messaging pattern that may be used
638 to publish *Streaming Data* from an *Agent*. When a *Publish/Subscribe* communi-
639 cation method is established between a client software application and an *Agent*,

the *Agent* will repeatedly publish a specific `MTConnectStreams` document at a defined period.

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

Query

General Meaning:

A portion of a request for information that more precisely defines the specific information to be published in response to the request.

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

Used in an HTTP Request Line:

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

Appears in the documents in the following form: `query`.

raw material

Crude or processed material that can be converted by manufacture, processing, or combination into a new and useful product.

Reference

Reference is a pointer to information that is associated with another *Structural Element*.

Request

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

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

Request/Response

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

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

Requester

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

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

671 ***reset***

672 A reset is associated with an occurrence of a *Data Entity* indicated by the `reset-`
 673 `Triggered` attribute. When a reset occurs, the accumulated value or statistic are
 674 reverted back to their initial value. A *Data Entity* with a *Data Set* representation
 675 removes all *key-value pairs*, setting the *Data Set* to an empty set.

676 ***Responder***

677 An entity that responds to a *Request* for information in a communications exchange.
 678 Appears in the documents in the following form: *Responder*.

679 ***Response Document***

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

682 ***Root Element***

683 The first *Structural Element* provided in a *Response Document* encoded using XML.
 684 The *Root Element* is an XML container and is the *Parent Element* for all other XML
 685 elements in the document. The *Root Element* appears immediately following the
 686 XML Declaration.

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

688 ***Sample***

689 General meaning:

690 The collection of one or more pieces of information.

691 Used when referring to the collection of information:

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

693 Appears in the documents in the following form: *sample*.

694 Used as an *MTConnect Request*:

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

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

698 Used as part of an *HTTP Request*:

699 Used in the `path` portion of an *HTTP Request Line*, by a client software applica-
 700 tion, to initiate a *Sample Request* to an *Agent* to publish an `MTConnectStreams`
 701 document.

702 Appears in the documents in the following form: `sample`.

703 Used to describe a *Data Entity*:

Used to define a specific type of *Data Entity*. A *Sample* type *Data Entity* reports the value for a continuously variable or analog piece of information.

Appears in the documents in the following form: *Sample* or *Samples*.

Used as an XML container or element:

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

Appears in the documents in the following form: *Samples*.

- When used as an abstract XML element. It is replaced in the XML document by types of *Sample* elements representing individual *Sample* type of *Data Entity*.

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

Sample Request

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

schema

General meaning:

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

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

Used in association with an *MTConnect Response Document*:

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

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

semantic data model

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

It provides the rules for encoding electronic information such that it can be interpreted by a software system.

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

sensing element

A mechanism that provides a signal or measured value.

735 **Sensor**

736 A *sensing element* that responds to a physical stimulus and transmits a resulting
737 signal.

738 **Sensor Configuration**

739 Data in the *MTConnectDevices Response Document* that provides the information
740 required for maintenance and support of the *sensor unit*.

741 **Sensor Data**

742 The value of a physical quantity reported by a measuring instrument or controller as
743 an *observation*.

744 **sensor element**

745 A *sensor element* provides a signal or measured value.

746 **sensor unit**

747 An intelligent piece of equipment that manages the signals of one or more *sensing*
748 *elements* and provides the measured values.

749 **sequence number**

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

752 *sequence number* is a monotonically increasing number within an instance of an
753 *Agent*.

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

755 **specification limits**

756 A set of limits defining a range of values designating acceptable performance for a
757 variable.

758 **Spindle**

759 A mechanism that provides rotational capabilities to a piece of equipment.

760 Typically used for either work holding, materials or cutting tools.

761 **Standard**

762 General meaning:

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

765 Used when referring to the MTConnect Standard:

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

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

Streaming Data

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

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

Streams Information Model

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

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

Structural Element

General meaning:

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

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

Used to indicate hierarchy of Components:

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

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

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

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

subtype

General meaning:

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

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

Appears in the documents in the following form: subtype.

Used as an attribute for a *Data Entity*:

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

Appears in the documents in the following form: subType.

800 ***Table***

801 A two dimensional set of values given by a set of *key-value pairs Table Entries*.
802 Each *Table Entry* contains a set of *key-value pairs* of *Table Cells*. The `Entry` and
803 `Cell` elements comprise a tabular representation of the information.

804 ***Table Cell***

805 A subdivision of a *Table Entry* representing a singular value.

806 ***Table Entry***

807 A subdivision of a *Table* containing a set of *key-value pairs* representing *Table Cells*.

808 ***time stamp***

809 General meaning:

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

812 Appears in the documents as "time stamp".

813 Used as an attribute for recorded or published data:

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

816 Appears in the documents in the following form: `timestamp`.

817 ***Top Level***

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

820 ***type***

821 General meaning:

822 A classification or categorization of information.

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

825 Appears in the documents in the following form: `type`.

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

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

829 Appears in the documents in the following form: `type`.

830 ***upper limit***

831 The upper conformance boundary for a variable.

832 Note: immediate concern or action may be required.

833 ***upper warning***

834 The upper boundary indicating increased concern and supervision may be required.

835 ***Valid Data Value***

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

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

839 **WARNING**

840 General Meaning:

841 A statement or action that indicates a possible danger, problem, or other unexpected
842 situation.

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

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

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

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

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

849 Appears in the documents in the following form: WARNING.

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

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

853 Appears in the documents in the following form: Warning.

854 ***XML Container***

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

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

859 ***XML Document***

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

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

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

865 ***XML Schema***

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

868 **3.2 MTConnect References**

869 [MTConnect Part 1.0] *MTConnect Standard Part 1.0 - Overview and Fundamentals*. Ver-
870 sion 1.8.0.

871 [MTConnect Part 2.0] *MTConnect Standard: Part 2.0 - Devices Information Model*. Ver-
872 sion 1.8.0.

873 [MTConnect Part 3.0] *MTConnect Standard: Part 3.0 - Streams Information Model*. Ver-
874 sion 1.8.0.

875 [MTConnect Part 4.0] *MTConnect Standard: Part 4.0 - Assets Information Model*. Ver-
876 sion 1.8.0.

877 [MTConnect Part 5.0] *MTConnect Standard: Part 5.0 - Interfaces*. Version 1.8.0.

878 4 MTConnect Standard

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

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

887 4.1 MTConnect Documents Organization

888 The MTConnect specification is organized into the following documents:

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

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

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

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

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

908 4.2 MTConnect Document Versioning

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

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

915 *major.minor.revision*

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

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

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

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

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

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

939 New releases of a specific document are indicated by a successively higher revision version
940 identifier.

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

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

Version M.N.R

943 **4.2.1 Document Releases**

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

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

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

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

961 4.3 MTConnect Document Naming Conventions

962 MTConnect Documents are identified as follows:

963 4.3.1 Document Title

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

MTConnect® Standard

Part #.# - Title

Version M.N.R.

965 The following keys are used to distinguish different Parts of the MTConnect Standard and
966 the version of the MTConnect Document:

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

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

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

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

971 R – Indicator of the revision of the MTConnect Document

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

MTConnect® Standard

Part 2.0 - Devices Information Model

Version 1.2.0

974 4.3.2 Electronic Document File Naming

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

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

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

980 MTC_Part_2-0_Devices_Information_Model_1-2-0.pdf

981 4.4 Document Conventions

982 Additional information regarding specific content in the MTConnect Standard is provided
983 in the sections below.

984 4.4.1 Use of **MUST**, **SHOULD**, and **MAY**

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

- 987 • The word **MUST** indicates content that is mandatory to be provided in an imple-
988 mentation where indicated.
- 989 • The word **SHOULD** indicates content that is recommended, but the exclusion of
990 which will not invalidate an implementation.
- 991 • The word **MAY** indicates content that is optional. It is up to the implementer to
992 decide if the content is relevant to an implementation.
- 993 • The word **NOT** may be added to the words **MUST** or **SHOULD** to negate the re-
994 quirement.

995 4.4.2 Text Conventions

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

999 These conventions are:

- 1000 • Standard text is provided in Times New Roman font.

- 1001 • References to documents, sections or sub-sections of a document, or figures within a
1002 document are *italicized*; e.g., *MTConnect Standard: Part 2.0 - Devices Information*
1003 *Model*.
- 1004 • Terms with a specific meaning in the MTConnect Standard will be *italicized*; e.g.,
1005 *major* indicating a version of the Standard.
- 1006 • When these same terms are used within the text without specific reference to their
1007 function within the MTConnect Standard, they will be provided as non-italicized
1008 font; e.g., *major* indicating a descriptor of another term.
- 1009 • Terms representing content of an MTConnect *semantic data model* or the protocol
1010 used in MTConnect will be provided in fixed size, Courier New font; e.g., `comp-`
1011 `onent`, `probe`, `current`.
- 1012 When these same terms are used within the text without specific reference to
1013 their function within the MTConnect Standard, they will be provided as Times New
1014 Roman font.
- 1015 • All *Valid Data Values* that are restricted to a limited or controlled vocabulary will be
1016 provided in upper case Courier New font with an `_`(underscore) separating words.
1017 For example: `ON`, `OFF`, `ACTUAL`, `COUNTER_CLOCKWISE`, etc.
- 1018 • All descriptive attributes associated with each piece of data defined in a *Response*
1019 *Document* will be provided in Courier New font and camel case font style. For
1020 example: `nativeUnits`.

1021 4.4.3 Code Line Syntax and Conventions

1022 The following conventions will be used throughout the MTConnect Documents to describe
1023 examples of software code produced by an *Agent* or commands provided to an *Agent* from
1024 a client software application.

1025 All examples are provided in fixed size Courier New font with line numbers.

1026 These conventions are:

- 1027 • XML Code examples:

Example 1: XML Code Examples

```

1028 1 <MTConnectStreams xmlns:m="urn:mtconnect.com:
1029 2   MTConnectStreams:1.1" xmlns:xsi=
1030 3   "http://www.w3.org/2001/XMLSchema-instance"
1031 4   xmlns="urn:mtconnect.com:MTConnectStreams:1.1"
```

1032 • HTTP URL examples:

1033 – http://<authority>/<path>[?<query>] When a portion of a URL is enclosed in
1034 angle brackets ("<" and ">"), that section of the URL is a place holder for
1035 specific information that will replace the term between the angle brackets.

1036 Note: The angle brackets in a URL do not relate to the angle brackets
1037 used as the `tag` elements in an XML example.

1038 – A portion of a URL that is enclosed in square brackets "[" and "]" indicates
1039 that the enclosed content is optional.

1040 – All other characters in the URL are literal.

1041 4.4.4 Semantic Data Model Content

1042 For each of the *semantic data models* defined in the MTConnect Standard, there are tables
1043 describing pieces of information provided in the data models. Each table has a column
1044 labeled *Occurrence*. *Occurrence* defines the number of times the content defined in the
1045 tables **MAY** be provided in the usage case specified.

1046 • If the *Occurrence* is 1, the content **MUST** be provided.

1047 • If the *Occurrence* is 0..1, the content **MAY** be provided and if provided, at most,
1048 only one occurrence of the content **MUST** be provided.

1049 • If the *Occurrence* is 0..*, the content **MAY** be provided and any number of occur-
1050 rences of the content **MAY** be provided.

1051 • If the *Occurrence* is 1..*, one or more occurrences of the content **MUST** be pro-
1052 vided.

1053 • If the *Occurrence* is a number, e.g., 2, exactly that number of occurrences of the
1054 content **MUST** be provided.

1055 Note: "*" indicates multiple number of occurrences and is represented by ∞ in the
1056 figures.

1057 4.4.5 Referenced Standards and Specifications

1058 Other standards and specifications may be used to describe aspects of the protocol, *data*
1059 *dictionary*, or *semantic data models* defined in the MTConnect Standard. When a spe-

1060 cific standard or specification is referenced in the MTConnect Standard, the name of the
1061 standard or specification will be provided in *italicized* font.

1062 See *Section 3 - Terminology and Conventions: Bibliography* for a complete listing of
1063 standards and specifications used or referenced in the MTConnect Standard.

1064 4.4.6 Deprecation and Deprecation Warnings

1065 When the MTConnect Institute adds new functionality to the MTConnect Standard, the
1066 new content may supersede some of the functionality of existing content or significantly
1067 enhance one of the *semantic data models*. When this occurs, existing content may no
1068 longer be valid for use in the new version of the Standard.

1069 4.4.6.1 Deprecation

1070 In cases when new content supersedes the functionality of the existing content, the original
1071 content **MUST** no longer be included in future implementations – only the new content
1072 should be used.

1073 The superseded content is identified by striking through the original content (~~original~~
1074 ~~content~~) and marking the content with the words "**DEPRECATED** in *Version M.N*".

1075 The deprecated content must remain in all future *minor* versions of the document. The
1076 content may be removed when a *major* version update is released. This provides imple-
1077 menters guidance on how to interpret data that may be provided from equipment utilizing
1078 an older version of the Standard. This content provides the information required for imple-
1079 menters to develop software applications that support backwards compatibility with older
1080 versions of the standard.

1081 A software application may be designed to be compliant with any specific *minor* version
1082 of the standard. That software application may be collecting data from many different
1083 pieces of equipment. Each of these pieces of equipment may be providing data defined
1084 by the current version or any of the previous *minor* versions of the standard. To maintain
1085 compatibility with existing pieces of equipment, software applications should be imple-
1086 mented to interpret data defined in the current release of the MTConnect Standard, as well
1087 as all deprecated content associated with earlier versions of the Standard.

1088 4.4.6.2 Deprecation Warning

1089 When new content provides improved alternatives for defining the *semantic data mod-*

1090 *els*, the MTConnect Institute may determine that the original content could possibly be
1091 deprecated in the future. When this occurs, a content will be marked with the words
1092 **"DEPRECATION WARNING "** to identify the content that may be deprecated in the
1093 future. This provides advanced notice to implementers that they should choose to utilize
1094 the improved alternatives when developing new products or software systems to avoid the
1095 possibility that the original content may be deprecated in a future version of the Standard.

1096 4.5 Backwards Compatibility

1097 MTConnect Documents with a different *major* version identifier represent a significant
1098 change in the *Base Functional Structure* of the MTConnect Standard. This means that
1099 the schema or protocol defined by the Standard may have changed in ways that will re-
1100 quire software applications to change how they request and/or interpret data received from
1101 an *Agent*. Software applications should be fully version aware since no assumption of
1102 backwards compatibility should be assumed at the time of a *major* revision change to the
1103 MTConnect Standard.

1104 The MTConnect Institute strives to maintain version compatibility through all *minor* re-
1105 visions of the MTConnect Standard. New *minor* versions may introduce extensions to
1106 existing *semantic data models*, extend the protocol used to communicate to the *Agent*,
1107 and/or add new *semantic data models* to extend the functionality of the Standard. Client
1108 software applications may be designed to be compliant with any specific *minor* version
1109 of the MTConnect Standard. Additionally, software applications should be capable of in-
1110 terpreting information from an *Agent* providing data based upon a lower *minor* version
1111 identifier. It should also be capable of interpreting information from an *Agent* providing
1112 data based upon a higher *minor* version identifier of the MTConnect Standard than the
1113 version supported by the client, even though the client may ignore or not be capable of
1114 interpreting the extended content provided by the *Agent*.

1115 A *revision* version of any MTConnect Document provides only editorial changes requiring
1116 no changes to an *Agent* or a client application.

1117 5 MTConnect Fundamentals

1118 The MTConnect Standard defines the functionality of an *Agent*. In an MTConnect instal-
 1119 lation, pieces of equipment publish information to an *Agent*. Client software applications
 1120 request information from the *Agent* using a communications protocol. Based on the spe-
 1121 cific information that the client software application has requested from the *Agent*, the
 1122 *Agent* forms a *Response Document* based upon one of the *semantic data models* defined
 1123 in the MTConnect Standard and then transmits that document to the client software appli-
 1124 cation.

1125 *Figure 2* illustrates the architecture of a typical MTConnect installation.

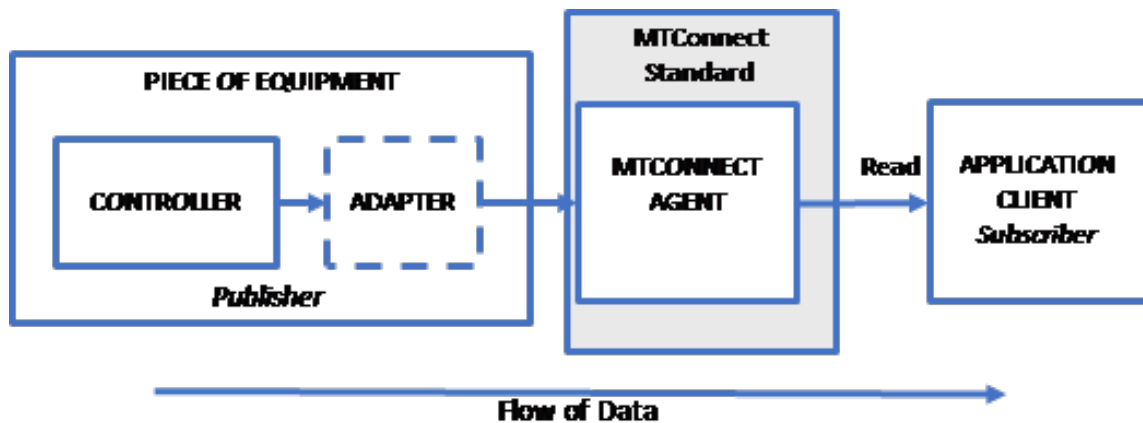


Figure 2: MTConnect Architecture Model

1126 Note: In each implementation of a communication system based on the MTConnect
 1127 Standard, there **MUST** be a schema defined that encodes the rules and termi-
 1128 nology defined for each of the *semantic data models*. These schemas **MAY** be
 1129 used by client software applications to validate the content and structure of the
 1130 *Response Documents* published by an *Agent*.

1131 5.1 Agent

1132 An *Agent* is the centerpiece of an MTConnect implementation. It provides two primary
 1133 functions:

- 1134 • Organizes and manages individual pieces of information published by one or more
 1135 pieces of equipment.

- 1136 • Publishes that information in the form of a *Response Document* to client software
1137 applications.

1138 The MTConnect Standard addresses the behavior of an *Agent* and the structure and mean-
1139 ing of the data published by an *Agent*. It is the responsibility of the implementer of an
1140 *Agent* to determine the means by which the behavior is achieved for a specific *Agent*.

1141 An *Agent* is software that may be installed as part of a piece of equipment or it may be
1142 installed separately. When installed separately, an *Agent* may receive information from
1143 one or more pieces of equipment.

1144 Some pieces of equipment may be able to communicate directly to an *Agent*. Other pieces
1145 of equipment may require an *Adapter* to transform the information provided by the equip-
1146 ment into a form that can be sent to an *Agent*. In either case, the method of transmitting
1147 information from the piece of equipment to an *Agent* is implementation dependent and is
1148 not addressed as part of the MTConnect Standard.

1149 One function of an *Agent* is to store information that it receives from a piece of equipment
1150 in an organized manner. A second function of an *Agent* is to receive *Requests* for informa-
1151 tion from one or many client software applications and then respond to those *Requests* by
1152 publishing a *Response Document* that contains the requested information.

1153 There are three types of information stored by an *Agent* that **MAY** be published in a *Re-*
1154 *sponse Document*. These are:

- 1155 • *Equipment Metadata* defines the *Structural Elements* that represent the physical and
1156 logical parts and sub-parts of each piece of equipment that can publish data to the
1157 *Agent*, the relationships between those parts and sub-parts, and the *Data Entities*
1158 associated with each of those *Structural Elements*. This *Equipment Metadata* is
1159 provided in an *MTConnectDevices Response Document*. See *MTConnect Standard:*
1160 *Part 2.0 - Devices Information Model* for more information on *Equipment Metadata*.

- 1161 • *Streaming Data* provides the values published by pieces of equipment for the *Data*
1162 *Entities* defined by the *Equipment Metadata*. *Streaming Data* is provided in an *MT-*
1163 *ConnectStreams Response Document*. See *MTConnect Standard: Part 2.0 - Devices*
1164 *Information Model* for more information on *Streaming Data*.

- 1165 • *MTConnect Assets* represent information used in a manufacturing operation that is
1166 commonly shared amongst multiple pieces of equipment and/or software applica-
1167 tions. *MTConnect Assets* are provided in an *MTConnectAssets Response Document*.
1168 See *MTConnect Standard: Part 4.0 - Assets Information Model* for more informa-
1169 tion on *MTConnect Assets*.

1170 The exchange between an *Agent* and a client software application is a *Request* and *Re-*
 1171 *sponse* information exchange mechanism. See *Section 5.4 - Request/Response Information*
 1172 *Exchange* for details on this *Request/Response* information exchange mechanism.

1173 5.1.1 Instance of an Agent

1174 As described above, an *Agent* collects and organizes values published by pieces of equip-
 1175 ment. As with any piece of software, an *Agent* may be periodically restarted. When an
 1176 *Agent* restarts, it **MUST** indicate to client software applications whether the information
 1177 available in the *buffer* represents a completely new set of data or if the *buffer* includes data
 1178 that had been collected prior to the restart of the *Agent*.

1179 Any time an *Agent* is restarted and begins to collect a completely new set of *Streaming*
 1180 *Data*, that set of data is referred to as an *instance* of the *Agent*. The *Agent* **MUST** maintain
 1181 a piece of information called `instanceId` that represents the specific *instance* of the
 1182 *Agent*.

1183 `instanceId` is represented by a 64-bit integer. The `instanceId` **MAY** be imple-
 1184 mented using any mechanism that will guarantee that the value for `instanceId` will be
 1185 unique each time the *Agent* begins collecting a new set of data.

1186 When an *Agent* is restarted and it provides a method to recover all, or some portion, of
 1187 the data that was stored in the *buffer* before it stopped operating, the *Agent* **MUST** use the
 1188 same `instanceId` that was defined prior to the restart.

1189 5.1.2 Storage of Equipment Metadata for a Piece of Equipment

1190 An *Agent* **MUST** be capable of publishing *Equipment Metadata* for each piece of equip-
 1191 ment that publishes information through the *Agent*. *Equipment Metadata* is typically a
 1192 static file defining the *Structural Elements* associated with each piece of equipment re-
 1193 porting information through the *Agent* and the *Data Entities* that can be associated with
 1194 each of these *Structural Elements*. See details on *Structural Elements* and *Data Entities* in
 1195 *MTConnect Standard: Part 2.0 - Devices Information Model*.

1196 The MTConnect Standard does not define the mechanism to be used by an *Agent* to ac-
 1197 quire, maintain, or store the *Equipment Metadata*. This mechanism **MUST** be defined as
 1198 part of the implementation of a specific *Agent*.

1199 5.1.3 Storage of Streaming Data

1200 *Streaming Data* that is published from a piece(s) of equipment to an *Agent* is stored by the
 1201 *Agent* based upon the sequence upon which each piece of data is received. As described
 1202 below, the order in which data is stored by the *Agent* is one of the factors that determines
 1203 the data that may be included in a specific *MTConnectStreams Response Document*.

1204 5.1.3.1 Management of Streaming Data Storage

1205 An *Agent* stores a fixed amount of data. The amount of data stored by an *Agent* is depen-
 1206 dent upon the implementation of a specific *Agent*. The examples below demonstrate how
 1207 discrete pieces of data received from pieces of equipment are stored.

1208 The method for storing *Streaming Data* in an *Agent* can be thought of as a tube that can
 1209 hold a finite set of balls. Each ball represents the occurrence of a *Data Entity* published
 1210 by a piece of equipment. This data is pushed in one end of the tube until there is no more
 1211 room for additional balls. At that point, any new data inserted will push the oldest data out
 1212 the back of the tube. The data in the tube will continue to shift in this manner as new data
 1213 is received.

1214 This tube is referred to as a *buffer* in an *Agent*.

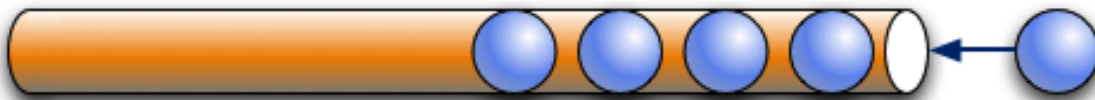


Figure 3: Data Storage in Buffer

1215 In *Figure 4*, the maximum number of *Data Entities* that can be stored in the *buffer* of
 1216 the *Agent* is 8. The maximum number of *Data Entities* that can be stored in the *buffer* is
 1217 represented by a value called `bufferSize`. This example illustrates that when the *buffer*
 1218 fills up, the oldest piece of data falls out the other end.

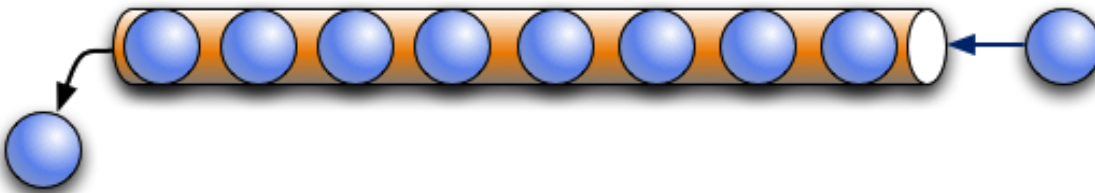


Figure 4: First In First Out Buffer Management

1219 This process constrains the memory storage requirements for an *Agent* to a fixed maximum
 1220 size since the MTConnect Standard only requires an *Agent* to store a finite number of
 1221 pieces of data.

1222 As an implementation guideline, the *buffer* **SHOULD** be sized large enough to provide
 1223 storage for a reasonable amount of information received from all pieces of equipment
 1224 that are publishing information to that *Agent*. The implementer should also consider the
 1225 impact of a temporary loss of communications between a client software application and
 1226 an *Agent* when determining the size for the *buffer*. A larger *buffer* will allow a client
 1227 software application more time to reconnect to an *Agent* without losing data.

1228 5.1.3.2 Sequence Numbers

1229 In an *Agent*, each occurrence of a *Data Entity* in the *buffer* will be assigned a monotoni-
 1230 cally increasing *sequence number* as it is inserted into the *buffer*. The *sequence number*
 1231 is a 64-bit integer and the values assigned as *sequence numbers* will never wrap around or
 1232 be exhausted; at least within the next 100,000 years based on the size of a 64-bit number.

1233 *sequence number* is the primary key identifier used to manage and locate a specific piece
 1234 of data in an *Agent*. The *sequence number* associated with each *Data Entity* reported by
 1235 an *Agent* is identified with an attribute called `sequence`.

1236 The *sequence number* for each piece of data **MUST** be unique for an instance of an *Agent*
 1237 (see Section 5.1.1 - *Instance of an Agent* for information on *instances* of an *Agent*). If data
 1238 is received from more than one piece of equipment, the *sequence numbers* are based on
 1239 the order in which the data is received regardless of which piece of equipment produced
 1240 that data. The *sequence number* **MUST** be a monotonically increasing number that spans
 1241 all pieces of equipment publishing data to an *Agent*. This allows for multiple pieces of
 1242 equipment to publish data through a single *Agent* with no *sequence number* collisions and
 1243 unnecessary protocol complexity.

1244 The *sequence number* **MUST** be reset to one (1) each time an *Agent* is restarted and begins
 1245 to collect a fresh set of data; i.e., each time `instanceId` is changed.

1246 *Figure 5* demonstrates the relationship between `instanceId` and `sequence` when an
 1247 *Agent* stops and restarts and begins collecting a new set of data. In this case, the `in-`
 1248 `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

1249 *Figure 6* also shows two additional pieces of information defined for an *Agent*:

- 1250 • `firstSequence` – the oldest piece of data contained in the *buffer*; i.e., the next
- 1251 piece of data to be moved out of the *buffer*
- 1252 • `lastSequence` – the newest data added to the *buffer*

1253 `firstSequence` and `lastSequence` provide guidance to a software application iden-

1254 tifying the range of data available that may be requested from an *Agent*.

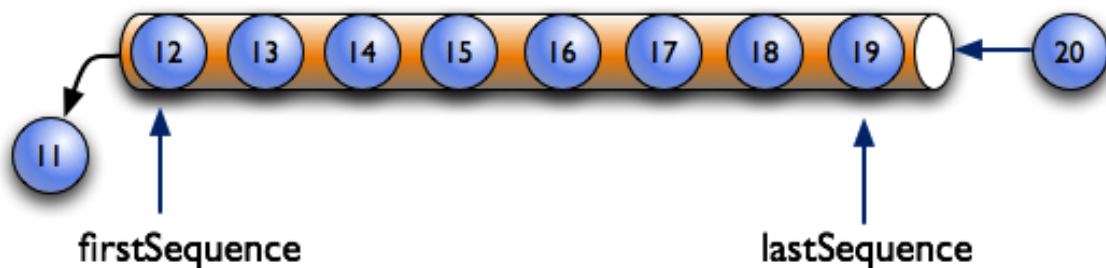


Figure 6: Identifying the range of data with `firstSequence` and `lastSequence`

1255 When a client software application requests data from an *Agent*, it can specify both the

1256 *sequence number* of the first piece of data (`from`) that **MUST** be included in the *Response*

1257 *Document* and the total number (*count*) of pieces of data that **SHOULD** be included in
 1258 that document.

1259 In *Figure 7*, the request specifies that the data to be returned starts at *sequence number* 15
 1260 (*from*) and includes a total of three items (*count*).

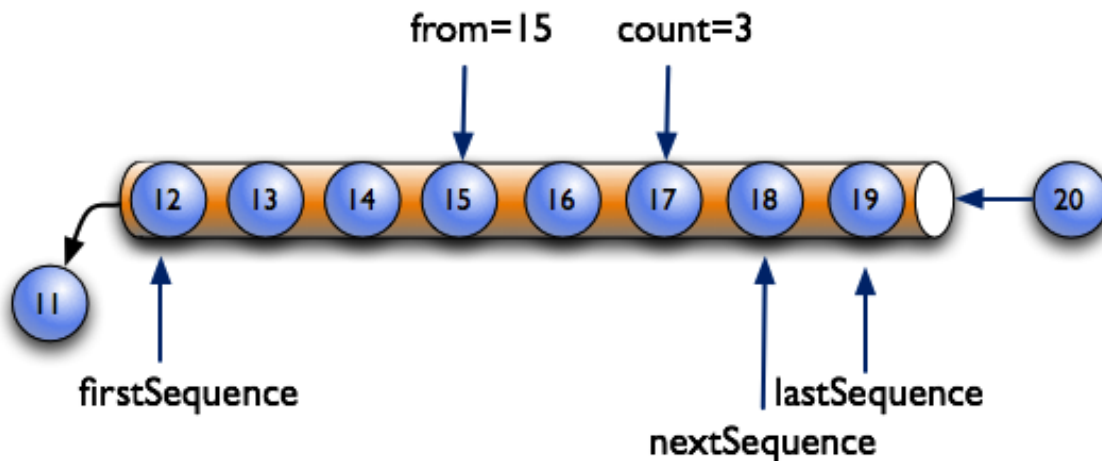


Figure 7: Identifying the range of data with *from* and *count*

1261 Once a *Response* to a *Request* has been completed, the value of *nextSequence* will be
 1262 established. *nextSequence* is the *sequence number* of the next piece of data available
 1263 in the *buffer*. In the example in *Figure 7*, the next *sequence number* (*nextSequence*)
 1264 will be 18.

1265 As shown in *Figure 8*, the combination of *from* and *count* defined by the *Request*
 1266 indicates a *sequence number* for data that is beyond that which is currently in the *buffer*.
 1267 In this case, *nextSequence* is set to a value of *lastSequence* + 1.

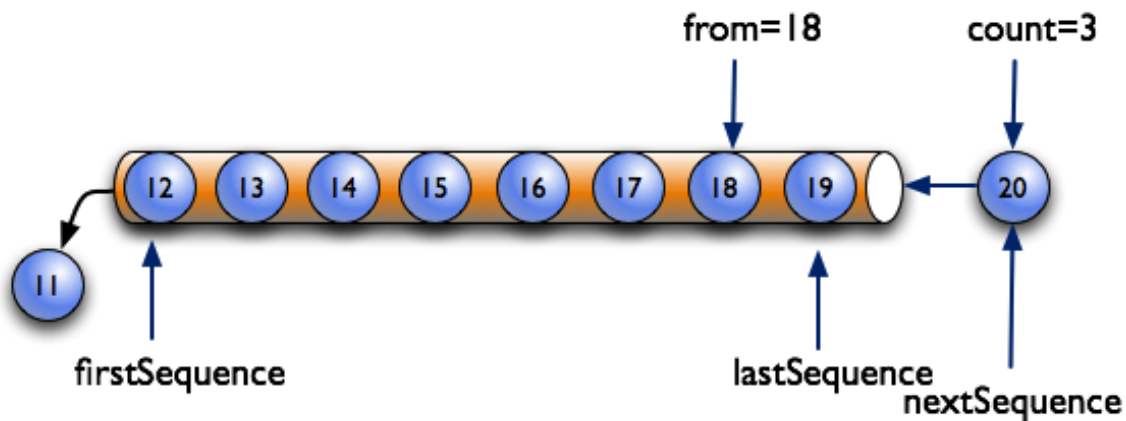


Figure 8: Identifying the range of data with nextSequence and lastSequence

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

1283 The storage mechanism for the data, the internal representation of the data, and the imple-
 1284 mentation of the *Agent* itself is not part of the MTConnect Standard. The implementer can
 1285 choose both the amount of data to be stored in the *Agent* and the mechanism for how the
 1286 data is stored. The only requirement is that an *Agent* publish the *Response Documents* in
 1287 the required format.

1288 5.1.3.4 Time Stamp

1289 Each piece of equipment that publishes information to an *Agent* **SHOULD** provide a time
 1290 stamp indicating when each piece of information was measured or determined. If no time
 1291 stamp is provided, the *Agent* **MUST** provide a time stamp for the information based upon
 1292 when that information was received at the *Agent*.

1293 The `timestamp` associated with each piece of information is reported by an *Agent* as
 1294 `timestamp`. `timestamp` **MUST** be reported in UTC (Coordinated Universal Time)
 1295 format; e.g., "2010-04-01T21:22:43Z".

1296 Note: Z refers to UTC/GMT time, not local time.

1297 Client software applications should use the value of `timestamp` reported for each piece
 1298 of information as the means for ordering when pieces of information were generated as
 1299 opposed to using `sequence` for this purpose.

1300 Note: It is assumed that `timestamp` provides the best available estimate of the time
 1301 that the value(s) for the published information was measured or determined.

1302 If two pieces of information are measured or determined at the exact same time, they
 1303 **MUST** be reported with the same value for `timestamp`. Likewise, all information that
 1304 is recorded in the *buffer* with the same value for `timestamp` should be interpreted as
 1305 having been recorded at the same point in time; even if that data was published by more
 1306 than one piece of equipment.

1307 5.1.3.5 Recording Occurrences of Streaming Data

1308 An *Agent* **MUST** record data in the *buffer* each time the value for that specific piece of data
 1309 changes. If a piece of equipment publishes multiple occurrences of a piece of data with
 1310 the same value, the *Agent* **MUST NOT** record multiple occurrence for that *Data Entity*.

1311 Note: There is one exception to this rule. Some *Data Entities* may be defined with a
 1312 `representation` attribute value of `DISCRETE` (**DEPRECATED** in *Ver-*
 1313 *sion 1.5*) (See *Section 7.2.2.12 of MTConnect Standard: Part 2.0 - Devices*
 1314 *Information Model* for details on `representation`.) In this case, each oc-
 1315 currence of the data represents a new and unique piece of information. The
 1316 *Agent* **MUST** then record each occurrence of the *Data Entity* that is published
 1317 by a piece of equipment.

1318 The value for each piece of information reported by an *Agent* must be considered by a
 1319 client software application to be valid until such a time that another occurrence of that
 1320 piece of information is published by the *Agent*.

1321 5.1.3.6 Maintaining Last Value for Data Entities

1322 An *Agent* **MUST** retain a copy of the last available value associated with each *Data Entity*
 1323 known to the *Agent*; even if an occurrence of that *Data Entity* is no longer in the *buffer*.
 1324 This function allows an *Agent* to provide a software application a view of the last known
 1325 value for each *Data Entity* associated with a piece of equipment.

1326 The *Agent* **MUST** also retain a copy of the last value associated with each *Data Entity* that
 1327 has flowed out of the *buffer*. This function allows an *Agent* to provide a software applica-
 1328 tion a view of the last known value for each *Data Entity* associated with a *Current Request*
 1329 with an `at` parameter in the `query` portion of its *HTTP Request Line* (See *Section 8.3.2 -*
 1330 *Current Request Implemented Using HTTP* for details on *Current Request*).

1331 **5.1.3.7 Unavailability of Data**

1332 An *Agent* **MUST** maintain a list of *Data Entities* that **MAY** be published by each piece of
 1333 equipment providing information to the *Agent*. This list of *Data Entities* is derived from
 1334 the *Equipment Metadata* stored in the *Agent* for each piece of equipment.

1335 Each time an *Agent* is restarted, the *Agent* **MUST** place an occurrence of every *Data*
 1336 *Entity* in the *buffer*. The value reported for each of these *Data Entities* **MUST** be set to
 1337 UNAVAILABLE and the `timestamp` for each **MUST** be set to the time that the last piece
 1338 of data was collected by the *Agent* prior to the restart.

1339 If at any time an *Agent* loses communications with a piece of equipment, or the *Agent* is
 1340 unable to determine a valid value for all, or any portion, of the *Data Entities* published by
 1341 a piece of equipment, the *Agent* **MUST** place an occurrence of each of these *Data Entities*
 1342 in the *buffer* with its value set to UNAVAILABLE. This signifies that the value is currently
 1343 indeterminate and no assumptions of a valid value for the data is possible.

1344 Since an *Agent* may receive information from multiple pieces of equipment, it **MUST**
 1345 consider the validity of the data from each of these pieces of equipment independently.

1346 There is one exception to the rules above. Any *Data Entity* that is constrained to a constant
 1347 data value **MUST** be reported with the constant value and the *Agent* **MUST NOT** set the
 1348 value of that *Data Entity* to UNAVAILABLE.

1349 Note: The schema for the *Devices Information Model* (defined in *MTConnect Stan-*
 1350 *dard: Part 2.0 - Devices Information Model*) defines how the value reported for
 1351 an individual piece of data may be constrained to one or more specific values.

1352 **5.1.3.8 Persistence and Recovery**

1353 The implementer of an *Agent* must decide on a strategy regarding the storage of *Streaming*
 1354 *Data* in the *buffer* of the *Agent*.

1355 In the simplest form, an *Agent* can hold the *buffer* information in volatile memory where
 1356 no data is persisted when the *Agent* is stopped. In this case, the *Agent* **MUST** update the
 1357 value for `instanceId` when the *Agent* restarts to indicate that the *Agent* has begun to
 1358 collect a new set of data.

1359 If the implementation of an *Agent* provides a method of persisting and restoring all or
 1360 a portion of the information in the *buffer* of the *Agent* (*sequence numbers*, *time stamps*,
 1361 *identify*, and *values*), the *Agent* **MUST NOT** change the value of the `instanceId` when
 1362 the *Agent* restarts. This will indicate to a client software application that it does not need to
 1363 reset the value for `nextSequence` when it requests the next set of data from the *Agent*.

1364 When an implementer chooses to provide a method to persist the information in an *Agent*,
 1365 they may choose to store as much data as is practical in a recoverable storage system. Such
 1366 a method may also include the ability to store historical information that has previously
 1367 been pushed out of the *buffer*.

1368 **5.1.3.9 Heartbeat**

1369 An *Agent* **MUST** provide a function that indicates to a client application that the HTTP
 1370 connection is still viable during times when there is no new data available to report in a
 1371 *Response Document*. This function is defined as *heartbeat*.

1372 *heartbeat* represents the amount of time after a *Response Document* has been published
 1373 until a new *Response Document* **MUST** be published, even when no new data is available.

1374 See Section 8.3.3.2 - *Query Portion of the HTTP Request Line for a Sample Request* for
 1375 more details on configuring the *heartbeat* function.

1376 **5.1.3.10 Data Sets**

1377 See *MTConnect Standard: Part 3.0 - Streams Information Model Section Part 3: DataItem*
 1378 *with representation of DATA_SET* for management of *Data Sets*.

1379 **5.1.4 Storage of Documents for MTConnect Assets**

1380 An *Agent* also stores information associated with *MTConnect Assets*.

1381 When a piece of equipment publishes a document that represents information associated
 1382 with an *MTConnect Asset*, an *Agent* stores that document in a *buffer*. This *buffer* is called
 1383 the *assets buffer*. The document is called an *Asset Document*.

1384 The *assets buffer* **MUST** be a separate *buffer* from the one where the *Streaming Data* is
 1385 stored.

1386 The *Asset Document* that is published by the piece of equipment **MUST** be organized
 1387 based upon one of the applicable *Asset Information Models* defined in one of the Parts 4.x
 1388 of the MTConnect Standard.

1389 An *Agent* will only retain a limited number of *Asset Documents* in the *assets buffer*. The
 1390 *assets buffer* functions similar to the *buffer* for *Streaming Data*; i.e., when the *assets buffer*
 1391 is full, the oldest *Asset Document* is pushed from the *buffer*.

1392 *Figure 10* demonstrates the oldest *Asset Document* being pushed from the *assets buffer*
 1393 when a new *Asset Document* is added and the *assets buffer* is full:

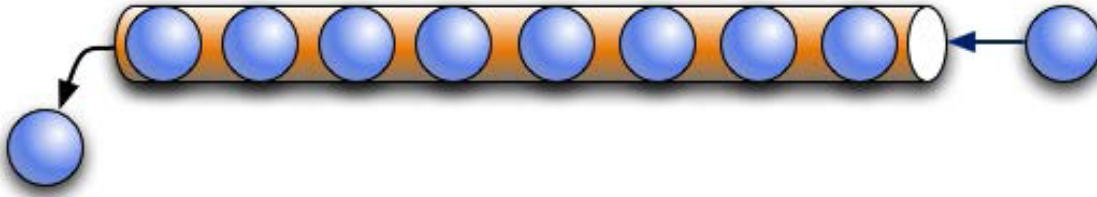


Figure 10: First In First Out Asset Buffer Management

1394 Within an *Agent*, the management of *Asset Documents* behave like a key/value storage in a
 1395 database. In the case of *MTConnect Assets*, the key is an identifier for an Asset (see details
 1396 on `assetId` in *MTConnect Standard: Part 4.0 - Assets Information Model*) and the value
 1397 is the *Asset Document* that was published by the piece of equipment.

1398 *Figure 11* demonstrates the relationship between the key (`assetId`) and the stored *Asset*
 1399 *Documents*:

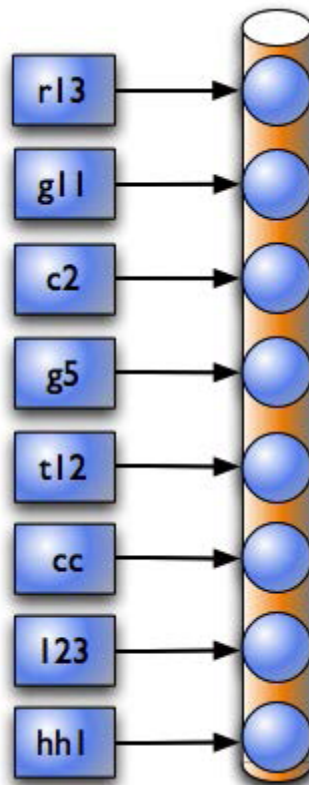


Figure 11: Relationship between `assetId` and stored Asset documents

1400 Note: The key (`assetId`) is independent of the order of the *Asset Documents* stored
 1401 in the *assets buffer*.

1402 When an *Agent* receives a new *Asset Document* representing an *MTConnect Asset*, it must
 1403 determine whether this document represents an *MTConnect Asset* that is not currently
 1404 represented in the *assets buffer* or if the document represents new information for an *MT-*
 1405 *Connect Asset* that is already represented in the *assets buffer*. When a new *Asset Document*
 1406 is received, one of the following **MUST** occur:

- 1407 • If the *Asset Document* represents an *MTConnect Asset* that is not currently repre-
 1408 sented in the *assets buffer*, the *Agent* **MUST** add the new document to the front
 1409 of the *assets buffer*. If the *assets buffer* is full, the oldest *Asset Document* will be
 1410 removed from the *assets buffer*.

- 1411 • If the *Asset Document* represents an *MTConnect Asset* that is already represented in
 1412 the *assets buffer*, the *Agent* **MUST** remove the existing *Asset Document* representing
 1413 that *MTConnect Asset* from the *assets buffer* and add the new *Asset Document* to the
 1414 front of the *assets buffer*.

1415 The *MTConnect Standard* does not specify the maximum number of *Asset Documents*
 1416 that may be stored in the *assets buffer*; that limit is determined by the implementation
 1417 of a specific *Agent*. The number of *Asset Documents* that may be stored in an *Agent* is
 1418 defined by the value for `assetBufferSize` (See *Section 6.5 - Document Header* for
 1419 more information on `assetBufferSize`). A value of 4,294,967,296 or 2^{32} can be
 1420 provided for `assetBufferSize` to indicate unlimited storage.

1421 There is no requirement for an *Agent* to provide persistence for the *Asset Documents* stored
 1422 in the *assets buffer*. If an *Agent* should fail, all *Asset Documents* stored in the *assets buffer*
 1423 **MAY** be lost. It is the responsibility of the implementer to determine if *Asset Documents*
 1424 stored in an *Agent* may be restored or if those *Asset Documents* are retained by some other
 1425 software application.

1426 Additional details on how an *Agent* organizes and manages information associated with
 1427 *MTConnect Assets* are provided in *MTConnect Standard: Part 4.0 - Assets Information*
 1428 *Model*.

1429 5.2 Response Documents

1430 *Response Documents* are electronic documents generated and published by an *Agent* in
 1431 response to a *Request* for data.

1432 The *Response Documents* defined in the MTConnect Standard are:

- 1433 • *MTConnectDevices Response Document*: An electronic document that contains the
1434 information published by an *Agent* describing the data that can be published by one
1435 or more piece(s) of equipment. The structure of the *MTConnectDevices Response*
1436 *Document* document is based upon the requirements defined by the *Devices Infor-*
1437 *mation Model*. See *MTConnect Standard: Part 2.0 - Devices Information Model* for
1438 details on this information model.
- 1439 • *MTConnectStreams Response Document*: An electronic document that contains the
1440 information published by an *Agent* that contains the data that is published by one
1441 or more piece(s) of equipment. The structure of the *MTConnectStreams Response*
1442 *Document* document is based upon the requirements defined by the *Streams Infor-*
1443 *mation Model*. See *MTConnect Standard: Part 3.0 - Streams Information Model* for
1444 details on this information model.
- 1445 • *MTConnectAssets Response Document*: An electronic document that contains the
1446 information published by an *Agent* that **MAY** include one or more *Asset Documents*.
1447 The structure of the *MTConnectAssets Response Document* document is based upon
1448 the requirements defined by the *Asset Information Models*. See *MTConnect Stan-*
1449 *dard: Part 4.0 - Assets Information Model* for details on this information model.
- 1450 • *MTConnectErrors Response Document*: An electronic document that contains the
1451 information provided by an *Agent* when an error has occurred when trying to re-
1452 spond to a *Request* for data. The structure of the *MTConnectErrors Response Doc-*
1453 *ument* is based upon the requirements defined by the *Error Information Model*. See
1454 *Section 9 - Error Information Model* of this document for details on this information
1455 model.

1456 *Response Documents* may be represented by any document format supported by an *Agent*.
1457 No matter what document format is used to structure these documents, the requirements
1458 for representing the data and other information contained in those documents **MUST** ad-
1459 here to the requirements defined in the *Information Models* associated with each document.

1460 5.2.1 XML Documents

1461 XML is currently the only document format supported by the MTConnect Standard for
1462 encoding *Response Documents*. Other document formats may be supported in the future.

1463 Since XML is the document format supported by the MTConnect Standard for encoding
1464 documents, all examples demonstrating the structure of the *Response Documents* provided

1465 throughout the MTConnect Standard are based on XML. These documents will be referred
1466 to as *MTConnect XML Documents* or *XML Documents*.

1467 *Section 6 - XML Representation of Response Documents* defines how each document is
1468 structured as an *XML Document*.

1469 5.3 Semantic Data Models

1470 A *semantic data model* is a software engineering method for representing data where the
1471 context and the meaning of the data is constrained and fully defined.

1472 Each of the *semantic data models* defined by the MTConnect Standard include:

- 1473 • The types of information that may be published by a piece of equipment,
- 1474 • The meaning of that information and units of measure, if applicable,
- 1475 • Structural information that defines how different pieces of information relate to each
1476 other, and
- 1477 • Structural information that defines how the information relates to where the infor-
1478 mation was measured or generated by the piece of equipment.

1479 As described previously, the content of the *Response Documents* provided by an *Agent* are
1480 each defined by a specific *semantic data model*. The details for the *semantic data model*
1481 used to define each of the *Response Documents* are detail as follows:

- 1482 • *MTConnectDevices Response Document: MTConnect Standard: Part 2.0 - Devices*
1483 *Information Model*.
- 1484 • *MTConnectStreams Response Document: MTConnect Standard: Part 3.0 - Streams*
1485 *Information Model*.
- 1486 • *MTConnectAssets Response Document: MTConnect Standard: Part 4.0 - Assets*
1487 *Information Model* and its sub-Parts.
- 1488 • *MTConnectErrors Response Document: MTConnect Standard Part 1.0 - Overview*
1489 *and Fundamentals, Section 9 - Error Information Model*.

1490 Without semantics, a single piece of data does not convey any relevant meaning to a person
1491 or a client software application. However, when that piece of data is paired with some

1492 semantic context, the data inherits significantly more meaning. The data can then be more
 1493 completely interpreted by a client software application without human intervention.

1494 The MTConnect *semantic data models* allows the information published by a piece of
 1495 equipment to be transmitted to client software application with a full definition of the
 1496 meaning of that information and in full context defining how that information relates to
 1497 the piece of equipment that measured or generated the information.

1498 5.4 Request/Response Information Exchange

1499 The transfer of information between an *Agent* and a client software application is based
 1500 on a *Request/Response* information exchange approach. A client software application
 1501 requests specific information from an *Agent*. An *Agent* responds to the *Request* by pub-
 1502 lishing a *Response Document*.

1503 In normal operation, there are four types of *MTConnect Requests* that can be issued by
 1504 a client software application that will result in different *Responses* by an *Agent*. These
 1505 *Requests* are:

- 1506 • *Probe Request*– A client software application requests the *Equipment Metadata* for
 1507 each piece of equipment that **MAY** publish information through an *Agent*. The *Agent*
 1508 publishes a *MTConnectDevices Response Document* that contains the requested in-
 1509 formation. A *Probe Request* is represented by the term `probe` in a *Request* from a
 1510 client software application.
- 1511 • *Current Request* – A client software application requests the current value for each
 1512 of the data types that have been published from a piece(s) of equipment to an *Agent*.
 1513 The *Agent* publishes a *MTConnectStreams Response Document* that contains the
 1514 requested information. A *Current Request* is represented by the term `current` in
 1515 a *Request* from a client software application.
- 1516 • *Sample Request* – A client software application requests a series of data values from
 1517 the *buffer* in an *Agent* by specifying a range of *sequence numbers* representing that
 1518 data. The *Agent* publishes a *MTConnectStreams Response Document* that contains
 1519 the requested information. A *Sample Request* is represented by the term `sample` in
 1520 a *Request* from a client software application.
- 1521 • *Asset Request* – A client software application requests information related to *MT-*
 1522 *Connect Assets* that has been published to an *Agent*. The *Agent* publishes an *MT-*
 1523 *ConnectAssets Response Document* that contains the requested information. An *As-*
 1524 *set Request* is represented by the term `asset` in a *Request* from a client software
 1525 application.

1526 Note: If an *Agent* is unable to respond to the request for information or the re-
 1527 quest includes invalid information, the *Agent* will publish an *MTConnectErrors*
 1528 *Response Document*. See *Section 9 - Error Information Model* for information
 1529 regarding *Error Information Model*

1530 The specific format for the *Request* for information from an *Agent* will depend on the
 1531 *Protocol* implemented as part of the *Request/Response* information exchange mechanism
 1532 deployed in a specific implementation. See *Section 7 - Protocol and Messaging, Protocol*
 1533 for details on implementing the *Request/Response* information exchange.

1534 Also, the specific format for the *Response Documents* may also be implementation de-
 1535 pendent. See *Section 6 - XML Representation of Response Documents* for details on the
 1536 format for the *Response Documents* encoded with XML.

1537 **5.5 Accessing Information from an Agent**

1538 Each of the *Requests* defined for the *Request/Response* information exchange requires
 1539 an *Agent* to respond with a specific view of the information stored by the *Agent*. The
 1540 following describes the relationships between the information stored by an *Agent* and the
 1541 contents of the *Response Documents*.

1542 **5.5.1 Accessing Equipment Metadata from an Agent**

1543 The *Equipment Metadata* associated with each piece of equipment that publishes infor-
 1544 mation to an *Agent* is typically static information that is maintained by the *Agent*. The
 1545 MTConnect Standard does not define how the *Agent* captures or maintains that informa-
 1546 tion. The only requirement that the MTConnect Standard places on an *Agent* regarding this
 1547 *Equipment Metadata* is that the *Agent* properly store this information and then configure
 1548 and publish a *MTConnectDevices Response Document* in response to a *Probe Request*.

1549 All issues associated with the capture and maintenance of the *Equipment Metadata* is the
 1550 responsibility of the implementer of a specific *Agent*.

1551 **5.5.2 Accessing Streaming Data from the Buffer of an Agent**

1552 There are two *Requests* defined for the *Request/Response* information exchange that re-
 1553 quire an *Agent* to provide different views of the information stored in the *buffer* of the
 1554 *Agent*. These *Requests* are *current* and *sample*.

1555 The example in *Figure 12* demonstrates how an *Agent* interprets the information stored
 1556 in the *buffer* to provide the content that is published in different versions of the *MTConnectStreams Response Document* based on the specific *Request* that is issued by a client
 1557 software application.
 1558

1559 In this example, an *Agent* with a *buffer* that can hold up to eight (8) *Data Entities*; i.e., the
 1560 value for `bufferSize` is 8. This *Agent* is collecting information for two pieces of data
 1561 – `Pos` representing a position and `Line` representing a line of logic or commands in a
 1562 control program.

1563 In this *buffer*, the value for `firstSequence` is 12 and the value for `lastSequence`
 1564 is 19. There are five (5) different values for `Pos` and three (3) different values for `Line`.

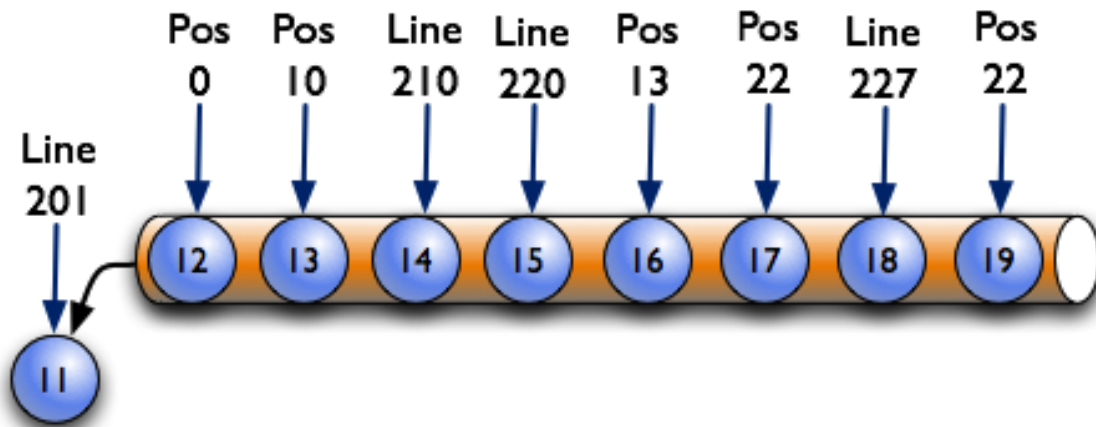


Figure 12: Example Buffer

1565 If an *Agent* receives a *Sample Request* from a client software application, the *Agent* **MUST**
 1566 publish an *MTConnectStreams Response Document* that contains a range of data values.
 1567 The range of values are defined by the `from` and `count` parameters that must be included
 1568 as part of the *Sample Request*. If the value of `from` is 14 and the value of `count` is 5,
 1569 the *Agent* **MUST** publish an *MTConnectStreams Response Document* that includes five
 1570 (5) pieces of data represented by *sequence numbers* 14, 15, 16, 17, and 18 – three (3)
 1571 occurrences of `Line` and two (2) occurrences of `Pos`. In this case, `nextSequence` will
 1572 also be returned with a value of 19.

1573 Likewise, if the same *Agent* receives a *Current Request* from a client software application,
 1574 the *Agent* **MUST** publish an *MTConnectStreams Response Document* that contains the
 1575 most current information available for each of the types of data that is being published to
 1576 the *Agent*. In this case, the specific data that **MUST** be represented in the *MTConnect-*
 1577 *Streams Response Document* is `Pos` with a value of 22 and a *sequence number* of 19 and
 1578 `Line` with a value of 227 and a *sequence number* of 18.

1579 There is also a derivation of the *Current Request* that will cause an *Agent* to publish an
 1580 *MTConnectStreams Response Document* that contains a set of data relative to a specific
 1581 sequence number. The *Current Request* **MAY** include an additional parameter called *at*.
 1582 When the *at* parameter, along with an *instanceId*, is included as part of a *Current Re-*
 1583 *quest*, an *Agent* **MUST** publish an *MTConnectStreams Response Document* that contains
 1584 the most current information available for each of the types of *Data Entities* that are being
 1585 published to the *Agent* that occur immediately at or before the *sequence number* specified
 1586 with the *at* parameter.

1587 For example, if the *Request* is *current?at=15*, an *Agent* **MUST** publish a *MTCon-*
 1588 *nectStreams Response Document* that contains the most current information available for
 1589 each of the *Data Entities* that are stored in the *buffer* of the *Agent* with a *sequence number*
 1590 of 15 or lower. In this case, the specific data that **MUST** be represented in the *MTCon-*
 1591 *nectStreams Response Document* is *Pos* with a value of 10 and a *sequence number* of 13
 1592 and *Line* with a value of 220 and a *sequence number* of 15.

1593 If a *current Request* is received for a *sequence number* of 11 or lower, an *Agent* **MUST**
 1594 return an *OUT_OF_RANGE MTConnectErrors Response Document*. The same *HTTP Er-*
 1595 *ror Message* **MUST** be given if a *sequence number* is requested that is greater than the
 1596 end of the *buffer*. See *Section 9 - Error Information Model* for more information on *MT-*
 1597 *ConnectErrors Response Document*.

1598 5.5.3 Accessing MTConnect Assets Information from an Agent

1599 When an *Agent* receives an *Asset Request*, the *Agent* **MUST** publish an *MTConnectAs-*
 1600 *sets* document that contains information regarding the *Asset Documents* that are stored
 1601 in the *Agent*.

1602 See *MTConnect Standard: Part 4.0 - Assets Information Model* for details on *MTConnect*
 1603 *Assets*, *Asset Requests*, and the *MTConnectAssets Response Document*.

1604 6 XML Representation of Response Documents

1605 As defined in *Section 5.2.1 - XML Documents*, XML is currently the only language sup-
 1606 ported by the MTConnect Standard for encoding *Response Documents*.

1607 *Response Documents* must be valid and conform to the *schema* defined in the *semantic*
 1608 *data model* defined for that document. The *schema* for each *Response Document* **MUST**
 1609 be updated to correlate to a specific version of the MTConnect Standard. Versions, within
 1610 a *major* version, of the MTConnect Standard will be defined in such a way to best maintain
 1611 backwards compatibility of the *semantic data models* through all *minor* revisions of the
 1612 Standard. However, new *minor* versions may introduce extensions or enhancements to
 1613 existing *semantic data models*.

1614 To be valid, a *Response Document* must be well-formed; meaning that, amongst other
 1615 things, each element has the required XML *start-tag* and *end-tag* and that the document
 1616 does not contain any illegal characters. The validation of the document may also include
 1617 a determination that required elements and attributes are present, they only occur in the
 1618 appropriate location in the document, and they appear only the correct number of times.
 1619 If the document is not well-formed, it may be rejected by a client software application.
 1620 The *semantic data model* defined for each *Response Document* also specifies the elements
 1621 and *Child Elements* that may appear in a document. XML elements may contain *Child*
 1622 *Elements*, CDATA, or both. The *semantic data model* also defines the number of times
 1623 each element and *Child Element* may appear in the document.

1624 Each *Response Document* encoded using XML consists of the following primary sections:

- 1625 • XML Declaration
- 1626 • Root Element
- 1627 • Schema and Namespace Declaration
- 1628 • Document Header
- 1629 • Document Body

1630 The following will provide details defining how each of the *Response Documents* are en-
 1631 coded using XML.

1632 Note: See *Section 3 - Terminology and Conventions* for the definition of XML related
 1633 terms used in the MTConnect Standard.

1634 6.1 Fundamentals of Using XML to Encode Response Documents

1635 The MTConnect Standard follows industry conventions for formatting the elements and
1636 attributes included in an XML document. The general guidelines are as follows:

- 1637 • All element names **MUST** be specified in Pascal case (first letter of each word is
1638 capitalized). For example: <PowerSupply/>.
- 1639 • The name for an attribute **MUST** be Camel case; similar to Pascal case, but the first
1640 letter will be lower case. For example: <MyElement nativeName="bob"/>
1641 where MyElement is the *Element Name* and nativeName is an attribute.
- 1642 • All CDATA values that are defined with a limited or controlled vocabulary **MUST**
1643 be in upper case with an _ (underscore) separating words. For example: ON, OFF,
1644 ACTUAL, and COUNTER_CLOCKWISE.
- 1645 • The values provided for a date and/or a time **MUST** follow the W3C ISO 8601
1646 format with an arbitrary number of decimals representing fractions of a second.
1647 Refer to the following specification for details on the format for dates and times:
1648 <http://www.w3.org/TR/NOTE-datetime>.
- 1649 The format for the value describing a date and a time will be
1650 YYYY-MM-DDThh:mm:ss.ffff. An example would be: 2017-01-13T13:01.213415Z.
- 1651 Note: Z refers to UTC/GMT time, not local time.
- 1652 The accuracy and number of decimals representing fractions of a second for a `time-`
1653 `stamp` **MUST** be determined by the capabilities of the piece of equipment publishing
1654 information to an *Agent*. All time values **MUST** be provided in UTC (GMT).
- 1655 • XML element names **MUST** be spelled out and abbreviations are not permitted. See
1656 the exclusion below regarding the use of the suffix `Ref`.
- 1657 • XML attribute names **SHOULD** be spelled out and abbreviations **SHOULD** be
1658 avoided. The exception to this rule is the use of `id` when associated with an identi-
1659 fier. See the exclusion below regarding the use of the suffix `Ref`.
- 1660 • The abbreviation `Ref` for *Reference* is permitted as a suffix to element names of
1661 either a *Structural Element* or a *Data Entity* to provide an efficient method to asso-
1662 ciate information defined in another location in a *Data Model* without duplicating
1663 that original data or structure. See *Section 4.8* in *MTConnect Standard: Part 2.0 -*
1664 *Devices Information Model* for more information on *Reference*.

1665 6.2 XML Declaration

1666 The first section of a *Response Document* encoded with XML **SHOULD** be the *XML*
1667 *Declaration*. The declaration is a single element.

1668 An example of an *XML Declaration* would be:

Example 2: Example of xml declaration

```
1669 1 <?xml version="1.0" encoding="UTF-8"?>
```

1670 This element provides information regarding how the XML document is encoded and the
1671 character type used for that encoding. See the W3C website for more details on the XML
1672 declaration.

1673 6.3 Root Element

1674 Every *Response Document* **MUST** contain only one root element. The MTConnect Stan-
1675 dard defines MTConnectDevices, MTConnectStreams, MTConnectAssets, and
1676 MTConnectError as *Root Elements*.

1677 The *Root Element* specifies a specific *Response Document* and appears at the top of the
1678 document immediately following the *XML Declaration*.

1679 6.3.1 MTConnectDevices Root Element

1680 MTConnectDevices is the *Root Element* for the *MTConnectDevices Response Docu-*
1681 *ment*.

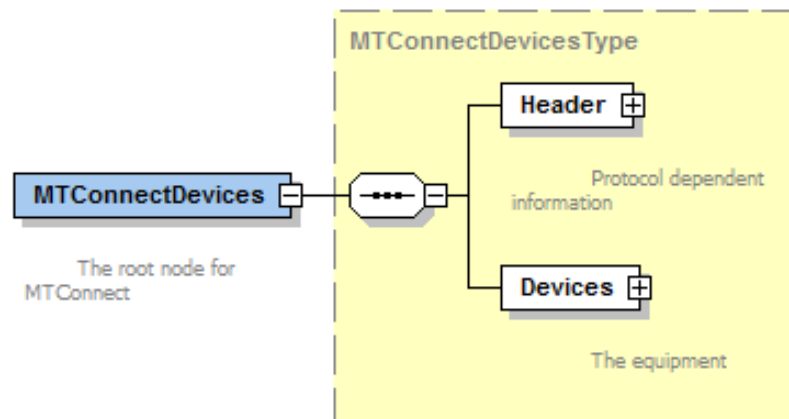


Figure 13: MTConnectDevices Structure

1682 MTConnectDevices **MUST** contain two *Child Elements* - Header and Devices.
 1683 Details for Header are defined in Section 6.5 - Document Header.

1684 Devices is an XML container that represents the *Document Body* for an *MTConnectDe-*
 1685 *vices Response Document* – see Section 6.6 - Document Body. Details for the *semantic*
 1686 *data model* describing the contents for Devices are defined in *MTConnect Standard:*
 1687 *Part 2.0 - Devices Information Model*.

1688 MTConnectDevices also has a number of attributes. These attributes are defined in
 1689 Section 6.4 - Schema and Namespace Declaration.

1690 6.3.1.1 MTConnectDevices Elements

1691 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

1692 6.3.2 MTConnectStreams Root Element

1693 MTConnectStreams is the *Root Element* for the *MTConnectStreams Response Document*.
 1694

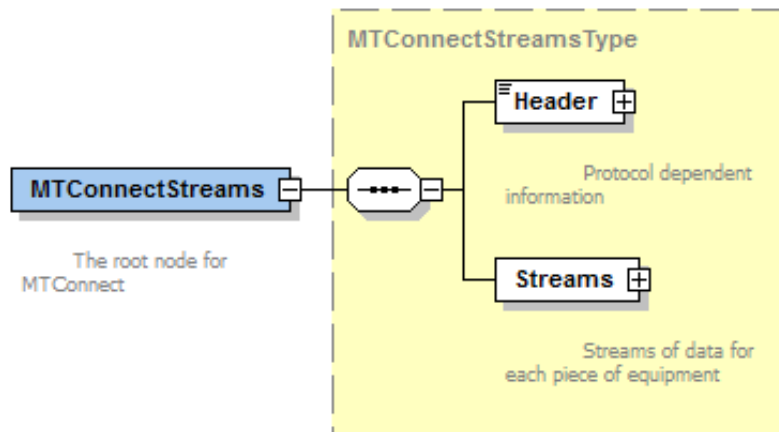


Figure 14: MTConnectStreams Structure

1695 MTConnectStreams **MUST** contain two *Child Elements* - Header and Streams.

1696 Details for Header are defined in *Section 6.5 - Document Header*.

1697 Streams is an XML container that represents the *Document Body* for a *MTConnect-*
 1698 *Streams Response Document* – see *Section 6.6 - Document Body*. Details for the *semantic*
 1699 *data model* describing the contents for Streams are defined in *MTConnect Standard:*
 1700 *Part 3.0 - Streams Information Model*.

1701 MTConnectStreams also has a number of attributes. These attributes are defined in
 1702 *Section 6.4 - Schema and Namespace Declaration*.

1703 6.3.2.1 MTConnectStreams Elements

1704 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

1705 6.3.3 MTConnectAssets Root Element

1706 MTConnectAssets is the *Root Element* for the *MTConnectAssets Response Document*.

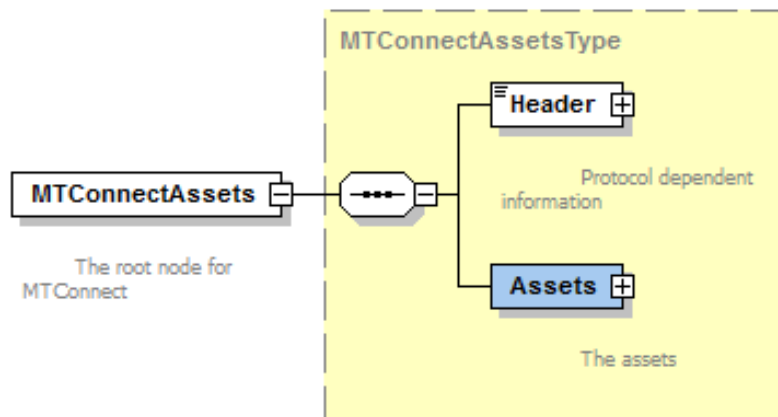


Figure 15: MTConnectAssets Structure

1707 MTConnectAssets **MUST** contain two *Child Elements* - Header and Assets.

1708 Details for Header are defined in *Section 6.5 - Document Header*.

1709 Assets is an XML container that represents the *Document Body* for an *MTConnectAssets*
 1710 *Response Document* – see *Section 6.6 - Document Body*. Details for the *semantic data*
 1711 *model* describing the contents for Assets are defined in *MTConnect Standard: Part 4.0*
 1712 *- Assets Information Model*.

1713 MTConnectAssets also has a number of attributes. These attributes are defined in
 1714 *Section 6.4 - Schema and Namespace Declaration*.

1715 **6.3.3.1 MTConnectAssets Elements**

1716 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

1717 **6.3.4 MTConnectError Root Element**

1718 MTConnectError is the *Root Element* for the *MTConnectErrors Response Document*.

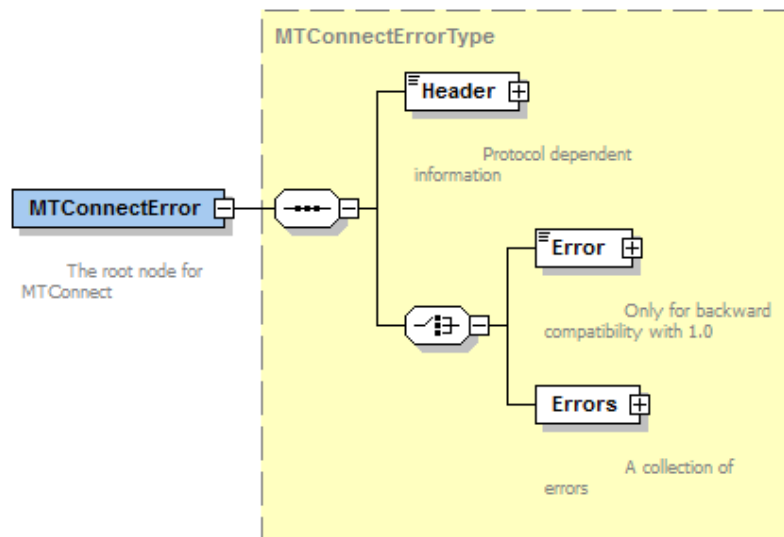


Figure 16: MTConnectError Structure

1719 MTConnectError **MUST** contain two *Child Elements* - Header and Errors.

1720 Note: When compatibility with *Version 1.0.1* and earlier of the MTConnect Standard
 1721 is required for an implementation, the *MTConnectErrors Response Document*
 1722 contains only a single *Error Data Entity* and the *Errors Child Element*
 1723 **MUST NOT** appear in the document.

1724 Details for Header are defined in *Section 6.5 - Document Header*.

1725 Errors is an XML container that represents the *Document Body* for an *MTConnectErrors*
 1726 *Response Document* – See *Section 6.6 - Document Body*. Details for the *semantic data*
 1727 *model* describing the contents for Errors are defined in *Section 9 - Error Information*
 1728 *Model*.

1729 MTConnectError also has a number of attributes. These attributes are defined in *Sec-*
 1730 *tion 6.4 - Schema and Namespace Declaration*.

1731 6.3.4.1 MTConnectError Elements

1732 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

1733 6.4 Schema and Namespace Declaration

1734 XML provides standard methods for declaring the *schema* and *namespace* associated with
 1735 a document encoded by XML. The declaration of the *schema* and *namespace* for MTCon-
 1736 nect *Response Documents* **MUST** be structured as attributes in the *Root Element* of the
 1737 document. XML defines these attributes as pseudo-attributes since they provide additional
 1738 information for the entire document and not just specifically for the *Root Element* itself.

1739 Note: If a *Response Document* contains sections that utilize different *schemas* and/or
 1740 *namespaces*, additional pseudo-attributes should appear in the document as de-
 1741 clared using standard conventions as defined by W3C.

1742 For further information on declarations refer to *Appendix C*.

1743 6.5 Document Header

1744 The *Document Header* is an XML container in an *MTConnect Response Document* that
 1745 provides information from an *Agent* defining version information, storage capacity, and
 1746 parameters associated with the data management within the *Agent*. This XML element is
 1747 called `Header`.

1748 `Header` **MUST** be the first XML element following the *Root Element* of any *Response*
 1749 *Document*. The `Header` XML element **MUST NOT** contain any *Child Elements*.

1750 The content of the `Header` element will be different for each type of *Response Document*.

1751 6.5.1 Header for MTConnectDevices

1752 The `Header` element for an *MTConnectDevices Response Document* defines information
 1753 regarding the creation of the document and the data storage capability of the *Agent* that
 1754 generated the document.

1755 6.5.1.1 XML Schema Structure for Header for MTConnectDevices

1756 The *XML Schema* in *Figure 17* represents the structure of the `Header` XML element that
 1757 **MUST** be provided for an *MTConnectDevices Response Document*.

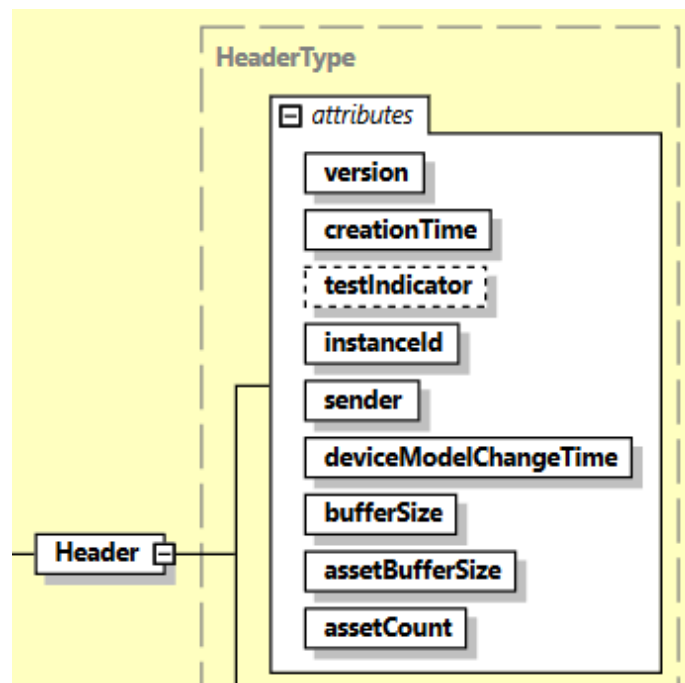


Figure 17: Header Schema Diagram for MTConnectDevices

1758 6.5.1.2 Attributes for Header for MTConnectDevices

1759 *Table 5* defines the attributes that may be used to provide additional information in the
 1760 `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 testIndicator 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 functioning in a test mode. <p>If testIndicator is not specified, the value for testIndicator MUST be interpreted to be false.</p> <p>testIndicator 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 instanceId MUST be a unique unsigned 64-bit integer.</p> <p>The value for instanceId 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>instanceId 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
deviceModelChangeTime	A timestamp in 8601 format of the last update of the <i>Device</i> information for any device.	1

1761 *Example 3* is an example of a Header XML element for an *MTConnectDevices Response*
 1762 *Document*:

Example 3: Example of Header XML Element for MTConnectDevices

```

1763 1 <Header creationTime="2017-02-16T16:44:27Z"
1764 2   sender="MyAgent" instanceId="1268463594"

```

```

1765 3    bufferSize="131072" version="1.4.0.10"
1766 4    assetCount="54" assetBufferSize="1024"/>

```

1767 6.5.2 Header for MTConnectStreams

1768 The `Header` element for an *MTConnectStreams Response Document* defines informa-
 1769 tion regarding the creation of the document and additional information necessary for an
 1770 application to interact and retrieve data from the *Agent*.

1771 6.5.2.1 XML Schema Structure for Header for MTConnectStreams

1772 The *XML Schema* in *Figure 18* represents the structure of the `Header` XML element that
 1773 **MUST** be provided for an *MTConnectStreams Response Document*.

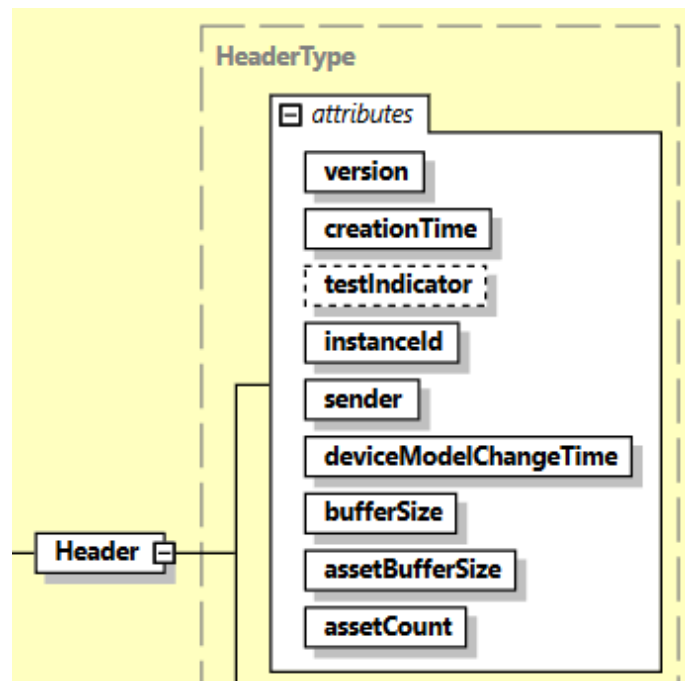


Figure 18: Header Schema Diagram for MTConnectStreams

1774 6.5.2.2 Attributes for MTConnectStreams Header

1775 *Table 6* defines the attributes that may be used to provide additional information in the
 1776 `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 <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 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

Continuation of Table 6		
Attribute	Description	Occurrence
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
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 testIndicator 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 functioning in a test mode. <p>If testIndicator is not specified, the value for testIndicator MUST be interpreted to be false.</p> <p>testIndicator is an optional attribute.</p>	0..1

Continuation of Table 6		
Attribute	Description	Occurrence
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
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

Continuation of Table 6		
Attribute	Description	Occurrence
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 bufferSize MUST be a number representing an unsigned 32-bit integer.</p> <p>bufferSize is a required attribute.</p> <p>Note 1: bufferSize 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 bufferSize.</p>	1
deviceModelChangeTime	A timestamp in 8601 format of the last update of the <i>Device</i> information for any device.	1

1777 *Example 4* is an example of a Header XML element for an *MTConnectStreams Response*
 1778 *Document*:

Example 4: Example of Header XML Element for MTConnectStreams

```

1779 1 <Header lastSequence="5430495" firstSequence="5299424"
1780 2   nextSequence="5430496" bufferSize="131072"
1781 3   version="1.4.0.12" instanceId="1579788747"
1782 4   sender="myagent" creationTime="2020-03-24T13:23:32Z"/>

```

1783 6.5.3 Header for MTConnectAssets

1784 The Header element for an *MTConnectAssets Response Document* defines information
 1785 regarding the creation of the document and the storage of *Asset Documents* in the *Agent*
 1786 that generated the document.

1787 6.5.3.1 XML Schema Structure for Header for MTConnectAssets

1788 The *XML Schema* in *Figure 19* represents the structure of the `Header` XML element that
 1789 **MUST** be provided for an *MTConnectAssets Response Document*.

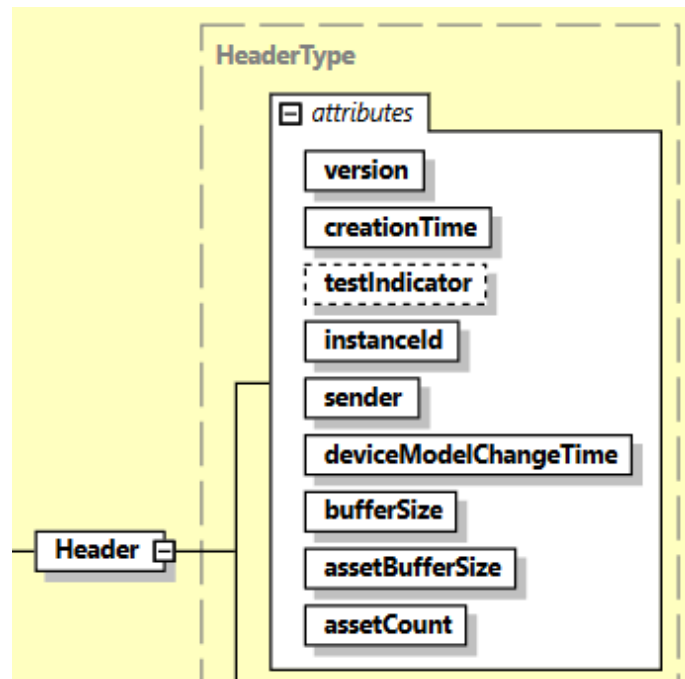


Figure 19: Header Schema Diagram for MTConnectAssets

1790 6.5.3.2 Attributes for Header for MTConnectAssets

1791 *Table 7* defines the attributes that may be used to provide additional information in the
 1792 `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 <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 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 testIndicator 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 functioning in a test mode. <p>If testIndicator is not specified, the value for testIndicator MUST be interpreted to be false.</p> <p>testIndicator 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 instanceId MUST be a unique unsigned 64-bit integer.</p> <p>The value for instanceId 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>instanceId 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

Continuation of Table 7		
Attribute	Description	Occurrence
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 <i>creationTime</i> that the <i>Agent</i> published the <i>Response Document</i>.</p> <p>The value reported for <i>assetCount</i> MUST be a number representing an unsigned 32-bit integer and MUST NOT be larger than the value reported for <i>assetBufferSize</i>.</p> <p><i>assetCount</i> is a required attribute.</p>	1
deviceModelChangeTime	A timestamp in 8601 format of the last update of the <i>Device</i> information for any device.	1

1793 *Example 5* is an example of a *Header XML* element for an *MTConnectAssets Response Document*:

Example 5: Example of Header XML Element for MTConnectAssets

```

1795 1 <Header creationTime="2017-02-16T16:44:27Z"
1796 2   sender="MyAgent" instanceId="1268463594"
1797 3   version="1.4.0.10" assetCount="54"
1798 4   assetBufferSize="1024"/>

```

1799 6.5.4 Header for MTConnectError

1800 The *Header* element for an *MTConnectErrors Response Document* defines information
 1801 regarding the creation of the document and the data storage capability of the *Agent* that
 1802 generated the document.

1803 6.5.4.1 XML Schema Structure for Header for MTConnectError

1804 The *XML Schema* in *Figure 20* represents the structure of the *Header XML* element that
 1805 **MUST** be provided for an *MTConnectErrors Response Document*.

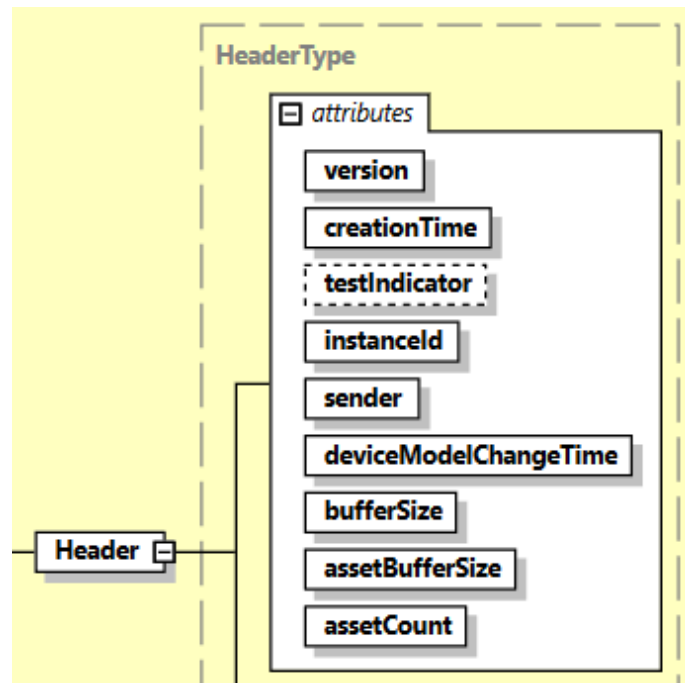


Figure 20: Header Schema Diagram for MTConnectError

1806 **6.5.4.2 Attributes for Header for MTConnectError**

1807 *Table 8* defines the attributes that may be used to provide additional information in the
 1808 `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 <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 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 testIndicator 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 functioning in a test mode. <p>If testIndicator is not specified, the value for testIndicator MUST be interpreted to be false.</p> <p>testIndicator 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 instanceId MUST be a unique unsigned 64-bit integer.</p> <p>The value for instanceId 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>instanceId 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
deviceModelChangeTime	<p>A timestamp in 8601 format of the last update of the <i>Device</i> information for any device.</p>	1

1809 *Example 6* is an example of a `Header` XML element for an *MTConnectErrors Response*
 1810 *Document*:

Example 6: Example of Header XML Element for MTConnectError

```

1811 1 <Header creationTime="2017-02-16T16:44:27Z"
1812 2   sender="MyAgent" instanceId="1268463594"
1813 3   bufferSize="131072" version="1.4.0.10"/>

```

1814 6.6 Document Body

1815 The *Document Body* contains the information that is published by an *Agent* in response
 1816 to a *Request* from a client software application. Each *Response Document* has a different
 1817 XML element that represents the *Document Body*.

1818 The structure of the content of the XML element representing the *Document Body* is de-
 1819 fined by the *semantic data models* defined for each *Response Document*.

1820 *Table 9* defines the relationship between each of the *Response Documents*, the XML ele-
 1821 ment that represents the *Document Body* for each document, and the *semantic data model*
 1822 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>

Continuation of Table 9		
Response Document	XML Element for Document Body	Semantic Data Model
<i>MTConnectErrors Response Document</i>	<p>Errors</p> <p>Note: Errors MUST NOT be used when backwards compatibility with MTConnect Standard Version 1.0.1 and earlier is required.</p>	<i>MTConnect Standard Part 1.0 - Overview and Fundamentals</i>

1823 6.7 Extensibility

1824 MTConnect is an extensible standard, which means that implementers **MAY** extend the
 1825 *Data Models* defined in the various sections of the MTConnect Standard to include in-
 1826 formation required for a specific implementation. When these *Data Models* are encoded
 1827 using XML, the methods for extending these *Data Models* are defined by the rules estab-
 1828 lished for extending any XML schema (see the W3C website for more details on extending
 1829 XML data models).

1830 The following are typical extensions that **MAY** be considered in the MTConnect *Data*
 1831 *Models*:

- 1832 • Additional `type` and `subType` values for *Data Entities*.
- 1833 • Additional *Structural Elements* as containers.
- 1834 • Additional Composition elements.
- 1835 • New *Asset* types that are sub-typed from the abstract *Asset* type.
- 1836 • *Child Elements* that may be added to specific XML elements contained within the
 1837 *MTConnect Information Models*. These extended elements **MUST** be identified in
 1838 a separate *namespace*.

1839 When extending an MTConnect *Data Model*, there are some basic rules restricting changes
1840 to the MTConnect *Data Models*.

1841 When extending an MTConnect *Data Model*, an implementer:

- 1842 • **MUST NOT** add new value for category for *Data Entities*,
- 1843 • **MUST NOT** add new *Root Elements*,
- 1844 • **SHOULD NOT** add new *Top Level Components*, and
- 1845 • **MUST NOT** add any new attributes or include any sub-elements to Composi-
1846 tion.

1847 Note: Throughout the documents additional information is provided where
1848 extensibility may be acceptable or unacceptable to maintain compliance with
1849 the MTConnect Standard.

1850 When a *schema* representing a *Data Model* is extended, the *schema* and *namespace* dec-
1851 laration at the beginning of the corresponding *Response Document* **MUST** be updated to
1852 reflect the new *schema* and *namespace* so that a client software application can properly
1853 validate the *Response Document*.

1854 An XML example of a *schema* and *namespace* declaration, including an extended *schema*
1855 and *namespace*, is shown in *Example 7*:

Example 7: Example of extended schema and namespace in declaration

```
1856 1 <?xml version="1.0" encoding="UTF-8"?>
1857 2 <MTConnectDevices
1858 3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
1859 4   xmlns="urn:mtconnect.org:MTConnectDevices:1.3"
1860 5   xmlns:m="urn:mtconnect.org:MTConnectDevices:1.3"
1861 6   xmlns:x="urn:MyLocation:MyFile:MyVersion"
1862 7   xsi:schemaLocation="urn:MyLocation:MyFile:MyVersion
1863 8   /schemas/MyFileName.xsd" />
```

1864 In this example:

- 1865 • `xmlns:x` is added in Line 6 to identify the *XML Schema* instance for the extended
1866 *schema*. *Element Names* identified with an "x" prefix are associated with this spe-
1867 cific *XML Schema* instance.

1868 Note: The "x" prefix **MAY** be replaced with any prefix that the implementer
1869 chooses for identifying the extended *schema* and *namespace*.

- 1870 • `xsi:schemaLocation` is modified in Line 7 to associate the *namespace* URN
1871 with the URL specifying the location of *schema* file.
 - 1872 • `MyLocation`, `MyFile`, `MyVersion`, and `MyFileName` in Lines 6 and 7 **MUST**
1873 be replaced by the actual name, version, and location of the extended *schema*.
- 1874 When an extended *schema* is implemented, each *Structural Element*, *Data Entity*, and
1875 *MTConnect Asset* defined in the extended *schema* **MUST** be identified in each respective
1876 *Response Document* by adding a prefix to the XML *Element Name* associated with that
1877 *Structural Element*, *Data Entity*, or *MTConnect Asset*. The prefix identifies the *schema*
1878 and *namespace* where that XML Element is defined.

1879 7 Protocol and Messaging

1880 An *Agent* performs two *major* communications tasks. It collects information from pieces
 1881 of equipment and it publishes *MTConnect Response Documents* in response to *Requests*
 1882 from client software applications.

1883 The *MTConnect Standard* does not address the method used by an *Agent* to collect in-
 1884 formation from a piece of equipment. The relationship between the *Agent* and a piece of
 1885 equipment is implementation dependent. The *Agent* may be fully integrated into the piece
 1886 of equipment or the *Agent* may be independent of the piece of equipment. Implementation
 1887 of the relationship between a piece of equipment and an *Agent* is the responsibility of the
 1888 supplier of the piece of equipment and/or the implementer of the *Agent*.

1889 The communications mechanism between an *Agent* and a client software application re-
 1890 quires the following primary components:

- 1891 • *Physical Connection*: The network transmission technologies that physically inter-
 1892 connect an *Agent* and a client software application. Examples of a *Physical Con-*
 1893 *nection* would be an Ethernet network or a wireless connection.
- 1894 • *Transport Protocol*: A set of capabilities that provide the rules and procedures used
 1895 to transport information between an *Agent* and a client software application through
 1896 a *Physical Connection*.
- 1897 • *Application Programming Interface*: The *Request* and *Response* interactions that
 1898 occur between an *Agent* and a client software application.
- 1899 • *Message*: The content of the information that is exchanged. The *Message* includes
 1900 both the content of the *MTConnect Response Document* and any additional informa-
 1901 tion required for the client software application to interpret the *Response Document*.

1902 Note: The *Physical Connections*, *Transport Protocols*, and *Application Pro-*
 1903 *gramming Interface* supported by an *Agent* are independent of the *Message* it-
 1904 self; i.e., the information contained in the *MTConnect Response Documents* is
 1905 not changed based on the methods used to transport those documents to a client
 1906 software application.

1907 An *Agent* **MAY** support multiple methods for communicating with client software ap-
 1908 plications. The *MTConnect Standard* specifies one methodology for communicating that
 1909 **MUST** be supported by every *Agent*. This methodology is a REST, which defines a state-
 1910 less, client-server communications architecture. This REST interface is the architectural
 1911 pattern that specifies the exchange of information between an *Agent* and a client software

1912 application. REST dictates that a server has no responsibility for tracking or coordinating
1913 with a client software application regarding which information or how much information
1914 the client software application may request from a server. This removes the burden for
1915 a server to keep track of client sessions. An *Agent* **MUST** be implemented as a server
1916 supporting the RESTful interface.

1917 8 HTTP Messaging Supported by an Agent

1918 This section describes the application of *HTTP Messaging* applied to a REST interface that
 1919 **MUST** be supported by an *Agent* to realize the *MTConnect Request/Response* information
 1920 exchange functionality.

1921 8.1 REST Interface

1922 An *Agent* **MUST** provide a REST interface that supports HTTP version 1.0 to commu-
 1923 nicate with client applications. This interface **MUST** support HTTP (RFC7230) and use
 1924 URIs (RFC3986) to identify specific information requested from an *Agent*. HTTP is most
 1925 often implemented on top of the Transmission Control Protocol (TCP) that provides an
 1926 ordered byte stream of data and the Internet Protocol (IP) that provides unified address-
 1927 ing and routing between computers. However, additional interfaces to an *Agent* may be
 1928 implemented in conjunction with any other communications technologies.

1929 The REST interface supports an *Application Programming Interface* (API) that adheres
 1930 to the architectural principles of a stateless, uniform interface to retrieve data and other
 1931 information related to either pieces of equipment or *MTConnect Assets*. The API allows
 1932 for access, but not modification of data stored within the *Agent* and is nullipotent, meaning
 1933 it will not produce any side effects on the information stored in an *Agent* or the function
 1934 of the *Agent* itself.

1935 *HTTP Messaging* is comprised of two basic functions – an *HTTP Request* and an *HTTP*
 1936 *Response*. A client software application forms a *Request* for information from an *Agent*
 1937 by specifying a specific set of information using an *HTTP Request*. In response, an *Agent*
 1938 provides either an *HTTP Response* or replies with an *HTTP Error Message* as defined
 1939 below.

1940 8.2 HTTP Request

1941 The *MTConnect Standard* defines that an *Agent* **MUST** support the HTTP GET verb – no
 1942 other HTTP methods are required to be supported.

1943 An *HTTP Request* **MAY** include three sections:

- 1944 • an *HTTP Request Line*
- 1945 • *HTTP Header Fields*

- 1946 • an *HTTP Body*

1947 The MTConnect Standard defines that an *HTTP Request* issued by a client application
 1948 **SHOULD** only have two sections:

- 1949 • an *HTTP Request Line*
- 1950 • *HTTP Header Fields*

1951 The *HTTP Request Line* identifies the specific information being requested by the client
 1952 software application. If an *Agent* receives any information in an *HTTP Request* that is not
 1953 specified in the MTConnect Standard, the *Agent* **MAY** ignore it.

1954 The structure of an *HTTP Request Line* consists of the following portions:

- 1955 • *HTTP Request Method*: GET
- 1956 • *HTTP Request URL*: `http://<authority>/<path>[?<query>]`
- 1957 • *HTTP Version*: HTTP/1.0

1958 For the following discussion, the *HTTP Request URL* will only be considered since the
 1959 Method will always be GET and the MTConnect Standard only requires HTTP/1.0.

1960 **8.2.1 authority Portion of an HTTP Request Line**

1961 The *authority* portion consists of the DNS name or IP address associated with an
 1962 *Agent* and an optional TCP port number [*port*] that the *Agent* is listening to for incoming
 1963 *Requests* from client software applications. If the port number is the default Port 80, *port*
 1964 is not required.

1965 Example forms for *authority* are:

- 1966 • `http://machine/`
- 1967 • `http://machine:5000/`
- 1968 • `http://192.168.1.2:5000/`

1969 8.2.2 path Portion of an HTTP Request Line

1970 The <Path> portion of the *HTTP Request Line* has the follow segments:

- 1971 • /<name or uuid>/<request>

1972 In this portion of the *HTTP Request Line*, name or uuid designates that the information to
 1973 be returned in a *Response Document* is associated with a specific piece of equipment that
 1974 has published data to the *Agent*. See Part 2 - *Devices Information Model* for details on
 1975 name or uuid for a piece of equipment.

1976 Note: If name or uuid are not specified in the *HTTP Request Line*, an *Agent* **MUST**
 1977 return the information for all pieces of equipment that have published data to
 1978 the *Agent* in the *Response Document*.

1979 In the <Path> portion of the *HTTP Request Line*, <request> designates one of the
 1980 *Requests* defined in Section 5.4 - *Request/Response Information Exchange*. The value
 1981 for <request> **MUST** be probe, current, sample, or asset(s) representing the
 1982 *Probe Request*, *Current Request*, *Sample Request*, and *Asset Request* respectively.

1983 8.2.3 query Portion of an HTTP Request Line

1984 The [?<query>] portion of the *HTTP Request Line* designates an HTTP *Query*. *Query* is
 1985 a string of parameters that define filters used to refine the content of a *Response Document*
 1986 published in response to an *HTTP Request*.

1987 8.3 MTConnect Request/Response Information Exchange Implemented 1988 with HTTP

1989 An *Agent* **MUST** support *Probe Requests*, *Current Requests*, *Sample Requests*, and *Asset*
 1990 *Requests*.

1991 The following sections define how the *HTTP Request Line* is structured to support each of
 1992 these types of *Requests* and the information that an *Agent* **MUST** provide in response to
 1993 these *Requests*.

1994 8.3.1 Probe Request Implemented Using HTTP

1995 An *Agent* responds to a *Probe Request* with an *MTConnectDevices Response Document*
 1996 that contains the *Equipment Metadata* for pieces of equipment that are requested and cur-
 1997 rently represented in the *Agent*.

1998 There are two forms of the *Probe Request*:

1999 • The first form includes an *HTTP Request Line* that does not specify a specific path
 2000 portion (name or uuid). In response to this *Request*, the *Agent* returns an *MT-*
 2001 *ConnectDevices Response Document* with information for all pieces of equipment
 2002 represented in the *Agent*.

2003 1. `http://<authority>/probe`

2004 • The second form includes an *HTTP Request Line* that specifies a specific path por-
 2005 tion that defines either a name or uuid. In response to this *Request*, the *Agent*
 2006 returns an *MTConnectDevices Response Document* with information for only the
 2007 one piece of equipment associated with that name or uuid.

2008 1. `http://<authority>/<name or uuid>/probe`

2009 8.3.1.1 Path Portion of the HTTP Request Line for a Probe Request

2010 The following segments of path **MUST** be supported in an *HTTP Request Line* for a
 2011 *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.

2012 8.3.1.2 Query Portion of the HTTP Request Line for a Probe Request

2013 The *HTTP Request Line* for a *Probe Request* **SHOULD NOT** contain a query. If the

2014 *Request* does contain a query, the *Agent* **MUST** ignore the query.

2015 **8.3.1.3 Response to a Probe Request**

2016 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
2017 *Request*.
2018

2019 The *Response Document* returned in response to a *Probe Request* **MUST** always provide
2020 the most recent information available to an *Agent*.

2021 The *Response* **MUST** also include an *HTTP Status Code*. If problems are encountered by
2022 an *Agent* while responding to a *Probe Request*, the *Agent* **MUST** also publish an *MTConnectErrors Response Document*.
2023

2024 **8.3.1.4 HTTP Status Codes for a Probe Request**

2025 The following *HTTP Status Codes* **MUST** be supported as possible responses to a *Probe*
2026 *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 <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> or <code>INVALID_REQUEST</code> as the <code>errorCode</code> .
404	Not Found	The <i>Request</i> could not be interpreted. 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 <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 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 <code>errorCode</code>.</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 <code>errorCode</code>.</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 <code>errorCode</code>.</p>
500	Internal Server Error	<p>There was an unexpected error in the Agent while responding to a <i>Request</i>.</p> <p>The Agent MUST return a 500 <i>HTTP Status Code</i>. Also, the Agent MUST publish an <i>MTConnectErrors Response Document</i> that identifies INTERNAL_ERROR as the <code>errorCode</code>.</p>

2027 8.3.2 Current Request Implemented Using HTTP

2028 An *Agent* responds to a *Current Request* with an *MTConnectStreams Response Document*
 2029 that contains the current value of *Data Entities* associated with each piece of *Streaming*
 2030 *Data* available from the *Agent*, subject to any filtering defined in the *Request*.

2031 There are two forms of the *Current Request*:

- 2032 • The first form is given without a specific path portion (*name* or *uuid*). In response
 2033 to this *Request*, the *Agent* returns an *MTConnectStreams Response Document* with
 2034 information for all pieces of equipment represented in the *buffer* of the *Agent*.

2035 1. `http://<authority>/current[?query]`

- 2036 • The second form includes a specific path portion that defines either a *name* or *uuid*.
 2037 In response to this *Request*, the *Agent* returns an *MTConnectStreams Response Doc-*
 2038 *ument* with information for only the one piece of equipment associated with the
 2039 *name* or *uuid* defined in the *Request*.

2040 1. `http://<authority>/<name or uuid>/current[?query]`

2041 8.3.2.1 Path Portion of the HTTP Request Line for a Current Request

2042 The following segments of path **MUST** be supported for an *HTTP Request Line* for a
 2043 *Current Request*:

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

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

2044 8.3.2.2 Query Portion of the HTTP Request Line for a Current Request

2045 A *Query* may be used to more precisely define the specific information to be included
 2046 in a *Response Document*. Multiple parameters may be used in a *Query* to further refine

2047 the information to be included. When multiple parameters are provided, each parameter
 2048 is separated by an ampersand (&) character and each parameter appears only once in the
 2049 *Query*. The parameters within the *Query* may appear in any sequence.

2050 The following `query` parameters **MUST** be supported in an *HTTP Request Line* for a
 2051 *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 <code>Component</code> 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 <code>at</code> parameter references a specific <i>sequence number</i>. The value MUST be an unsigned 64-bit value.</p> <p>The <code>at</code> parameter MUST NOT be used in conjunction with the <code>interval</code> parameter since this would cause an <i>Agent</i> to repeatedly return the same data.</p> <p>If the value provided for the <code>at</code> 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> <code>errorCode</code>.</p> <p>If the value provided for the <code>at</code> parameter is either lower than the value of <code>firstSequence</code> or greater than the value of <code>lastSequence</code>, 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> <code>errorCode</code>.</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 <code>at</code> 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>
interval	<p>The <i>Agent</i> MUST continuously publish <i>Response Documents</i> when the query parameters include <code>interval</code> using the value as the period between adjacent publications.</p> <p>The <code>interval</code> value MUST be in milliseconds, and MUST be a positive integer greater than zero (0).</p> <p>The <i>Query</i> MUST NOT specify both <code>interval</code> and <code>at</code> parameters.</p>

2053 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*.

2055 The *Response* to a *Current Request* **MUST** always provide the most recent information
2056 available to an *Agent* or, when the `at` parameter is specified, the value of the data at the
2057 given *sequence number*.

2058 The *Data Entities* provided in the *MTConnectStreams Response Document* will be limited
2059 to those specified in the combination of the `path` segment of the *Current Request* and the
2060 value of the XPath defined for the `path` attribute provided in the `query` segment of that
2061 *Request*.

2062 8.3.2.4 HTTP Status Codes for a Current Request

2063 The following *HTTP Status Codes* **MUST** be supported as possible responses to a *Current*
2064 *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 <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 14		
HTTP Status Code	Code Name	Description
404	Not Found	<p>The <i>Request</i> could not be interpreted.</p> <p>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 NO_DEVICE as the <code>errorCode</code>.</p> <p>If the value of the <code>at</code> parameter was greater than the <code>lastSequence</code> or is less than the <code>firstSequence</code>, 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 OUT_OF_RANGE 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 <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 <code>errorCode</code>.</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 <code>errorCode</code>.</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 <code>errorCode</code>.</p>

Continuation of Table 14		
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 <code>errorCode</code>.</p>

2065 8.3.3 Sample Request Implemented Using HTTP

2066 An *Agent* responds to a *Sample Request* with an *MTConnectStreams Response Document*
 2067 that contains a set of values for *Data Entities* currently available for *Streaming Data* from
 2068 the *Agent*, subject to any filtering defined in the *Request*.

2069 There are two forms to the *Sample Request*:

- 2070 • The first form is given without a specific `path` portion (`name` or `uuid`). In re-
 2071 sponse to this *Request*, the *Agent* returns an *MTConnectStreams Response Docu-*
 2072 *ment* with information for all pieces of equipment represented in the *Agent*.

2073 1. `http://<authority>/sample[?query]`

- 2074 • The second form includes a specific `path` portion that defines either a `name` or
 2075 `uuid`.

2076 In response to this *Request*, the *Agent* returns an *MTConnectStreams Response Doc-*
 2077 *ument* with information for only the one piece of equipment associated with the
 2078 `name` or `uuid` defined in the *Request*.

2079 1. `http://<authority>/<name or uuid>/sample?query`

2080 8.3.3.1 Path Portion of the HTTP Request Line for a Sample Request

2081 The following segments of `path` **MUST** be supported in the *HTTP Request Line* for a
 2082 *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 <code>name</code> or <code>uuid</code> 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.

2083 8.3.3.2 Query Portion of the HTTP Request Line for a Sample Request

2084 A *Query* may be used to more precisely define the specific information to be included
 2085 in a *Response Document*. Multiple parameters may be used in a *Query* to further refine
 2086 the information to be included. When multiple parameters are provided, each parameter
 2087 is separated by an & character and each parameter appears only once in the *Query*. The
 2088 parameters within the *Query* may appear in any sequence.

2089 The following query parameters **MUST** be supported in an *HTTP Request Line* for a
 2090 *Sample Request*:

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

Query Parameters	Description
path	An XPath that defines specific information or a set of information to be included in an <i>MTConnectStreams Response Document</i> . 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> . 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> .

Continuation of Table 16	
Query Parameters	Description
from	<p>The <code>from</code> parameter designates the <i>sequence number</i> of the first <i>observation</i> in the <i>buffer</i> the Agent MUST consider publishing in the <i>Response Document</i>.</p> <p>The value of <code>from</code> MUST be an unsigned 64-bit integer.</p> <p>If <code>from</code> is zero (0), it MUST be set to the <code>firstSequence</code>, the oldest <i>observation</i> in the <i>buffer</i>.</p> <p>If <code>from</code> and <code>count</code> parameters are not given, <code>from</code> MUST default to the <code>firstSequence</code>.</p> <p>If <code>from</code> is not given and <code>count</code> parameter is given, see <code>count</code> for default behavior.</p> <p>If the <code>from</code> parameter is less than the <code>firstSequence</code> or greater than <code>lastSequence</code>, the Agent MUST return a 404 <i>HTTP Status Code</i> and MUST publish an <i>MTConnectErrors Response Document</i> with an <code>OUT_OF_RANGE</code> <code>errorCode</code>.</p> <p>If the <code>from</code> parameter is not a positive numeric value, the Agent MUST return a 400 <i>HTTP Status Code</i> and MUST publish an <i>MTConnectErrors Response Document</i> with an <code>INVALID_REQUEST</code> <code>errorCode</code>.</p>

Continuation of Table 16	
Query Parameters	Description
interval	<p>The <i>Agent</i> MUST continuously publish <i>Response Documents</i> when the query parameters include <code>interval</code> using the value as the minimum period between adjacent publications.</p> <p>The <code>interval</code> value MUST be in milliseconds, and MUST be a positive integer greater than or equal to zero (0).</p> <p>The <i>Query</i> MUST NOT specify both <code>interval</code> and <code>from</code> parameters.</p> <p>If the value for the <code>interval</code> parameter is zero (0), the <i>Agent</i> MUST publish <i>Response Documents</i> at the fastest rate possible.</p> <p>If the period between the publication of a <i>Response Document</i> and reception of <i>observations</i> exceeds the <code>interval</code>, the <i>Agent</i> MUST wait for a maximum of <code>heartbeat</code> milliseconds for <i>observations</i>. Upon the arrival of <i>observations</i>, the <i>Agent</i> MUST immediately publish a <i>Response Document</i>. When the period equals or exceeds the <code>heartbeat</code>, the <i>Agent</i> MUST publish an empty <i>Response Document</i>.</p>

Continuation of Table 16	
Query Parameters	Description
count	<p>The <code>count</code> parameter designates the maximum number of <i>observations</i> the Agent MUST publish in the <i>Response Document</i>.</p> <p>The value of <code>count</code> MUST be a signed integer.</p> <p>The <code>count</code> MUST NOT be zero (0).</p> <p>When the <code>count</code> is greater than zero (0), the <code>from</code> parameter MUST default to the <code>firstSequence</code>. The evaluation of <i>observations</i> starts at <code>from</code> and moves forward accumulating newer <i>observations</i> until the number of <i>observations</i> equals the <code>count</code> or the <i>observation</i> at <code>lastSequence</code> is considered.</p> <p>When the <code>count</code> is less than zero (0), the <code>from</code> parameter MUST default to the <code>lastSequence</code>. The evaluation of <i>observations</i> starts at <code>from</code> and moves backward accumulating older <i>observations</i> until the number of <i>observations</i> equals the absolute value of <code>count</code> or the <i>observation</i> at <code>firstSequence</code> is considered.</p> <p><code>count</code> MUST NOT be less than zero (0) when an <code>interval</code> parameter is given.</p> <p>If <code>count</code> is not provided, it MUST default to 100.</p> <p>If the absolute value of <code>count</code> is greater than the size of the <i>buffer</i> or equal to zero (0), the Agent MUST return a 404 <i>HTTP Status Code</i> and MUST publish an <i>MTConnectErrors Response Document</i> with an <code>OUT_OF_RANGE</code> <code>errorCode</code>.</p> <p>If the <code>count</code> parameter is not a numeric value, the Agent MUST return a 400 <i>HTTP Status Code</i> and MUST publish an <i>MTConnectErrors Response Document</i> with an <code>INVALID_REQUEST</code> <code>errorCode</code>.</p>

Continuation of Table 16	
Query Parameters	Description
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>

Continuation of Table 16	
Query Parameters	Description
to	<p>The to parameter specifies the sequence number of the observation in the buffer that will be the upper bound of the observations in the Response Document.</p> <ul style="list-style-type: none"> • The value of to MUST be an unsigned 64-bit integer. • The value of to MUST be greater than the firstSequence. • The value of to MUST be less than or equal to the lastSequence. • The value of to MUST be greater than from. • If to and count are given, the count parameter MUST be greater than zero. • If to and count are given, the maximum number of <i>observations</i> published in the <i>Response Document</i> MUST NOT be greater than the value of count. • If to is not given, see the from parameter for default behavior. • If the to parameter is less than the firstSequence or greater than lastSequence, the <i>Agent</i> MUST return a 404 <i>HTTP Status Code</i> and MUST publish an <i>MTConnectErrors Response Document</i> with an OUT_OF_RANGE errorCode. • If the to parameter is not a positive numeric value, the <i>Agent</i> MUST return a 400 <i>HTTP Status Code</i> and MUST publish an <i>MTConnectErrors Response Document</i> with an INVALID_REQUEST errorCode.

Continuation of Table 16	
Query Parameters	Description
t _o (continued)	<ul style="list-style-type: none"> • If the t_o parameter is less than the from parameter, the <i>Agent</i> MUST return a 400 <i>HTTP Status Code</i> and MUST publish an <i>MTConnectErrors Response Document</i> with an INVALID_REQUEST errorCode. • If the t_o parameter is given and the count parameter is less than zero, the <i>Agent</i> MUST return a 400 <i>HTTP Status Code</i> and MUST publish an <i>MTConnectErrors Response Document</i> with an INVALID_REQUEST errorCode.

2091 8.3.3.3 Response to a Sample Request

2092 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*.

2094 The *Response* to a *Sample Request* **MUST** always provide the most recent information available to an *Agent* or, when the at parameter is specified, the value of the data at the given *sequence number*.

2097 The *Data Entities* provided in the *MTConnectStreams Response Document* will be limited to those specified in the combination of the path segment of the *Sample Request* and the value of the XPath defined for the path attribute provided in the query segment of that *Request*.

2101 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 document will have an empty <Streams/> element in the MTConnectStreams document to indicate no data is available at the point in time that the *Agent* published the *Response Document*.

2106 8.3.3.4 HTTP Status Codes for a Sample Request

2107 The following *HTTP Status Codes* **MUST** be supported as possible responses to a *Sample 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>
404	Not Found	<p>The <i>Request</i> could not be interpreted.</p> <p>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 <code>NO_DEVICE</code> as the <code>errorCode</code>.</p> <p>If the value of the <code>at</code> parameter was greater than the <code>lastSequence</code> or is less than the <code>firstSequence</code>, 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 <code>OUT_OF_RANGE</code> as the <code>errorCode</code>.</p>
405	Method Not Allowed	<p>A method other than <code>GET</code> 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 <code>UNSUPPORTED</code> as the <code>errorCode</code>.</p>

Continuation of Table 17		
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 <code>errorCode</code>.</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 <code>errorCode</code>.</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 <code>errorCode</code>.</p>

2109 8.3.4 Asset Request Implemented Using HTTP

2110 An *Agent* responds to an *Asset Request* with an *MTConnectAssets Response Document*
 2111 that contains information for *MTConnect Assets* from the *Agent*, subject to any filtering
 2112 defined in the *Request*.

2113 There are multiple forms to the *Asset Request*:

- 2114 • The first form is given without a specific path portion (name or uuid). In re-
 2115 sponse to this *Request*, the *Agent* returns an *MTConnectAssets Response Document*
 2116 that contains information for all *Asset Document* represented in the *Agent*.

2117 1. `http://<authority>/assets`

- 2118 • The second form includes a specific path portion that defines the identity (as-
 2119 set_id) for one or more specific *Asset Documents*. In response to this *Request*,
 2120 the *Agent* returns an *MTConnectAssets Response Document* that contains informa-
 2121 tion for the specific Assets represented in the *Agent* and defined by each of the
 2122 asset_id values provided in the *Request*. Each asset_id is separated by a ";".
 2123 1. http://<authority>/asset/asset_id;asset_id;asset_id....

2124 Note: An *HTTP Request Line* may include combinations of path and query to
 2125 achieve the desired set of *Asset Documents* to be included in a specific *MT-*
 2126 *ConnectAssets Response Document*.

2127 8.3.4.1 Path Portion of the HTTP Request Line for an Asset Request

2128 The following segments of path **MUST** be supported in the *HTTP Request Line* for an
 2129 *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> .

2130 8.3.4.2 Query Portion of the HTTP Request Line for an Asset Request

2131 A *Query* may be used to more precisely define the specific information to be included
 2132 in a *Response Document*. Multiple parameters may be used in a *Query* to further refine
 2133 the information to be included. When multiple parameters are provided, each parameter
 2134 is separated by an & character and each parameter appears only once in the *Query*. The
 2135 parameters within the *Query* may appear in any sequence.

2136 The following query parameters **MUST** be supported in an *HTTP Request Line* for an
 2137 *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 <code>Asset</code> container and describes the highest-level element in the <i>Asset</i> hierarchy. See <i>MTConnect Standard: Part 4.0 - Assets Information Model, Section 3.2.3</i> for more information on the type of an <i>Asset</i>.</p>
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 <code>removed</code> are <code>true</code> or <code>false</code>.</p> <p>If the value of the <code>removed</code> parameter in the <code>query</code> 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 <code>query</code> 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 <code>query</code>, 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 <code>query</code>, the default value for <code>count</code> MUST be determined to be 100.</p>

2138 8.3.4.3 Response to an Asset Request

2139 The *Response* to an *Asset Request* **SHOULD** be an *MTConnectAssets Response Document*
 2140 containing information for one or more *Asset Documents* designated by the *Request*. The
 2141 *Response* to an *Asset Request* **MUST** always provide the most recent information available
 2142 to an *Agent*.

2143 The *Asset Documents* provided in the *MTConnectAssets Response Document* will be lim-

2144 ited to those specified in the combination of the `path` segment of the *Asset Request* and
 2145 the parameters provided in the `query` segment of that *Request*.

2146 If the `removed` query parameter is not provided with a value of `true`, *Asset Documents*
 2147 for *Assets* that have been marked as removed will not be provided in the response.

2148 8.3.4.4 HTTP Status Codes for a Asset Request

2149 The following *HTTP Status Codes* **MUST** be supported as possible responses to an *Asset*
 2150 *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 <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> or <code>INVALID_REQUEST</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>
404	Not Found	<p>The <i>Request</i> could not be interpreted.</p> <p>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 <code>NO_DEVICE</code> or <code>ASSET_NOT_FOUND</code> as the <code>errorCode</code>.</p>

Continuation of Table 20		
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 <code>errorCode</code>.</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 <code>errorCode</code>.</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 <code>errorCode</code>.</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 <code>errorCode</code>.</p>

2151 8.3.5 HTTP Errors

2152 When an *Agent* receives an *HTTP Request* that is incorrectly formatted or is not supported
 2153 by the *Agent*, the *Agent* **MUST** publish an *HTTP Error Message* which includes a specific

2154 status code from the tables above indicating that the *Request* could not be handled by the
2155 *Agent*.

2156 Also, if the *Agent* experiences an internal error and is unable to provide the requested
2157 *Response Document*, it **MUST** publish an *HTTP Error Message* that includes a specific
2158 status code from the table above.

2159 When an *Agent* encounters an error in interpreting or responding to an *HTTP Request*,
2160 the *Agent* **MUST** also publish an *MTConnectErrors Response Document* that provides
2161 additional details about the error. See *Section 9 - Error Information Model* for details on
2162 the *MTConnectErrors Response Document*.

2163 8.3.6 Streaming Data

2164 *HTTP Data Streaming* is a method for a server to provide a continuous stream of informa-
2165 tion in response to a single *Request* from a client software application. *Data Streaming* is
2166 a version of a *Publish/Subscribe* method of communications.

2167 When an *HTTP Request* includes an `interval <query>` parameter, an *Agent* **MUST**
2168 provide data with a minimum delay between the end of one data transmission and the
2169 beginning of the next data transmission defined by the value (in milliseconds) provided
2170 for `interval` parameter. A value of zero (0) for the `interval` parameter indicates
2171 that the *Agent* should deliver data at the highest rate possible.

2172 The format of the response **MUST** use a MIME encoded message with each section sep-
2173 arated by a MIME boundary. Each section **MUST** contain an entire *MTConnectStreams*
2174 *Response Document*.

2175 If there are no available *Data Entities* to be published after the `interval` time has
2176 elapsed, an *Agent* **MUST** wait until additional information is available to be published.
2177 If no new no new information is available to be published within the time defined by the
2178 `heartbeat` parameter, the *Agent* **MUST** then send a new section to ensure the receiver
2179 that the *Agent* is functioning correctly. In this case, the content of the *MTConnect-*
2180 *Streams* document **MUST** be empty since no data is available.

2181 For more information on MIME see IETF RFC 1521 and RFC 822.

2182 An example of the format for a *HTTP Request* that includes an `interval` parameter is:

Example 8: Example for HTTP Request with interval parameter

2183 1 `http://localhost:5000/sample?interval=1000`

2184 HTTP Response Header:

Example 9: HTTP Response header

```

2185 1 HTTP/1.1 200 OK
2186 2 Connection: close
2187 3 Date: Sat, 13 Mar 2010 08:33:37 UTC
2188 4 Status: 200 OK
2189 5 Content-Disposition: inline
2190 6 X-Runtime: 144ms
2191 7 Content-Type: multipart/x-mixed-replace;boundary=
2192 8 a8e12eced4fb871ac096a99bf9728425
2193 9 Transfer-Encoding: chunked

```

2194 Lines 1-9 in *Example 9* represent a standard header for a MIME `multipart/x-mixed-`
 2195 `replace` message. The boundary is a separator for each section of the stream. Lines 7-8
 2196 indicate this is a multipart MIME message and the boundary between sections.

2197 With streaming protocols, the `Content-length` **MUST** be omitted and `Transfer-`
 2198 `Encoding` **MUST** be set to `chunked` (line 9). See IETF RFC 7230 for a full description
 2199 of the HTTP protocol and chunked encoding.

Example 10: HTTP Response header 2

```

2200 10 --a8e12eced4fb871ac096a99bf9728425
2201 11 Content-type: text/xml
2202 12 Content-length: 887
2203 13
2204 14 <?xml version="1.0" encoding="UTF-8"?>
2205 15 <MTConnectStreams ...>...

```

2206 Each section of the document begins with a boundary preceded by two hyphens (-). The
 2207 `Content-type` and `Content-length` MIME header fields **MUST** be provided for
 2208 each section and **MUST** be followed by `<CR><LF><CR><LF>` (ASCII code for `<CR>` is
 2209 13 and `<LF>` is 10) before the XML document. The header and the `<CR><LF><CR><LF>`
 2210 **MUST NOT** be included in the computation of the content length.

2211 An *Agent* **MUST** continue to stream results until the client closes the connection. The
 2212 *Agent* **MUST NOT** stop the streaming for any other reason other than the *Agent* process
 2213 shutting down or the client application becoming unresponsive and not receiving data (as
 2214 indicated by not consuming data and the write operation blocking).

2215 8.3.6.1 Heartbeat

2216 When *Streaming Data* is requested from a *Sample Request*, an *Agent* **MUST** support a
 2217 *heartbeat* to indicate to a client application that the HTTP connection is still viable during

2218 times when there is no new data available to be published. The *heartbeat* is indicated by
 2219 an *Agent* by sending an *MTConnect Response Document* with an empty *Streams* container
 2220 (See *MTConnect Standard: Part 3.0 - Streams Information Model, Section 4.1 Streams* for
 2221 more details on the *Streams* container) to the client software application.

2222 The *heartbeat* **MUST** occur on a periodic basis given by the optional *heartbeat* query
 2223 parameter and **MUST** default to 10 seconds. An *Agent* **MUST** maintain a separate *heart-*
 2224 *beat* for each client application for which the *Agent* is responding to a *Data Streaming*
 2225 *Request*.

2226 An *Agent* **MUST** begin calculating the interval for the time-period of the *heartbeat* for
 2227 each client application immediately after a *Response Document* is published to that spe-
 2228 cific client application.

2229 The *heartbeat* remains in effect for each client software application until the *Data Stream-*
 2230 *ing Request* is terminated by either the *Agent* or the client application.

2231 8.3.7 References

2232 A *Structural Element* **MAY** include a set of *References* of the following types that **MAY**
 2233 alter the content of the *MTConnectStreams Response Documents* published in response to
 2234 a *Current Request* or a *Sample Request* as specified:

- 2235 • A *Component Reference* (*ComponentRef*) modifies the set of resulting *Data Enti-*
 2236 *ties*, limited by a path query parameter of a *Current Request* or *Sample Request*,
 2237 to include the *Data Entities* associated with the *Structural Element* whose value for
 2238 its *id* attribute matches the value provided for the *idRef* attribute of the *Compo-*
 2239 *nentRef* element. Additionally, *Data Entities* defined for any *Lower Level Struc-*
 2240 *tural Element(s)* associated with the identified *Structural Element* **MUST** also be
 2241 returned. The result is equivalent to appending `//[@id=<"idRef">]` to the path
 2242 query parameters of the *Current Request* or *Sample Request*. See *Section 8.3.2 -*
 2243 *Current Request Implemented Using HTTP* for more details on path queries.
- 2244 • A *Data Item Reference* (*DataItemRef*) modifies the set of resulting *Data Enti-*
 2245 *ties*, limited by a path query parameter of a *Current Request* or *Sample Request*, to
 2246 include the *Data Entity* whose value for its *id* attribute matches the value provided
 2247 for the *idRef* attribute of the *DataItemRef* element. The result is equivalent
 2248 to appending `//[@id=<"idRef">]` to the path query parameters of the *Current*
 2249 *Request* or *Sample Request*. See *Section 8.3.2 - Current Request Implemented Using*
 2250 *HTTP* for more details on path queries.

2251 9 Error Information Model

2252 The *Error Information Model* establishes the rules and terminology that describes the *Re-*
 2253 *sponse Document* returned by an *Agent* when it encounters an error while interpreting a
 2254 *Request* for information from a client software application or when an *Agent* experiences
 2255 an error while publishing the *Response* to a *Request* for information.

2256 An *Agent* provides the information regarding errors encountered when processing a *Re-*
 2257 *quest* for information by publishing an *MTConnectErrors Response Document* to the client
 2258 software application that made the *Request* for information.

2259 9.1 MTConnectError Response Document

2260 The *MTConnectErrors Response Document* is comprised of two sections: Header and
 2261 Errors.

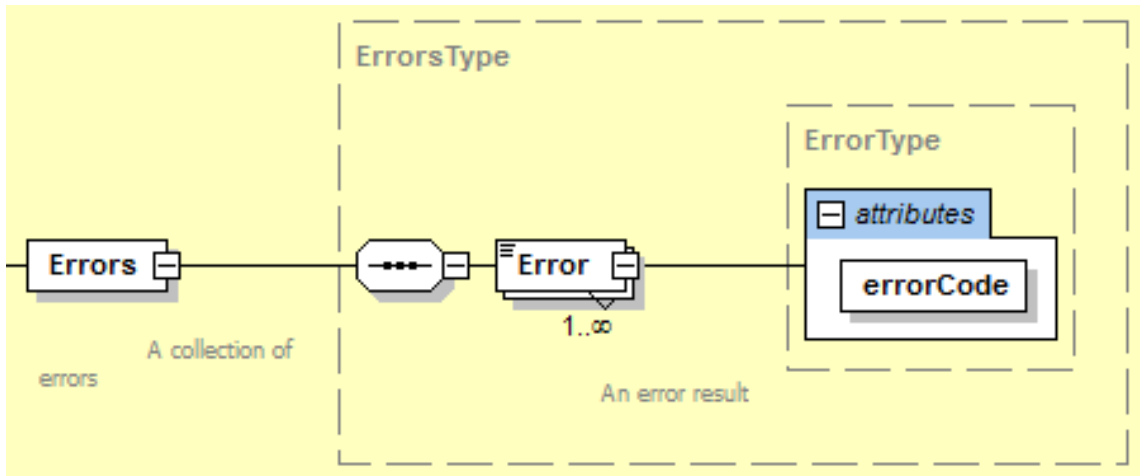
2262 The Header section contains information defining the creation of the document and the
 2263 data storage capability of the *Agent* that generated the document. (See *Section 6.5.4 -*
 2264 *Header for MTConnectError*)

2265 The Errors section of the *MTConnectErrors Response Document* is a *Structural Element*
 2266 that organizes *Data Entities* describing each of the errors reported by an *Agent*.

2267 9.1.1 Structural Element for MTConnectError

2268 *Structural Elements* are XML elements that form the logical structure for an XML docu-
 2269 ment. The *MTConnectErrors Response Document* has only one *Structural Element*. This
 2270 *Structural Element* is Errors. Errors is an XML container element that organizes the
 2271 information and data associated with all errors relevant to a specific *Request* for informa-
 2272 tion.

2273 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 <code>Errors</code> element in an <i>MTConnectErrors Response Document</i>.</p> <p>The <code>Errors</code> element MUST contain at least one <code>Error Data Entity</code> element.</p>	1

2274 Note: When compatibility with Version 1.0.1 and earlier of the MTConnect Standard
 2275 is required for an implementation, the *MTConnectErrors Response Document*
 2276 contains only a single *Error Data Entity* and the *Errors Structural Element*
 2277 **MUST NOT** appear in the document.

2278 9.1.2 Error Data Entity

2279 When an *Agent* encounters an error when responding to a *Request* for information from
 2280 a client software application, the information describing the error(s) is reported as a *Data*
 2281 *Entity* in an *MTConnectErrors Response Document*. *Data Entities* are organized in the
 2282 `Errors` XML container.

2283 There is only one type of *Data Entity* defined for an *MTConnectErrors Response Docu-*
 2284 *ment*. That *Data Entity* is called `Error`.

2285 The following is an illustration of the structure of an XML document demonstrating how
 2286 `Error Data Entities` are reported in an *MTConnectErrors Response Document*:

Example 11: Example of Error in MTConnectError

```
2287 1 <MTConnectError>
2288 2   <Header/>
2289 3   <Errors>
2290 4     <Error/>
2291 5     <Error/>
2292 6     <Error/>
2293 7   </Errors>
2294 8 </MTConnectError>
```

2295 The `Errors` element **MUST** contain at least one *Data Entity*. Each *Data Entity* describes
 2296 the details for a specific error reported by an *Agent* and is represented by the XML element
 2297 named `Error`.

2298 `Error` XML elements **MAY** contain both attributes and CDATA that provide details fur-
 2299 ther defining a specific error. The CDATA **MAY** provide the complete text provided by an
 2300 *Agent* for the specific error.

2301 9.1.2.1 XML Schema Structure for Error

2302 The *XML Schema* in *Figure 22* represents the structure of an `Error` XML element show-
 2303 ing the attributes defined for `Error`.

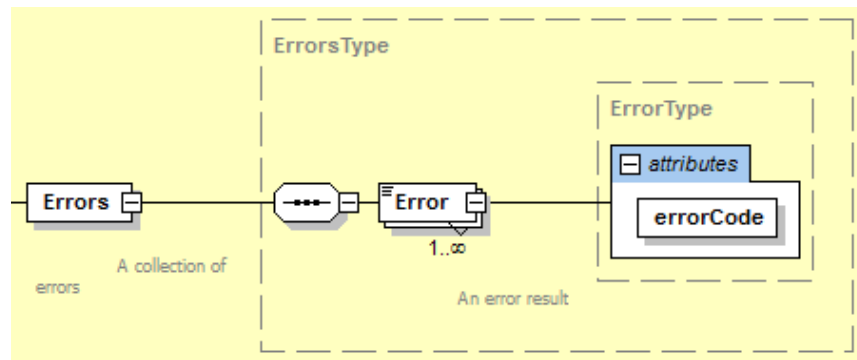


Figure 22: Error Schema Diagram

2304 9.1.2.2 Attributes for Error

2305 Error has one attribute. Table 22 defines this attribute that provides additional informa-
 2306 tion for an Error XML element.

Table 22: Attributes for Error

Attribute	Description	Occurrence
errorCode	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. errorCode is a required attribute.	1

2307 9.1.2.3 Values for errorCode

2308 There is a limited vocabulary defined for errorCode. The value returned for error-
 2309 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.
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> .

2310 9.1.2.4 CDATA for Error

2311 The CDATA for Error contains a textual description of the error and any additional
 2312 information an *Agent* is capable of providing regarding a specific error. The *Valid Data*
 2313 *Value* returned for Error **MAY** be any text string.

2314 9.1.3 Examples for MTConnectError

2315 *Example 12* is an example demonstrating the structure of an *MTConnectErrors Response*
 2316 *Document*:

Example 12: Example of structure for MTConnectError

```

2317 1 <?xml version="1.0" encoding="UTF-8"?>
2318 2   <MTConnectError
2319 3     xmlns="urn:mtconnect.org:MTConnectError:1.4"
2320 4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
2321 5     xsi:schemaLocation="urn:mtconnect.org:MTConnectError
2322 6       :1.4/schemas/MTConnectError_1.4.xsd">
2323 7   <Header creationTime="2010-03-12T12:33:01Z"
2324 8     sender="MyAgent" version="1.4.1.10"
2325 9     bufferSize="131000" instanceId="1383839" />
2326 10  <Errors>
2327 11    <Error errorCode="OUT_OF_RANGE" >Argument was
2328 12      out of range</Error>
2329 13    <Error errorCode="INVALID_XPATH" >Bad
2330 14      path</Error>
2331 15  </Errors>
2332 16 </MTConnectError>

```

2333 *Example 13* is an example demonstrating the structure of an *MTConnectErrors Response*
 2334 *Document* when backward compatibility with Version 1.0.1 and earlier of the MTConnect
 2335 Standard is required. In this case, the *Document Body* contains only a single *Error Data*
 2336 *Entity* and the *Errors Structural Element* **MUST NOT** appear in the document.

Example 13: Example of structure for MTConnectError when backward compatibility is required

```

2337 1 <?xml version="1.0" encoding="UTF-8"?>
2338 2 <MTConnectError
2339 3   xmlns="urn:mtconnect.org:MTConnectError:1.1"
2340 4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
2341 5   xsi:schemaLocation="urn:mtconnect.org:MTConnectError
2342 6     :1.1/schemas/MTConnectError_1.1.xsd">
2343 7   <Header creationTime="2010-03-12T12:33:01Z"
2344 8     sender="MyAgent" version="1.1.0.10"
2345 9     bufferSize="131000" instanceId="1383839" />

```

```
2346 10    <Error errorCode="OUT_OF_RANGE" >Argument was out
2347 11      of range</Error>
2348 12 </MTConnectError>
```

2349 Appendices

2350 A Bibliography

- 2351 Engineering Industries Association. *EIA Standard - EIA-274-D*, Interchangeable Variable,
 2352 Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically
 2353 Controlled Machines. Washington, D.C. 1979.
- 2354 ISO TC 184/SC4/WG3 N1089. *ISO/DIS 10303-238*: Industrial automation systems and
 2355 integration Product data representation and exchange Part 238: Application Protocols: Ap-
 2356 plication interpreted model for computerized numerical controllers. Geneva, Switzerland,
 2357 2004.
- 2358 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
 2359 tems and integration – Physical device control – Data model for computerized numerical
 2360 controllers – Part 10: General process data. Geneva, Switzerland, 2004.
- 2361 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
 2362 tems and integration – Physical device control – Data model for computerized numerical
 2363 controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.
- 2364 International Organization for Standardization. *ISO 6983/1* – Numerical Control of ma-
 2365 chines – Program format and definition of address words – Part 1: Data format for posi-
 2366 tioning, line and contouring control systems. Geneva, Switzerland, 1982.
- 2367 Electronic Industries Association. *ANSI/EIA-494-B-1992*, 32 Bit Binary CL (BCL) and
 2368 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines.
 2369 Washington, D.C. 1992.
- 2370 National Aerospace Standard. *Uniform Cutting Tests - NAS Series: Metal Cutting Equip-*
 2371 *ment Specifications*. Washington, D.C. 1969.
- 2372 International Organization for Standardization. *ISO 10303-11*: 1994, Industrial automa-
 2373 tion systems and integration Product data representation and exchange Part 11: Descrip-
 2374 tion methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.
- 2375 International Organization for Standardization. *ISO 10303-21*: 1996, Industrial automa-
 2376 tion systems and integration – Product data representation and exchange – Part 21: Imple-
 2377 mentation methods: Clear text encoding of the exchange structure. Geneva, Switzerland,
 2378 1996.
- 2379 H.L. Horton, F.D. Jones, and E. Oberg. *Machinery's Handbook*. Industrial Press, Inc.

2380 New York, 1984.

2381 International Organization for Standardization. *ISO 841-2001: Industrial automation sys-*
2382 *tems and integration - Numerical control of machines - Coordinate systems and motion*
2383 *nomenclature*. Geneva, Switzerland, 2001.

2384 *ASME B5.59-2 Version 9c: Data Specification for Properties of Machine Tools for Milling*
2385 *and Turning*. 2005.

2386 *ASME/ANSI B5.54: Methods for Performance Evaluation of Computer Numerically Con-*
2387 *trolled Lathes and Turning Centers*. 2005.

2388 OPC Foundation. *OPC Unified Architecture Specification, Part 1: Concepts Version 1.00.*
2389 *July 28, 2006*.

2390 View the following site for RFC references: <http://www.faqs.org/rfcs/>.

2391 B Fundamentals of Using XML to Encode Response Documents

2392 The MTConnect Standard specifies the structures and constructs that are used to encode
 2393 *Response Documents*. When these *Response Documents* are encoded using XML, there
 2394 are additional rules defined by the XML standard that apply for creating an XML compli-
 2395 ant document. An implementer should refer to the W3C website for additional information
 2396 on XML documentation and implementation details - <http://www.w3.org/XML>.

2397 The following provides specific terms and guidelines referenced in the MTConnect Stan-
 2398 dard for forming *Response Documents* with XML:

- 2399 • tag: A tag is an XML construct that forms the foundation for an XML expression.
 2400 It defines the scope (beginning and end) of an XML expression. The main types of
 2401 tags are:
- 2402 • start-tag: Designates the beginning on an XML element; e.g., `<Element Name>`
- 2403 • end-tag: Designates the end on an XML element; e.g., `</Element Name>`.
 2404 Note: If an element has no *Child Elements* or CDATA, the end-tag may be
 2405 shortened to `/>`.
- 2406 • Element: An element is an XML statement that is the primary building block
 2407 for a document encoded using XML. An element begins with a start-tag and
 2408 ends with a matching end-tag. The characters between the start-tag and the
 2409 end-tag are the element's content. The content may contain attributes, CDATA,
 2410 and/or other elements. If the content contains additional elements, these elements
 2411 are called *Child Elements*.
 2412 An example would be: `<Element Name>Content of the Element</Element Name>`.
- 2413 • *Child Element*: An XML element that is contained within a higher-level *Parent El-*
 2414 *ement*. A *Child Element* is also known as a sub-element. XML allows an unlimited
 2415 hierarchy of *Parent Element-Child Element* relationships that establishes the struc-
 2416 ture that defines how the various pieces of information in the document relate to
 2417 each other. A *Parent Element* may have multiple associated *Child Elements*.
- 2418 • *Element Name*: A descriptive identifier contained in both the start-tag and
 2419 end-tag that provides the name of an XML element.
- 2420 • Attribute: A construct consisting of a name-value pair that provides additional
 2421 information about that XML element. The format for an attribute is `name="value"`;
 2422 where the value for the attribute is enclosed in a set of quotation (") marks. An XML
 2423 attribute **MUST** only have a single value and each attribute can appear at most once
 2424 in each element. Also, each attribute **MUST** be defined in a *schema* to either be
 2425 required or optional.

- 2426 • An example of attributes for an XML element is *Example 14*:

Example 14: Example of attributes for an element

```
2427 1 <DataItem category="SAMPLE" id="S1load"
2428 2   nativeUnits="PERCENT" type="LOAD"
2429 3   units="PERCENT"/>
```

2430 In this example, `DataItem` is the `ElementName`. `category`, `id`, `nativeU-`
 2431 `nits`, `type`, and `units` are the names of the attributes. “SAMPLE”, “S1load”,
 2432 “PERCENT”, “LOAD”, and “PERCENT” are the values for each of the respective
 2433 attributes.

- 2434 • **CDATA:** CDATA is an XML term representing *Character Data*. *Character Data*
 2435 contains a value(s) or text that is associated with an XML element. CDATA can be
 2436 restricted to certain formats, patterns, or words.

2437 An example of CDATA associated with an XML element would be *Example 15*:

Example 15: Example of cdata associated with element

```
2438 1 <Message id="M1">This is some text</Message>
```

2439 In this example, `Message` is the `ElementName` and `This is some text` is
 2440 the CDATA.

- 2441 • **namespace:** An XML *namespace* defines a unique vocabulary for named elements
 2442 and attributes in an XML document. An XML document may contain content that is
 2443 associated with multiple *namespaces*. Each *namespace* has its own unique identifier.

2444 Elements and attributes are associated with a specific *namespace* by placing a pre-
 2445 fix on the name of the element or attribute that associates that name to a specific
 2446 *namespace*; e.g., `x:MyTarget` associates the element name `MyTarget` with the
 2447 *namespace* designated by `x:` (the prefix).

2448 *namespaces* are used to avoid naming conflicts within an XML document. The
 2449 naming convention used for elements and attributes may be associated with either
 2450 the default *namespace* specified in the *Header* of an XML document or they may
 2451 be associated with one or more alternate *namespaces*. All elements or attributes
 2452 associated with a *namespace* that is not the default *namespace*, must include a prefix
 2453 (e.g., `x:`) as part of the name of the element or attribute to associate it with the proper
 2454 *namespace*. See *Appendix C* for details on the structure for XML *Headers*.

2455 The names of the elements and attributes declared in a *namespace* may be identified
 2456 with a different prefix than the prefix that signifies that specific *namespace*. These
 2457 prefixes are called *namespace aliases*. As an example, MTConnect Standard spe-
 2458 cific *namespaces* are designated as `m:` and the names of the elements and attributes
 2459 defined in that *namespace* have an alias prefix of `mt:` which designates these names
 2460 as MTConnect Standard specific vocabulary; e.g., `mt:MTConnectDevices`.

2461 XML documents are encoded with a hierarchy of elements. In general, XML elements
 2462 may contain *Child Elements*, CDATA, or both. However, in the MTConnect Standard,
 2463 an element **MUST NOT** contain mixed content; meaning it cannot contain both *Child*
 2464 *Elements* and CDATA.

2465 The *semantic data model* defined for each *Response Document* specifies the elements and
 2466 *Child Elements* that may appear in a document. The *semantic data model* also defines the
 2467 number of times each element and *Child Element* may appear in the document.

2468 *Example 16* demonstrates the hierarchy of XML elements and *Child Elements* used to
 2469 form an XML document:

Example 16: Example of hierarchy of XML elements

```

2470 1 <Root Level>      (Parent Element)
2471 2   <First Level>   (Child Element to Root Level and
2472 3   Parent Element to Second Level)
2473 4     <Second Level> (Child Element to First Level
2474 5     and Parent Element to Third Level)
2475 6       <Third Level name="N1"></Third Level>
2476 7       (Child Element to Second Level)
2477 8       <Third Level name="N2"></Third Level>
2478 9       (Child Element to Second Level)
2479 10      <Third Level name="N3"></Third Level>
2480 11      (Child Element to Second Level)
2481 12      </Second Level>   (end-tag for Second Level)
2482 13      </First Level>    (end-tag for First Level)
2483 14      </Root Level>     (end-tag for Root Level)
  
```

2484 In the *Example 16*, *Root Level* and *First Level* have one *Child Element* (sub-elements)
 2485 each and *Second Level* has three *Child Elements*; each called *Third Level*. Each *Third*
 2486 *Level* element has a different name attribute. Each level in the structure is an element and
 2487 each lower level element is a *Child Element*.

2488 C Schema and Namespace Declaration Information

2489 There are four pseudo-attributes typically included in the *Header* of a *Response Document*
 2490 that declare the *schema* and *namespace* for the document. Each of these pseudo-attributes
 2491 provides specific information for a client software application to properly interpret the
 2492 content of the *Response Document*.

2493 The pseudo-attributes include:

2494 • `xmlns:xsi` – The `xsi` portion of this attribute name stands for *XML Schema*
 2495 instance. An *XML Schema* instance provides information that may be used by a
 2496 software application to interpret XML specific information within a document. See
 2497 the W3C website for more details on `xmlns:xsi`.

2498 • `xmlns` – Declares the default *namespace* associated with the content of the *Re-*
 2499 *sponse Document*. The default *namespace* is considered to apply to all elements and
 2500 attributes whenever the name of the element or attribute does not contain a prefix
 2501 identifying an alternate *namespace*.

2502 The value of this attribute is an URN identifying the name of the file that defines
 2503 the details of the *namespace* content. This URN provides a unique identify for the
 2504 *namespace*.

2505 • `xmlns:m` – Declares the MTConnect specific *namespace* associated with the con-
 2506 tent of the *Response Document*. There may be multiple *namespaces* declared for
 2507 an XML document. Each may be associated to the default *namespace* or it may be
 2508 totally independent. The `:m` designates that this is a specific MTConnect *namespace*
 2509 which is directly associated with the default *namespace*.

2510 Note: See *Section 6.7 - Extensibility* for details regarding extended *namespaces*.

2511 The value associated with this attribute is an URN identifying the name of the file
 2512 that defines the details of the *namespace* content.

2513 • `xsi:schemaLocation` - Declares the name for the *schema* associated with the
 2514 *Response Document* and the location of the file that contains the details of the
 2515 *schema* for that document.

2516 The value associated with this attribute has two parts:

2517 - A URN identifying the name of the specific *XML Schema* instance associated
 2518 with the *Response Document*.

2519 - The path to the location where the file describing the specific *XML Schema*
 2520 instance is located. If the file is located in the same root directory where the *Agent*
 2521 is installed, then the local path MAY be declared. Otherwise, a fully qualified URL
 2522 must be declared to identify the location of the file.

2523 Note: In the format of the value associated with `xsi:schemaLocation`, the
 2524 URN and the path to the *schema* file **MUST** be separated by a “space”.

2525 In *Example 17*, the first line is the *XML Declaration*. The second line is a *Root Ele-*
 2526 *ment* called `MTConnectDevices`. The remaining four lines are the pseudo-attributes of
 2527 `MTConnectDevices` that declare the XML *schema* and *namespace* associated with an
 2528 `MTConnectDevices Response Document`.

Example 17: Example of schema and namespace declaration

```
2529 1  <?xml version="1.0" encoding="UTF-8"?>
2530 2  <MTConnectDevices
2531 3      xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
2532 4      xmlns="urn:mtconnect.org:MTConnectDevices:1.3"
2533 5      xmlns:m="urn:mtconnect.org:MTConnectDevices:1.3"
2534 6      xsi:schemaLocation="urn:mtconnect.org:
2535 7          MTConnectDevices:1.3 /schemas/MTConnectDevices\_1.3.xsd">
```

2536 The format for the values provided for each of the pseudo-attributes **MUST** reference
 2537 the *semantic data model* (e.g., `MTConnectDevices`, `MTConnectStreams`, `MTCon-`
 2538 `nectAssets`, or `MTConnectError`) and the version (i.e.; 1.1, 1.2, 1.3, etc.) of
 2539 the MTConnect Standard that depict the *schema* and *namespace(s)* associated with a spe-
 2540 cific *Response Document*.

2541 When an implementer chooses to extend an MTConnect *Data Model* by adding custom
 2542 data types or additional *Structural Elements*, the *schema* and *namespace* for that *Data*
 2543 *Model* should be updated to reflect the additional content. When this is done, the *names-*
 2544 *pace* and *schema* information in the *Header* should be updated to reflect the URI for the
 2545 extended *namespace* and *schema*.



MTConnect® Standard

Part 2.0 – Devices Information Model

Version 1.8.0

Prepared for: MTConnect Institute
Prepared on: September 6, 2021

MTConnect® is a registered trademark of AMT - The Association for Manufacturing Technology. Use of *MTConnect* is limited to use as specified on <http://www.mtconnect.org/>.

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	11
2.3	MTConnect References	11
3	Devices Information Model	12
4	Structural Elements for MTConnectDevices	14
4.1	Devices	17
4.2	Device	18
4.2.1	Agent	18
4.3	Components	19
4.4	Component	19
4.4.1	XML Schema Structure for Component	20
4.4.2	Attribute for Component	20
4.4.3	Elements of Component	24
4.4.3.1	Description for Component	24
4.4.3.2	Configuration for Component	26
4.4.3.3	DataItems for Component	27
4.4.3.4	Components within Component	28
4.4.3.5	Compositions for Component	28
4.4.3.6	References for Component	28
4.5	Compositions	28
4.6	Composition	29
4.6.1	XML Schema Structure for Composition	30
4.6.2	Attributes for Composition	31
4.6.3	Elements of Composition	32
4.6.3.1	Description for Composition	33
4.7	References	34
4.8	Reference	35
4.8.1	ComponentRef	36
4.8.2	DataItemRef	37
5	Component Structural Elements	39
5.1	Axes	40
5.1.1	Axis	41
5.1.2	Cartesian Coordinate Naming Conventions	41
5.1.2.1	Linear Motion	41
5.1.2.2	Rotary Motion	42

5.1.3	Articulated Machine Control Systems	42
5.1.4	Articulated Machine Axis Names	42
5.1.5	Rotary Component	42
5.1.6	Linear Component	43
5.2	Controller	43
5.2.1	Path	43
5.3	Systems	43
5.3.1	System	44
5.3.2	Hydraulic	44
5.3.3	Pneumatic	44
5.3.4	Coolant	44
5.3.5	Lubrication	44
5.3.6	Electric	44
5.3.7	Enclosure	45
5.3.8	Protective	45
5.3.9	ProcessPower	45
5.3.10	Feeder	45
5.3.11	Dielectric	46
5.3.12	EndEffector	46
5.3.13	WorkEnvelope	46
5.3.14	Heating	46
5.3.15	Cooling	46
5.3.16	Pressure	47
5.3.17	Vacuum	47
5.4	Auxiliaries	47
5.4.1	Auxiliary	47
5.4.2	Loader	47
5.4.2.1	BarFeeder	47
5.4.3	WasteDisposal	48
5.4.4	ToolingDelivery	48
5.4.4.1	AutomaticToolChanger	48
5.4.4.2	ToolMagazine	48
5.4.4.3	Turret	48
5.4.4.4	GangToolBar	48
5.4.4.5	ToolRack	49
5.4.5	Environmental	49
5.4.6	Sensor	49
5.4.7	Deposition	49
5.5	Resources	49
5.5.1	Resource	50
5.5.2	Materials	50
5.5.2.1	Stock	50

5.5.3	Personnel	50
5.6	Interfaces	50
5.6.1	Interface	50
5.7	Adapters	51
5.7.1	Adapter	51
5.8	Structures	51
5.8.1	Structure	51
5.8.2	Link	51
5.9	Other Components	51
5.9.1	Actuator	52
5.9.2	Door	52
5.9.3	Sensor	52
5.9.4	Processes	52
5.9.4.1	ProcessOccurrence	52
5.9.5	Parts	53
5.9.5.1	PartOccurrence	53
5.9.6	Lock	53
6	Composition Type Structural Elements	54
7	Data Entities for Device	59
7.1	DataItems	60
7.2	DataItem	60
7.2.1	XML Schema Structure for DataItem	60
7.2.2	Attributes for DataItem	61
7.2.2.1	name Attribute for DataItem	66
7.2.2.2	id Attribute for DataItem	66
7.2.2.3	type and subType Attributes for DataItem	68
7.2.2.4	statistic Attribute for DataItem	68
7.2.2.5	units Attribute for DataItem	70
7.2.2.6	nativeUnits Attribute for DataItem	72
7.2.2.7	nativeScale Attribute for DataItem	74
7.2.2.8	category Attribute for DataItem	74
7.2.2.9	coordinateSystem Attribute for DataItem	75
7.2.2.10	compositionId Attribute for DataItem	76
7.2.2.11	sampleRate Attribute for DataItem	76
7.2.2.12	representation Attribute for DataItem	76
7.2.2.13	significantDigits Attribute for DataItem	78
7.2.2.14	discrete Attribute for DataItem	78
7.2.3	Elements for DataItem	79
7.2.3.1	Source Element for DataItem	80
7.2.3.1.1	Attributes for Source	81

7.2.3.2	Constraints Element for DataItem	82
7.2.3.2.1	Schema for Constraints	82
7.2.3.3	Filters Element for DataItem	85
7.2.3.3.1	Filter	86
7.2.3.4	InitialValue Element for DataItem	87
7.2.3.5	ResetTrigger Element for DataItem	87
7.2.3.6	Definition Element for DataItem	89
7.2.3.6.1	EntryDefinitions Element for Definition	91
7.2.3.6.2	EntryDefinition Element for Definition	91
7.2.3.6.3	CellDefinitions Element for Definition	92
7.2.3.6.4	CellDefinition Element for CellDefinitions	93
7.2.3.7	Relationships Element for DataItem	94
7.2.3.7.1	DataItemRelationship	94
7.2.3.7.2	SpecificationRelationship	95
8	Listing of Data Items	96
8.1	Data Items in category SAMPLE	97
8.2	Data Items in category EVENT	119
8.3	Data Items in category CONDITION	159
9	Configuration	161
9.1	Sensor	162
9.1.1	Sensor Data	163
9.1.2	Sensor Unit	164
9.1.3	Sensor Configuration	166
9.1.3.1	Elements for SensorConfiguration	168
9.1.3.1.1	Attributes for Channel	169
9.1.3.1.2	Elements for Channel	170
9.2	Relationships	172
9.2.1	Relationship	172
9.2.1.1	DeviceRelationship	175
9.2.1.2	ComponentRelationship	180
9.3	Specifications	182
9.3.1	Specification	184
9.3.1.1	Attributes for Specification	184
9.3.1.2	Elements for Specification	186
9.3.2	ProcessSpecification	186
9.3.2.1	Elements for ProcessSpecification	188
9.3.2.2	ControlLimits	188
9.3.2.2.1	Elements for ControlLimits	188
9.3.2.3	SpecificationLimits	189
9.3.2.3.1	Elements for SpecificationLimits	189

9.3.2.4	AlarmLimits	189
9.3.2.4.1	Elements for AlarmLimits	189
9.4	CoordinateSystems	190
9.4.1	CoordinateSystem	191
9.4.1.1	Attributes for CoordinateSystem	191
9.4.1.1.1	CoordinateSystem types	191
9.4.1.2	Elements for CoordinateSystem	193
9.4.1.2.1	Elements for Transformation	193
9.5	Motion	194
9.5.1	Attributes for Motion	195
9.5.1.1	Motion types	196
9.5.1.2	Motion actuation types	196
9.5.2	Elements for Motion	197
9.6	SolidModel	198
9.6.1	Attributes for SolidModel	199
9.6.1.1	SolidModel mediaType	200
9.6.2	Elements for SolidModel	201
Appendices		202
A	Bibliography	202

Table of Figures

Figure 1: Example Device Structural Elements	16
Figure 2: Example Composition Structural Elements	17
Figure 3: Component Diagram	20
Figure 4: Description of Component Diagram	25
Figure 5: Component Configuration Diagram	27
Figure 6: Composition Diagram	31
Figure 7: Description of Composition Diagram	33
Figure 8: Reference Diagram	36
Figure 9: ComponentRef Diagram	36
Figure 10:DataItemRef Diagram	37
Figure 11:Example Data Entities for Device (DataItem)	59
Figure 12:DataItem Diagram	61
Figure 13:Source Diagram	81
Figure 14:Constraints Diagram	83
Figure 15:Filter Diagram	86
Figure 16:Definition Diagram	90
Figure 17:Configuration Element	161
Figure 18:Sensor Data Associations	163
Figure 19:SensorConfiguration Diagram	167
Figure 20:Relationship Diagram	174
Figure 21:DeviceRelationship Diagram	176
Figure 22:ComponentRelationship Diagram	180
Figure 23:Specifications Diagram	183
Figure 24:ProcessSpecification Diagram	187
Figure 25:CoordinateSystems Diagram	190
Figure 26:Motion Diagram	195
Figure 27:SolidModel Diagram	199

List of Tables

Table 1: MTConnect Devices Element	17
Table 2: Attributes for Device	18
Table 3: MTConnect Components Element	19
Table 4: MTConnect Component Element	19
Table 5: Attributes for Component	21
Table 6: Elements for Component	24
Table 7: Attributes for Description for Component	25
Table 8: MTConnect Configuration Element for Component	27
Table 9: MTConnect Compositions Element	29
Table 10: MTConnect Composition Element	30
Table 11: Attributes for Composition	31
Table 12: Elements for Composition	33
Table 13: Attributes for Description for Composition	34
Table 14: MTConnect References Element	35
Table 15: Attributes for ComponentRef	37
Table 16: Attributes for DataItemRef	38
Table 17: Top Level Component Elements	39
Table 18: Composition type Elements	54
Table 19: MTConnect DataItems Element	60
Table 20: MTConnect DataItem Element	60
Table 21: Attributes for DataItem	62
Table 22: DataItem attribute statistic type	69
Table 23: DataItem attribute units type	70
Table 24: DataItem attribute nativeunits type	72
Table 25: DataItem attribute coordinateSystem type	75
Table 26: DataItem attribute representation type	77
Table 27: Elements for DataItem	79
Table 28: Attributes for Source	81
Table 29: Elements for Constraints	84
Table 30: MTConnect Filters Element	85
Table 31: DataItem Element Filter type	86
Table 32: MTConnect ResetTrigger Element	88
Table 33: DataItem Element ResetTrigger type	88
Table 34: Elements for Definition	90
Table 35: Elements for EntryDefinitions	91
Table 36: Attributes for EntryDefinition	92
Table 37: Elements for EntryDefinition	92
Table 38: Elements for CellDefinitions	93
Table 39: Attributes for CellDefinition	93
Table 40: Elements for CellDefinition	94

Table 41: Attributes for DataItemRelationship	94
Table 42: Attributes for SpecificationRelationship	95
Table 43: DataItem type subType for category SAMPLE	97
Table 44: DataItem type subType for category EVENT	119
Table 45: DataItem type for category CONDITION	159
Table 46: Types of Configuration	162
Table 47: MTConnect SensorConfiguration Element	168
Table 48: Elements for SensorConfiguration	168
Table 49: Attributes for Channel	170
Table 50: Elements for Channel	170
Table 51: MTConnect Relationships Element	172
Table 52: Attributes for DeviceRelationship	177
Table 53: Attributes for ComponentRelationship	181
Table 54: Attributes for Specification	184
Table 55: Elements for Specification	186
Table 56: Elements for ProcessSpecification	188
Table 57: Elements for ControlLimits	188
Table 58: Elements for SpecificationLimits	189
Table 59: Elements for AlarmLimits	190
Table 60: Attributes for CoordinateSystem	191
Table 61: CoordinateSystem types	192
Table 62: Elements for CoordinateSystem	193
Table 63: Elements for Transformation	193
Table 64: Attributes for Motion	195
Table 65: Motion types	196
Table 66: Motion actuation types	197
Table 67: Elements for Motion	197
Table 68: Attributes for SolidModel	199
Table 69: SolidModel mediaType	200
Table 70: Elements for SolidModel	201

1 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 NMTOKEN

28 The data type for XML identifiers.

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

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

33 URI

34 Stands for Universal Resource Identifier.

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

36 URL

37 Stands for Uniform Resource Locator.

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

39 UUID

40 General meaning:

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

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

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

46 Used as an attribute for an XML element:

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

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

50 W3C

51 The World Wide Web Consortium (W3C) is an international community that devel-
52 ops open standards to ensure the long-term growth of the Web.

53 See <https://www.w3.org/>.

54 XML

55 Stands for eXtensible Markup Language.

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

58 XML is the language used for all code examples in the MTConnect Standard.

59 Refer to <http://www.w3.org/XML> for more information about XML.

60 *Adapter*

61 An optional piece of hardware or software that transforms information provided by
62 a piece of equipment into a form that can be received by an *Agent*.

63 Appears in the documents in the following form: `adapter`.

64 *Agent*

65 Refers to an MTConnect Agent.

66 Software that collects data published from one or more piece(s) of equipment, orga-
67 nizes that data in a structured manner, and responds to requests for data from client
68 software systems by providing a structured response in the form of a *Response Doc-*
69 *ument* that is constructed using the *semantic data models* defined in the Standard.

70 Appears in the documents in the following form: *Agent*.

71 *Asset*

72 item, thing or entity that has potential or actual value to an organization *Ref:ISO*
73 *55000:2014(en)*

Note 1 to entry: Value can be tangible or intangible, financial or non-financial, and includes consideration of risks and liabilities. It can be positive or negative at different stages of the asset life.

Note 2 to entry: Physical assets usually refer to equipment, inventory and properties owned by the organization. Physical assets are the opposite of intangible assets, which are non-physical assets such as leases, brands, digital assets, use rights, licences, intellectual property rights, reputation or agreements.

Note 3 to entry: A grouping of assets referred to as an asset system could also be considered as an asset.

Attachment

The connection by which one thing is associated with another.

Child Element

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

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

Component

General meaning:

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

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

Used in *Information Models*:

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

- When used as an XML container to organize *Lower Level* *Component* elements.

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

- When used as an abstract XML element. *Component* is replaced in a data model by a type of *Component* element. *Component* is also an XML container used to organize *Lower Level* *Component* elements, *Data Entities*, or both.

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

106 ***Controlled Vocabulary***

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

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

110 ***Current Request***

111 A *Current Request* is a *Request* to an *Agent* to produce an *MTConnectStreams Re-*
112 *sponse Document* containing the *Observations Information Model* for a snapshot of
113 the latest *observations* at the moment of the *Request* or at a given *sequence number*.

114 ***Data Entity***

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

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

119 ***Data Set***

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

121 ***Devices Information Model***

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

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

125 ***engineering units***

126 A quantity, dimension, or magnitude used in engineering adopted as a standard in
127 terms of which the magnitude of other quantities of the same kind can be expressed
128 or calculated.

129 ***Equipment Metadata***

130 See *Metadata*

131 ***Force***

132 A push or pull on a mass which results in an acceleration.

133 ***Information Model***

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

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

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

Interface

The means by which communication is achieved between independent systems.

key

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

key-value pair

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

Lower Level

A nested element that is below a higher level element.

lower limit

The lower conformance boundary for a variable.

Note: immediate concern or action may be required.

lower warning

The lower boundary indicating increased concern and supervision may be required.

Metadata

Data that provides information about other data.

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

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

MTConnect Agent

See definition for *Agent*.

MTConnectDevices Response Document

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

167 ***MTConnectStreams Response Document***

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

170 ***nominal***

171 The ideal or desired value for a variable.

172 ***observation***

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

174 ***Observations Information Model***

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

177 ***organize***

178 The act of containing and owning one or more elements.

179 ***Parent Element***

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

182 Appears in the documents in the following form: *Parent Element*.

183 ***Part***

184 *Part* is defined as a discrete item that has both defined and measurable physical
185 characteristics including mass, material and features and is created by applying one
186 or more manufacturing process steps to a workpiece.

187 ***Probe Request***

188 A *Probe Request* is a *Request* to an *Agent* to produce an *MTConnectDevices Re-*
189 *sponse Document* containing the *Devices Information Model*.

190 ***Request***

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

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

194 ***Response Document***

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

197 ***Sample Request***

198 A *Sample Request* is a *Request* to an *Agent* to produce an *MTConnectStreams Re-*
 199 *sponse Document* containing the *Observations Information Model* for a set of time-
 200 stamped *observations* made by *Components*.

201 ***semantic data model***

202 A methodology for defining the structure and meaning for data in a specific logical
 203 way.

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

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

207 ***sensing element***

208 A mechanism that provides a signal or measured value.

209 ***Sensor***

210 A *sensing element* that responds to a physical stimulus and transmits a resulting
 211 signal.

212 ***sensor element***

213 A *sensor element* provides a signal or measured value.

214 ***sensor unit***

215 An intelligent piece of equipment that manages the signals of one or more *sensing*
 216 *elements* and provides the measured values.

217 ***sequence number***

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

220 *sequence number* is a monotonically increasing number within an instance of an
 221 *Agent*.

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

223 ***Spindle***

224 A mechanism that provides rotational capabilities to a piece of equipment.

225 Typically used for either work holding, materials or cutting tools.

226 ***Streaming Data***

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

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

230 ***Streams Information Model***

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

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

235 ***Structural Element***

236 General meaning:

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

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

240 Used to indicate hierarchy of Components:

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

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

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

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

247 ***Table***

248 A two dimensional set of values given by a set of *key-value pairs Table Entries*.
 249 Each *Table Entry* contains a set of *key-value pairs* of *Table Cells*. The `Entry` and
 250 `Cell` elements comprise a tabular representation of the information.

251 ***Table Cell***

252 A subdivision of a *Table Entry* representing a singular value.

253 ***Table Entry***

254 A subdivision of a *Table* containing a set of *key-value pairs* representing *Table Cells*.

255 ***Top Level***

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

258 ***upper limit***

259 The upper conformance boundary for a variable.

260 Note: immediate concern or action may be required.

261 *upper warning*

283 3 Devices Information Model

284 The *Devices Information Model* provides a representation of the physical and logical con-
285 figuration for a piece of equipment used for a manufacturing process or for any other
286 purpose. It also provides the definition of data that may be reported by that equipment.

287 Using information defined in the *Devices Information Model*, a software application can
288 determine the configuration and reporting capabilities of a piece of equipment. To do this,
289 the software application issues a *Probe Request* (defined in *MTConnect Standard Part 1.0*
290 - *Overview and Fundamentals Section 8.1.1*) to an *Agent* associated with a piece of equip-
291 ment. An *Agent* responds to the *Probe Request* with an `MTConnectDevices` XML
292 document that contains information describing both the physical and logical structure of
293 the piece of equipment and a detailed description of each *Data Entity* that can be reported
294 by the *Agent* associated with the piece of equipment. This information allows the client
295 software application to interpret the document and to extract the data with the same mean-
296 ing, value, and context that it had at its original source.

297 The `MTConnectDevices` XML document is comprised of two sections: `Header` and
298 `Devices`.

299 The `Header` section contains protocol related information as defined in *MTConnect Stan-*
300 *dard Part 1.0 - Overview and Fundamentals Section 6.5.1*.

301 The `Devices` section of the `MTConnectDevices` document contains a `Device` XML
302 container for each piece of equipment described in the document. Each `Device` container
303 is comprised of two primary types of XML elements - *Structural Elements* and *Data Enti-*
304 *ties*.

305 *Structural Elements* are defined as XML elements that organize information that repre-
306 sents the physical and logical parts and sub-parts of a piece of equipment (See *Section 4 -*
307 *Structural Elements for MTConnectDevices* for more details).

308 *Data Entities* are defined as XML elements that describe data that can be reported by
309 a piece of equipment. In the *Devices Information Model*, *Data Entities* are defined as
310 `DataItem` elements (See *Section 7 - Data Entities for Device* and *Section 8 - Listing of*
311 *Data Items*).

312 The *Structural Elements* and *Data Entities* in the `MTConnectDevices` document pro-
313 vide information representing the physical and logical structure for a piece of equipment
314 and the types of data that the piece of equipment can report relative to that structure. The
315 `MTConnectDevices` document does not contain values for the data types reported by
316 the piece of equipment. The `MTConnectStreams` document defined in *MTConnect*

317 *Standard: Part 3.0 - Streams Information Model* provides the data values that are reported
 318 by the piece of equipment. As such, most *Structural Elements* and *Data Entities* in the
 319 *MTConnectDevices* document do not contain CDATA. XML elements that provide
 320 values or information in the CDATA will be specifically identified in *Section 4 - Structural*
 321 *Elements for MTConnectDevices*, *Section 7 - Data Entities for Device*, and *Section 9.1 -*
 322 *Sensor*.

323 Note: The *MTConnect Standard* also defines the information model for *Assets*. An
 324 *Asset* is something that is used in the manufacturing process, but is not perma-
 325 nently associated with a single piece of equipment, can be removed from the
 326 piece of equipment without compromising its function, and can be associated
 327 with other pieces of equipment during its lifecycle. See *MTConnect Standard:*
 328 *Part 4.0 - Assets Information Model* for more details on *Assets*.

329 4 Structural Elements for MTConnectDevices

330 *Structural Elements* are XML elements that form the logical structure for the MTCon-
 331 nectDevices XML document. These elements are used to organize information that
 332 represents the physical and logical architecture of a piece of equipment. Refer to *Figure 1*
 333 for an overview of the *Structural Elements* used in an MTConnectDevices document.

334 A variety of *Structural Elements* are defined to describe a piece of equipment. Some
 335 of these elements **MUST** always appear in the MTConnectDevices XML document,
 336 while others are optional and **MAY** be used, as required, to provide additional structure.

337 The first, or highest level, *Structural Element* in a MTConnectDevices XML document
 338 is `Devices`. `Devices` is a container type XML element used to group one or more
 339 pieces of equipment into a single XML document. `Devices` **MUST** always appear in the
 340 MTConnectDevices document.

341 `Device` is the next *Structural Element* in the MTConnectDevices XML document.
 342 `Device` is also a container type XML element. A separate `Device` container is used
 343 to identify each piece of equipment represented in the MTConnectDevices document.
 344 Each `Device` container provides information on the physical and logical structure of
 345 the piece of equipment and the data associated with that equipment. `Device` can also
 346 represent any logical grouping of pieces of equipment that function as a unit or any other
 347 data source that provides data through an *Agent*.

348 One or more `Device` element(s) **MUST** always appear in an MTConnectDevices
 349 document.

350 `Components` is the next *Structural Element* in the MTConnectDevices XML doc-
 351 ument. `Components` is also a container type XML element. `Components` is used to
 352 group information describing *Lower Level* physical parts or logical functions of a piece of
 353 equipment.

354 If the `Components` container appears in the XML document, it **MUST** contain one or
 355 more `Component` type XML elements.

356 `Component` is the next level of *Structural Element* in the MTConnectDevices XML
 357 document. `Component` is both an abstract type XML element and a container type ele-
 358 ment.

359 As an abstract type element, `Component` will never appear in the XML document de-
 360 scribing a piece of equipment and will be replaced by a specific `Component` type defined
 361 in *Section 5 - Component Structural Elements*. Each `Component` type is also a container
 362 type element. As a container, the `Component` type element is used to organize infor-

363 mation describing *Lower Level Structural Elements* or *Data Entities* associated with the
 364 Component.

365 If *Lower Level Structural Elements* are described, these elements are by definition child
 366 Component elements of a parent Component. At this next level, the *Lower Level* child
 367 Component elements are grouped into an XML container called Components.

368 This *Lower Level* Components container is comprised of one or more child Compo-
 369 nent XML elements representing the sub-parts of the parent Component. Just like the
 370 parent Component element, the child Component element is an abstract type XML el-
 371 ement and will never appear in the XML document – only the different *Lower Level* child
 372 Component types will appear.

373 This parent-child relationship can continue to any depth required to fully define a piece of
 374 equipment.

375 *Example 1* illustrates the relationship between a parent Component and *Lower Level*
 376 child components:

Example 1: Component Levels

```

377 1 <Devices>
378 2   <Device>
379 3     <Components>
380 4       <Axes>   Parent Component
381 5         <Components>
382 6           <Rotary>  Child component of Axes and Parent component of Lower Level compo-
383 nents
384 7             <Components>
385 8               <Chuck>  Child Component of Rotary
```

386 *Figure 1* demonstrates the various *Structural Elements* provided to describe a piece of
 387 equipment and the relationship between these elements.

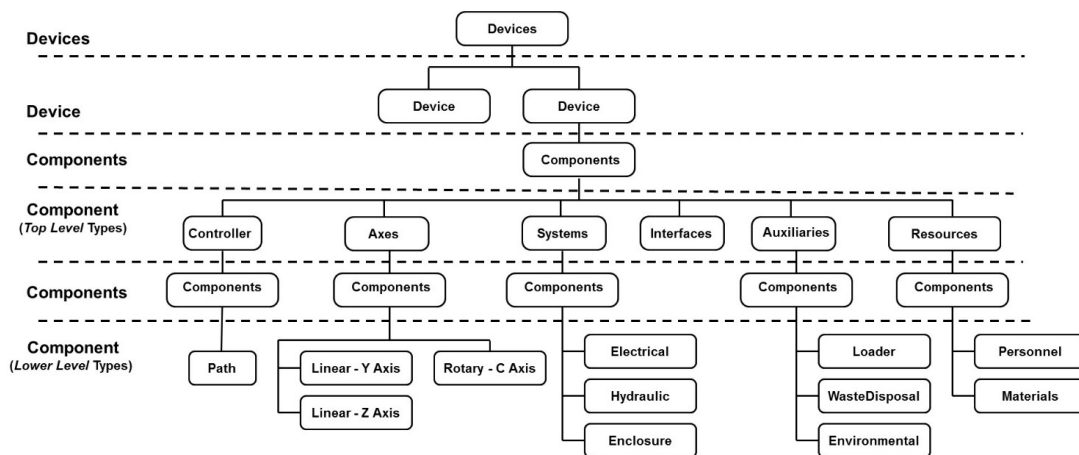


Figure 1: Example Device Structural Elements

Component type XML elements **MAY** be further decomposed into Composition type XML elements. Composition elements describe the lowest level basic structural or functional building blocks contained within a Component. Any number of Composition elements **MAY** be used. Data provided for a Component provides more specific meaning when it is associated with one of the Composition elements of the Component. The different Composition types that **MAY** appear in the XML document are defined in Section 6 - Composition Type Structural Elements.

The Composition elements are organized into a Compositions container. The Compositions container **MAY** appear in the XML document further describing a Component. If one or more Composition element(s) is provided to describe a Component, a Compositions container **MUST** be defined for the Component.

Example 2 represents an XML document structure that demonstrates the relationship between a parent Component and its Composition elements.

Example 2: Component levels with Composition

```

401 1 <Devices>
402 2   <Device>
403 3     <Components>
404 4       <Axes>   (Component)
405 5       <Components>
406 6         <Linear> (Component)
407 7         <Compositions>
408 8           <Composition>
409 9           <Composition>
410 10          <Composition>
  
```

Figure 2 demonstrates this relationship between a Component and some of its potential Composition elements.

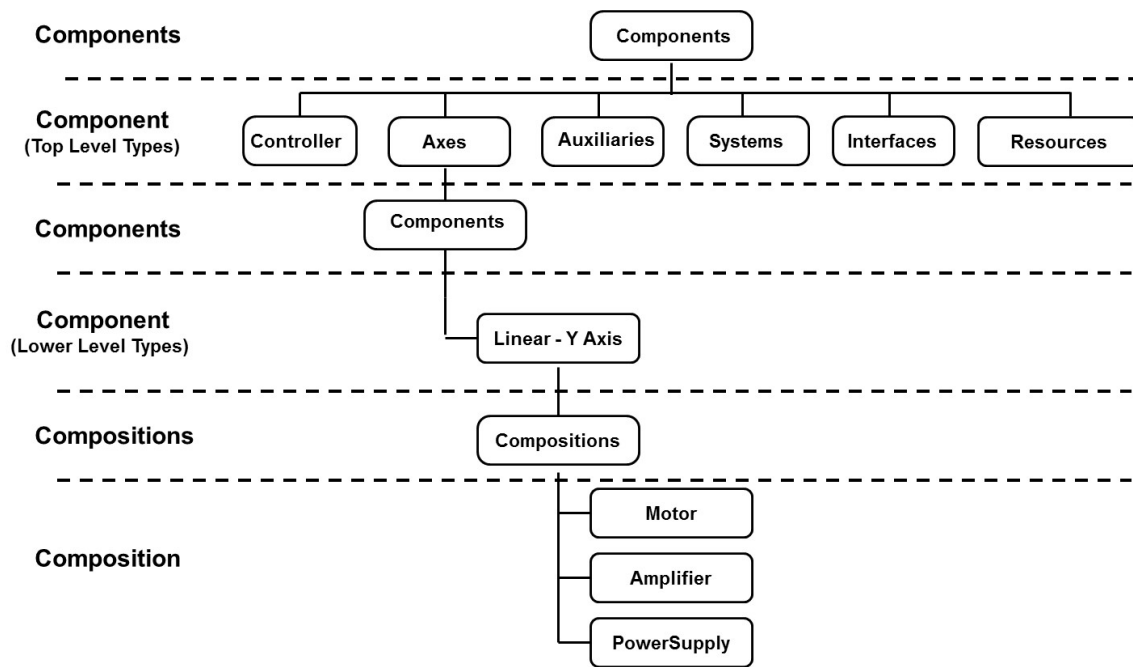


Figure 2: Example Composition Structural Elements

413 4.1 Devices

414 Devices **MUST** *organize* one or more Device elements.

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

415 4.2 Device

416 A Device is a Component that represents a piece of equipment that produces *observa-*
 417 *tions* about itself. It *organizes* its parts as Components.

418 A Device **MUST** have a name and uuid attribute to identify itself.

419 A Device **MUST** have the following DataItems: AVAILABILITY, ASSET_CHANGED,
 420 and ASSET_REMOVED.

421 See *Section 4.4 - Component* for details on the Device model.

422 Table 2 defines additional attributes for a Device Component.

Table 2: Attributes for Device

Attribute	Description	Occurrence
mtconnectVersion	The MTConnect version of the <i>Devices Information Model</i> used to configure the information to be published for a piece of equipment in an <i>MTConnect Response Document</i> .	0..1

423 4.2.1 Agent

424 Agent is a Device representing the *MTConnect Agent* and all its connected data sources.

- 425 • It **MUST** be provided by all *MTConnect Agent* implementations.
- 426 • It **MUST** provide notifications when devices are added or changed.
- 427 • It **MUST** provide connection information for each data source currently supplying
 428 data to the *MTConnect Agent*.
- 429 • It **MAY** provide information about telemetry relating to data sources.
- 430 • It **MAY** provide information about the *MTConnect Agent* resource utilization.

431 4.3 Components

432 `Components` is an XML container used to group information describing physical parts
 433 or logical functions of a piece of equipment. `Components` contains one or more `Com-`
 434 `ponent` XML elements.

Table 3: MTConnect Components Element

Element	Description	Occurrence
Components	An XML container that consists of one or more types of Component 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

435 4.4 Component

436 A `Component` XML element is a container type XML element used to organize informa-
 437 tion describing a physical part or logical function of a piece of equipment. It also provides
 438 structure for describing the *Lower Level Structural Elements* associated with the `Compo-`
 439 `nent`. `Component` is an abstract type XML element and will never appear directly in
 440 the MTConnect XML document. As an abstract type XML element, `Component` will be
 441 replaced in the XML document by specific `Component` types. XML elements represent-
 442 ing `Component` are described in *Section 5 - Component Structural Elements* and include
 443 elements such as `Axes`, `Controller`, and `Systems`.

Table 4: MTConnect Component Element

Element	Description	Occurrence
Component	An abstract XML element. Replaced in the XML document by types of <code>Component</code> elements representing physical parts and logical functions of a piece of equipment. There can be multiple types of <code>Component</code> XML elements in the document.	1..*

444 4.4.1 XML Schema Structure for Component

445 *Figure 3* represents the structure of a Component XML element showing the attributes
 446 defined for Component and the elements that **MAY** be associated with Component.

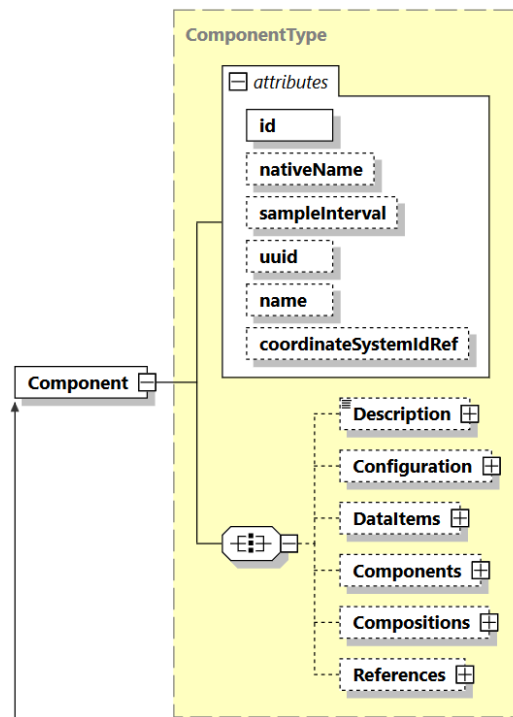


Figure 3: Component Diagram

447 4.4.2 Attribute for Component

448 *Table 5* defines the attributes that may be used to provide additional information for a
 449 Component type XML element.

Table 5: Attributes for Component

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 a specific physical or logical part of a piece of equipment.</p> <p><code>nativeName</code> is an optional attribute.</p>	0..1

Continuation of Table 5		
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 <code>Component</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 for a specific <code>Component</code> element is expected to be refreshed.</p> <p>The refresh rate for data from all <i>Lower Level</i> <code>Component</code> elements will be the same as for the parent <code>Component</code> element unless specifically overridden by another <code>sampleInterval</code> provided for the <i>Lower Level</i> <code>Component</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 ^{††}
sampleRate	DEPRECATED in MTConnect Version 1.2. Replaced by <code>sampleInterval</code> .	0..1 ^{†††}

Continuation of Table 5		
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 NMTOKEN 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 <i>Linear</i>), 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 NMTOKEN XML type.</p>	0..1
coordinateSystemIdRef	Specifies the CoordinateSystem for this Component and its children.	0..1

Notes: [†]While uuid **MUST** be provided for the Device element, it is optional for Component elements.

^{††}The sampleInterval is used to aid a client software application in in-

453 interpreting values provided by some *Data Entities*. This is the desired sample
 454 interval and may vary depending on the capabilities of the piece of equipment.
 455 †††Remains in schema for backwards compatibility.

456 4.4.3 Elements of Component

457 *Table 6* lists the elements defined to provide additional information for a Component
 458 type XML element.

Table 6: 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 [†]

459 Note: †At least one of Components, DataItems, or References **MUST** be
 460 provided.

461 4.4.3.1 Description for Component

462 *Figure 4* illustrates the structure of the `Description` XML element showing the at-
 463 tributes defined for `Description`. `Description` can contain any descriptive content
 464 of this `Component`. This element is defined to contain mixed content and additional
 465 XML elements (indicated by the `any` element) **MAY** be added to extend the schema for
 466 `Description`.

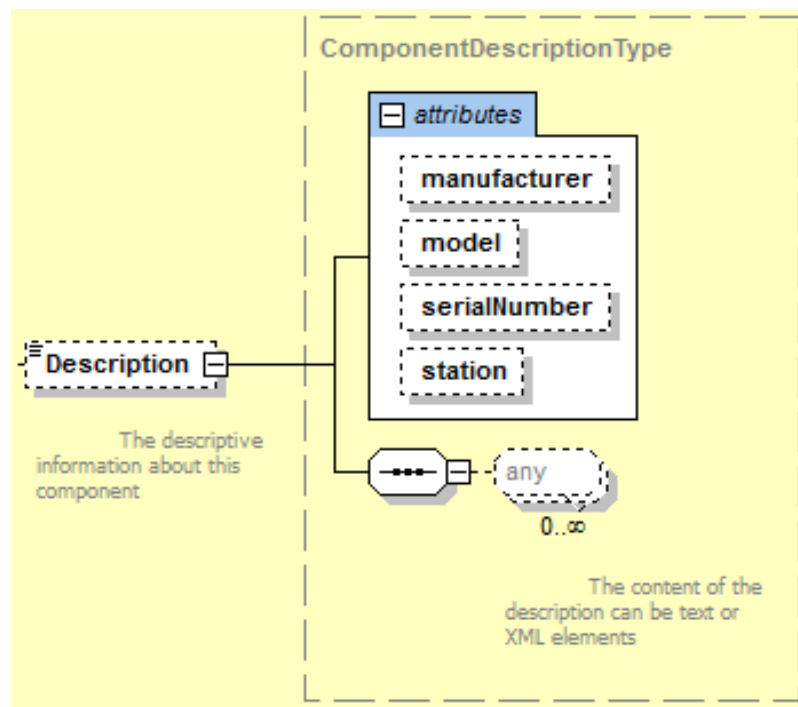


Figure 4: Description of Component Diagram

467 *Table 7* lists the attributes defined for the `Description` XML element.

Table 7: Attributes for Description for Component

Attribute	Description	Occurrence
manufacturer	The name of the manufacturer of the physical or logical part of a piece of equipment represented by the <code>Component</code> element. manufacturer is an optional attribute.	0..1
model	The model description of the physical part or logical function of a piece of equipment represented by the <code>Component</code> element. model is an optional attribute.	0..1

Continuation of Table 7		
Attribute	Description	Occurrence
serialNumber	The serial number associated with the physical part or logical function of a piece of equipment represented by the Component element. serialNumber is an optional attribute.	0..1
station	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. station is an optional attribute.	0..1

468 The content of Description **MAY** include any additional descriptive information the
 469 implementer chooses to include regarding the Component element. This content **SHOULD**
 470 be limited to information not included elsewhere in the MTConnectDevices XML doc-
 471 ument.

Example 3: Example of Description

```

472 1 <Description manufacturer="Example Co"
473 2     serialNumber="EXCO-TT-099PP-XXXX"> Advanced Pulse
474 3     watt-hour transducer with pulse output
475 4 </Description>

```

4.4.3.2 Configuration for Component

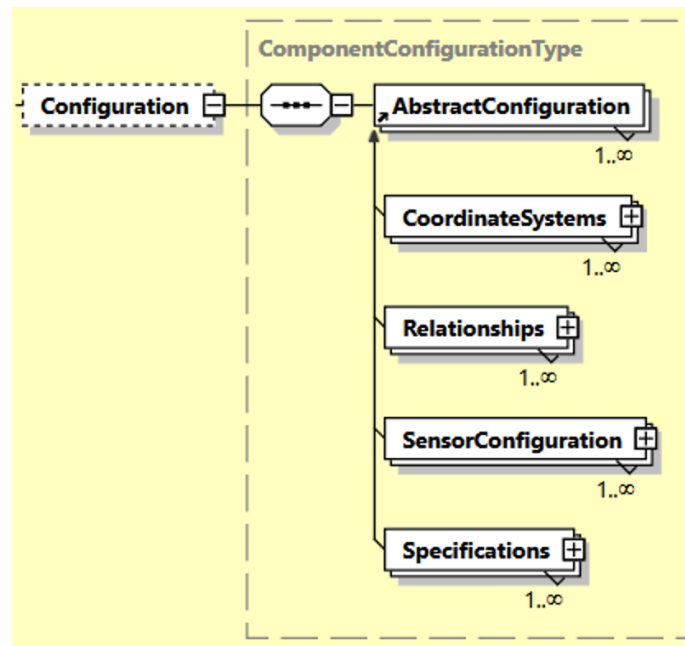
477 The Configuration XML element contains technical information about a component.
 478 Configuration **MAY** include any information describing the physical layout or func-
 479 tional characteristics of a component, such as capabilities, testing, installation, operation,
 480 calibration, or maintenance. Configuration **MAY** also include information represent-
 481 ing the inter-relationships between components within a piece of equipment.

Table 8: 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

482 Configuration data for Component is structured in the MTConnectDevices XML
 483 document as shown in *Figure 5*. AbstractConfiguration is an abstract type XML
 484 element. It will never appear in the XML document representing a piece of equipment.
 485 When Configuration is provided for a component, that type of Configuration
 486 will appear in the XML document.

487 See *Section 9 - Configuration* for details on the types of Configuration.

**Figure 5:** Component Configuration Diagram

488 4.4.3.3 DataItems for Component

489 DataItems is an XML container that provides structure for organizing the data reported
 490 by a piece of equipment that is associated with the Component.

491 See *Section 7 - Data Entities for Device* for details on the `DataItems` XML element.

492 **4.4.3.4 Components within Component**

493 The use of the XML container `Components` within a `Component` element provides
 494 the ability to further break down the structure of a `Component` element into even *Lower*
 495 *Level* physical and logical sub-parts. These *Lower Level* elements can add more clarity
 496 and granularity to the physical or logical structure of a piece of equipment and the data
 497 associated with that equipment.

498 This parent-child relationship can be extended down to any level necessary to fully de-
 499 scribe a piece of equipment. These *Lower Level* `Component` elements use the same XML
 500 structure as `Component` defined in *Section 4.4.1 - XML Schema Structure for Component*.

Example 4: Example of parent Component and Child Elements

```

501 1 <Devices>
502 2   <Device>
503 3     <Components>
504 4       <Axes> (Component)
505 5       <Components>
506 6         <Linear> (Component)
507 7         <Components>
508 8         <Etc. > (Component)

```

509 **4.4.3.5 Compositions for Component**

510 `Compositions` is an XML container used to organize the lowest level structural build-
 511 ing blocks contained within a `Component` as defined below.

512 **4.4.3.6 References for Component**

513 `References` is an XML container used to organize `Reference` elements associated
 514 with a `Component` element. See *Section 4.7 - References* for details on `References`.

515 **4.5 Compositions**

516 `Compositions` is an XML container that defines the lowest level structural building
 517 blocks contained within a `Component` element.

518 `Compositions` contains one or more `Composition` XML elements.

Table 9: 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

519 4.6 Composition

520 Composition XML elements are used to describe the lowest level physical building
521 blocks of a piece of equipment contained within a Component.

522 Composition provides the ability to organize information describing parts of its parent
523 Component. A Composition **MUST NOT** have child Components, Composi-
524 tions, or DataItems elements.

525 Composition elements are used to add more clarity and granularity to the data being
526 retrieved from a piece of equipment. The meaning of the data associated with a Com-
527 ponent may be enhanced by designating a specific Composition element associated
528 with that data.

529 An example of the additional detail provided when using Composition elements would
530 be:

531 A TEMPERATURE associated with a Linear type axis may be further clarified by ref-
532 erencing the MOTOR or AMPLIFIER type Composition element associated with that
533 axis, which differentiates the temperature of the motor from the temperature of the ampli-
534 fier.

535 Composition is a typed XML element and will always define a specific type of struc-
536 tural building block contained within a Component. XML elements representing the
537 types of Composition elements are described in *Section 6 - Composition Type Struc-*
538 *tural Elements* and include elements describing such basic building blocks as motors, am-
539 plifiers, filters, and pumps.

Example 5: Example of parent Component and child Composition elements

```

540 1 <Devices>
541 2   <Device>
542 3     <Components>
543 4       <Axes> (Component)
544 5       <Components>

```

```

545 6      <Linear> (Component)
546 7      <Compositions>
547 8      <Composition>
548 9      <Composition>
549 10     <Composition>

```

Table 10: MTConnect Composition Element

Element	Description	Occurrence
Composition	Composition is a functional part of a piece of equipment contained within a Component that MUST NOT be further decomposed into Components or Compositions.	1..*

550 4.6.1 XML Schema Structure for Composition

551 *Figure 6* illustrates a Composition XML element showing the attributes defined for
552 Composition and the elements that may be associated with Composition type XML
553 elements.

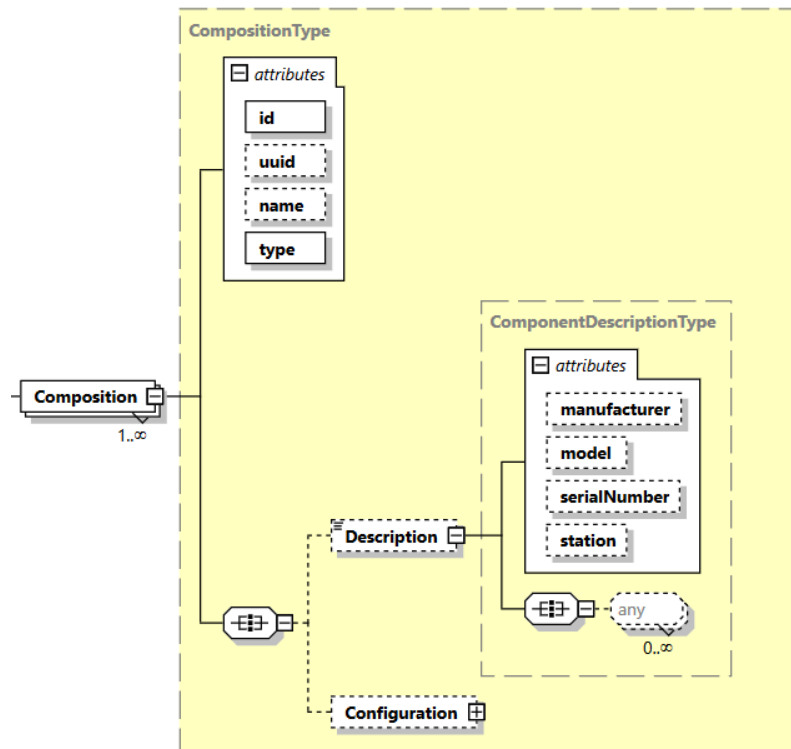


Figure 6: Composition Diagram

554 4.6.2 Attributes for Composition

555 *Table 11* defines the attributes that may be used to provide additional information for a
 556 *Composition* type XML element.

Table 11: Attributes for Composition

Attribute	Description	Occurrence
id	<p>The unique identifier for this element.</p> <p>id is a required attribute.</p> <p>An id MUST be unique across all the id attributes in the document.</p> <p>An XML ID-type.</p>	1

Continuation of Table 11		
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>If more than one Composition elements have the same type for the same Component, then the name attribute MUST be provided. Otherwise, the name attribute is optional.</p> <p>If provided, name MUST be unique within a Component element. name is 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

557 4.6.3 Elements of Composition

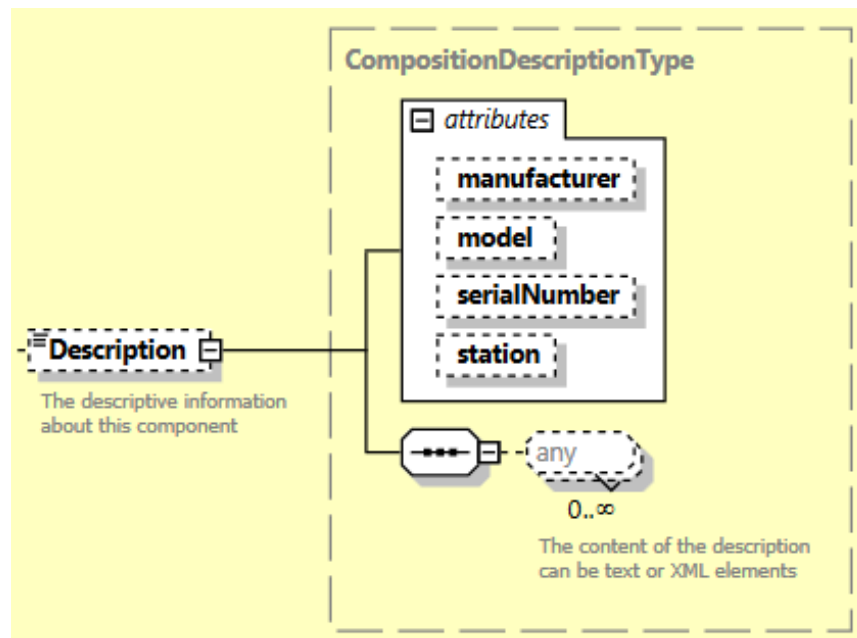
558 Table 12 lists the elements defined to provide additional information for a Composition
559 type XML element.

Table 12: Elements for Composition

Element	Description	Occurrence
Description	An element that can contain any descriptive content.	0..1
Configuration	An element that contains technical information about a piece of equipment describing its physical layout or functional characteristics. See <i>Section 9 - Configuration</i> for details on Configuration.	0..1

560 4.6.3.1 Description for Composition

561 *Figure 7* represents the structure of the Description XML element showing the at-
562 tributes defined for Description. Description can contain any descriptive content
563 for this Composition element. This element is defined to contain mixed content and
564 additional XML elements (indicated by the any element) **MAY** be added to extend the
565 schema for Description.

**Figure 7:** Description of Composition Diagram

566 *Table 13* lists the attributes defined for the Description XML element.

Table 13: Attributes for Description for Composition

Attribute	Description	Occurrence
manufacturer	The name of the manufacturer of the physical part of a piece of equipment represented by the <code>Composition</code> element. manufacturer is an optional attribute.	0..1
model	The model description of the physical part of a piece of equipment represented by the <code>Composition</code> element. model is an optional attribute.	0..1
serialNumber	The serial number associated with the physical part of a piece of equipment represented by the <code>Composition</code> element. serialNumber is an optional attribute.	0..1
station	The station where the physical part of a piece of equipment represented by the <code>Composition</code> element is located when it is part of a manufacturing unit or cell with multiple stations. station is an optional attribute.	0..1

567 The content of `Description` **MAY** include any additional descriptive information the
568 implementer chooses to include regarding the `Composition` element. This content
569 **SHOULD** be limited to information not included elsewhere in the `MTConnectDevices`
570 XML document.

Example 6: Example of Description

```

571 1 <Description manufacturer="Example Co"
572 2     serialNumber="A124FFF" station="2"> Spindle motor
573 3     associated with Path 2.
574 4 </Description>

```

575 4.7 References

576 `References` is an XML container that organizes pointers to information defined else-
577 where within the XML document for a piece of equipment.

References may be modeled as part of a Device, Component or Interface type *Structural Element*.

References contains one or more Reference XML elements.

Table 14: 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 <i>Interface</i> element.	0..1

4.8 Reference

Reference is a pointer to information that is associated with another *Structural Element* defined elsewhere in the XML document for a piece of equipment. That information may be data from the other element or the entire structure of that element.

Reference is an efficient method to associate information with an element without duplicating any of the data or structure. For example, a Bar Feeder System may make a request for the BarFeederInterface and receive all the relevant data for the interface and the associated spindle (Rotary element) that is referenced as part of the BarFeederInterface.

Reference is an abstract type XML element and will never appear directly in the MTConnect XML document. As an abstract type XML element, Reference will be replaced in the XML document by a specific Reference type. The current supported types of Reference are DataItemRef and ComponentRef XML elements.

Figure 8 represents the structure of the Reference XML element.

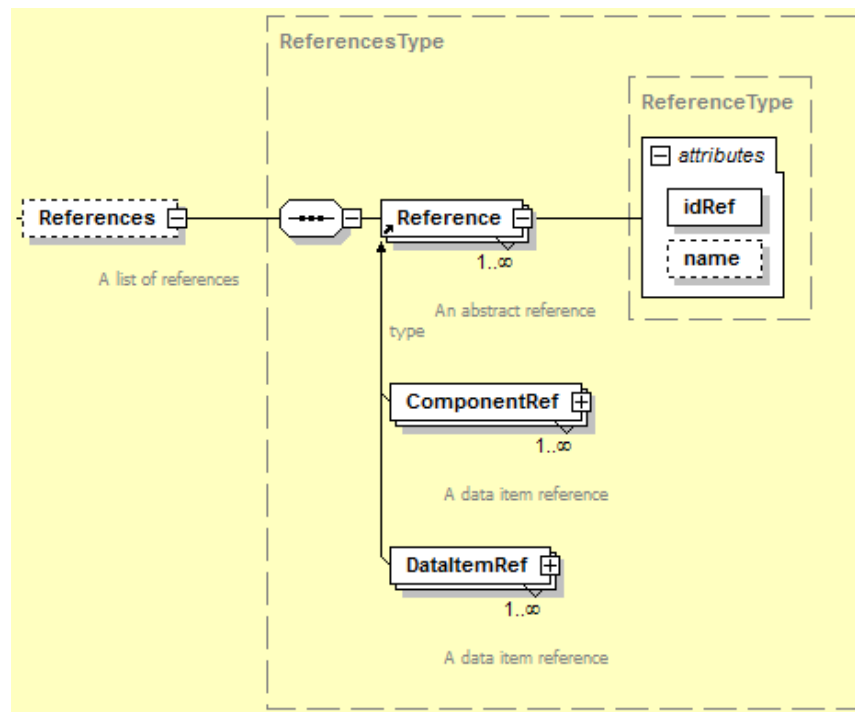


Figure 8: Reference Diagram

595 4.8.1 ComponentRef

596 ComponentRef XML element is a pointer to all of the information associated with an-
 597 other *Structural Element* defined elsewhere in the XML document for a piece of equip-
 598 ment. ComponentRef allows all of the information (*Lower Level Components* and all
 599 *Data Entities*) that is associated with the other *Structural Element* to be directly associated
 600 with this XML element.

601 *Figure 9* represents the structure of a ComponentRef XML element showing the at-
 602 tributes defined for ComponentRef.

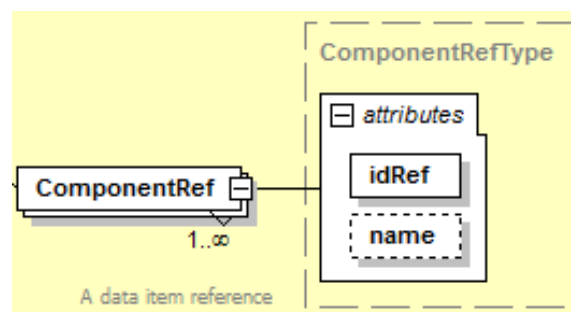


Figure 9: ComponentRef Diagram

603 *Table 15* lists the attributes defined for the `ComponentRef` element.

Table 15: Attributes for `ComponentRef`

Attribute	Description	Occurrence
<code>idRef</code>	A pointer to the <code>id</code> attribute of the <code>Component</code> that contains the information to be associated with this XML element. <code>idRef</code> is a required attribute.	1
<code>name</code>	The optional name of the <code>ComponentRef</code> . Only informative. <code>name</code> is an NMTOKEN XML type.	0..1

604 4.8.2 DataItemRef

605 `DataItemRef` XML element is a pointer to a *Data Entity* associated with another *Struc-*
 606 *tural Element* defined elsewhere in the XML document for a piece of equipment. `DataItem-`
 607 `Ref` allows the data associated with a data item defined in another *Structural Element* to
 608 be directly associated with this XML element.

609 *Figure 10* represents the structure of a `DataItemRef` XML element showing the at-
 610 tributes defined for `DataItemRef`.

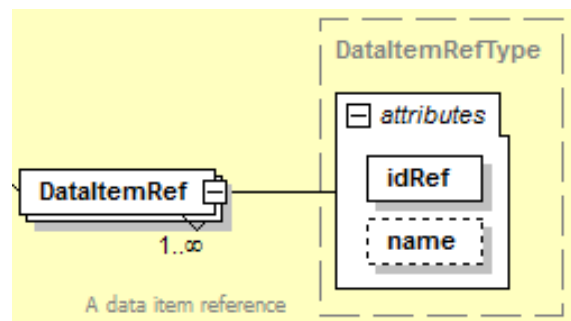


Figure 10: `DataItemRef` Diagram

611 *Table 16* lists the attributes defined for the `DataItemRef` element.

Table 16: Attributes for DataItemRef

Attribute	Description	Occurrence
idRef	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
name	The optional name of the <code>DataItemRef</code> . Only informative. <code>name</code> is an NMTOKEN XML type.	0..1

612 5 Component Structural Elements

613 Component *Structural Elements* are XML containers used to represent physical parts or
 614 logical functions of a piece of equipment.

615 Component *Structural Elements* are defined into two major categories:

- 616 • *Top Level* Component elements are used to group the *Structural Elements* repre-
 617 senting the most significant physical or logical functions of a piece of equipment.
 618 The *Top Level* Component elements provided in an MTConnectDevices docu-
 619 ment **SHOULD** be restricted to those defined in Table 17. However, these *Top Level*
 620 Component elements **MAY** also be used as *Lower Level* Component elements;
 621 as required.
- 622 • *Lower Level* Component elements are used to describe the sub-parts of the par-
 623 ent Component to provide more clarity and granularity to the physical or logical
 624 structure of the *Top Level* Component elements.

625 This section of the *Devices Information Model* provides guidance for the most common re-
 626 lationships between *Top Level* Component elements and *Lower Level* child components.
 627 However, all Component elements **MAY** be used in any configuration, as required, to
 628 fully describe a piece of equipment.

629 As described in Section 4 - *Structural Elements for MTConnectDevices*, Component is
 630 an abstract type *Structural Element* within the *Devices Information Model* and will never
 631 appear directly in the MTConnectDevices XML document. As abstract type XML
 632 elements, Component will be replaced in the XML document by a specific Component
 633 type.

634 Table 17 defines the *Top Level* Component elements available to describe a piece of
 635 equipment.

Table 17: Top Level Component Elements

Top Level Component Element ^{††}	Description
Axis	Axis <i>organizes</i> Axis component types.
Controller	Controller represents the computational regulation and management function of a piece of equipment.

Continuation of Table 17	
Top Level Component Element ^{††}	Description
Systems	Systems <i>organizes</i> System component types.
Auxiliaries	Auxiliaries <i>organizes</i> Auxiliary component types.
Resources	Resources <i>organizes</i> Resource component types.
Interfaces	Interfaces <i>organizes</i> Interface component types.
Adapters	Adapters <i>organizes</i> Adapter component types.
Structures	Structures <i>organizes</i> Structure component types.

Note: ^{††}The following components have been relocated or redefined since they are not classified as restricted *Top Level* components:

- Power was **DEPRECATED** in MTConnect Version 1.1 and was replaced by the *Data Entity* called AVAILABILITY.

- Door has been redefined as a *Lower Level* component of a parent Component element or as a Composition element.

- Actuator, due to its uniqueness, has been redefined as a piece of equipment with the ability to be represented as a *Lower Level* component of a parent Component element or as a Composition element.

- Sensor, due to its uniqueness, has been redefined as a piece of equipment with the ability to be represented as a *Lower Level* component of a parent Component element (See *Section 9.1 - Sensor* for further detail).

- Stock has been redefined as a *Lower Level* component of the Resources *Top Level* Component element.

The common relationship between the *Top Level* Component elements and the *Lower Level* child Component elements are described below. It should be noted that as the MTConnect Standard evolves, more Component types will be added to organize information for new types of equipment and/or new physical or logical sub-parts of equipment.

5.1 Axes

Axes *organizes* Axis component types.

656 5.1.1 Axis

657 Axis is an abstract Component that represents linear or rotational motion for a piece of
658 equipment.

659 The Linear axis Component represents linear motion, and the Rotary axis Compo-
660 nent represents rotational motion.

661 In robotics, the term "Axis" is synonymous with "Joint". A "Joint" is the connection
662 between two parts of the structure that move in relation to each other.

663 Linear and Rotary components **MUST** have a name attribute that **MUST** follow
664 the conventions described below. Use the `nativeName` attribute for the manufacturer's
665 name of the axis if it differs from the assigned `name`.

666 *MTConnect* has two high-level classes for automation equipment as follows: (1) Equip-
667 ment that controls cartesian coordinate axes and (2) Equipment that controls articulated
668 axes. There are ambiguous cases where some machines exhibit both characteristics; when
669 this occurs, the primary control system's configuration determines the classification.

670 Examples of cartesian coordinate equipment are CNC Machine Tools, Coordinate mea-
671 surement machines, as specified in ISO 841, and 3D Printers. Examples of articulated
672 automation equipment are Robotic systems as specified in ISO 8373.

673 The following sections define the designation of names for the axes and additional guid-
674 ance when selecting the correct scheme to use for a given piece of equipment.

675 5.1.2 Cartesian Coordinate Naming Conventions

676 A Three-Dimensional Cartesian Coordinate control system organizes its axes orthogonally
677 relative to a machine coordinate system where the manufacturer of the equipment specifies
678 the origin.

679 Axes name **SHOULD** comply with ISO 841, if possible.

680 5.1.2.1 Linear Motion

681 A piece of equipment **MUST** represent prismatic motion using a Linear axis Compo-
682 nent and assign its name using the designations X, Y, and Z. A Linear axis name
683 **MUST** append a monotonically increasing suffix when there are more than one parallel
684 axes; for example, X2, X3, and X4.

685 5.1.2.2 Rotary Motion

686 *MTConnect* **MUST** assign the name to Rotary axes exhibiting rotary motion using A,
 687 B, and C. A Rotary axis name **MUST** append a monotonically increasing suffix when
 688 more than one Rotary axis rotates around the same Linear axis; for example, A2, A3,
 689 and A4.

690 5.1.3 Articulated Machine Control Systems

691 An articulated control system's axes represent the connecting linkages between two ad-
 692 jacent rigid members of an assembly. The Linear axis represents prismatic motion,
 693 and the Rotary axis represents the rotational motion of the two related members. The
 694 control organizes the axes in a kinematic chain from the mounting surface (base) to the
 695 end-effector or tooling.

696 5.1.4 Articulated Machine Axis Names

697 The axes of articulated machines represent forward kinematic relationships between me-
 698 chanical linkages. Each axis is a connection between linkages, also referred to as joints,
 699 and **MUST** be named using a J followed by a monotonically increasing number; for ex-
 700 ample, J1, J2, J3. The numbering starts at the base axis connected or closest to the
 701 mounting surface, J1, incrementing to the mechanical interface, Jn, where n is the num-
 702 ber of the last axis. The chain forms a parent-child relationship with the parent being the
 703 axis closest to the base.

704 A machine having an axis with more than one child **MUST** number each branch using its
 705 numeric designation followed by a branch number and a monotonically increasing number.
 706 For example, if J2 has two children, the first child branch **MUST** be named J2.1.1 and
 707 the second child branch J2.2.1. A child of the first branch **MUST** be named J2.1.2,
 708 incrementing to J2.1.n, where J2.1.n is the number of the last axis in that branch.

709 5.1.5 Rotary Component

710 A Rotary axis represents rotation about a fixed axis.

711 5.1.6 Linear Component

712 A `Linear` axis represents prismatic motion along a fixed axis.

713 5.2 Controller

714 `Controller` represents the computational regulation and management function of a
715 piece of equipment.

716 Typical types of controllers for a piece of equipment include CNC (Computer Numerical
717 Control), PAC (Programmable Automation Control), IPC (Industrialized Computer), or IC
718 (Imbedded Computer).

719 Note: MTConnect Version 1.1.0 and later implementations **SHOULD** use a *Lower*
720 *Level* Component element called `Path` to represent an individual tool path or
721 other independent function within a `Controller` element. When the `Con-`
722 `troller` element is capable of executing more than one simultaneous and in-
723 dependent programs, the implementation **MUST** specify a *Lower Level* `Path`
724 element representing each of the independent functions of the `Controller`.

725 5.2.1 Path

726 `Path` is a Component that represents the information for an independent operation or
727 function within a `Controller`. For many types of equipment, `Path` represents a set
728 of `Axes`, one or more `Program` elements, and the data associated with the motion of a
729 control point as it moves through space. However, it **MAY** also represent any independent
730 function within a `Controller` that has unique data associated with that function.

731 `Path` **SHOULD** provide an `EXECUTION` data item to define the operational state of the
732 `Controller` component of the piece of equipment.

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

736 5.3 Systems

737 `Systems` *organizes* `System` component types.

738 **5.3.1 System**

739 `System` is an abstract `Component` that represents part(s) of a piece of equipment that is
740 permanently integrated into the piece of equipment.

741 **5.3.2 Hydraulic**

742 `Hydraulic` is a `System` that represents the information for a system comprised of all
743 the parts involved in moving and distributing pressurized liquid throughout the piece of
744 equipment.

745 **5.3.3 Pneumatic**

746 `Pneumatic` is a `System` that uses compressed gasses to actuate components or do work
747 within the piece of equipment.

748 Note: Actuation is usually performed using a cylinder.

749 **5.3.4 Coolant**

750 `Coolant` is a `System` that represents the information for a system comprised of all the
751 parts involved in distribution and management of fluids that remove heat from a piece of
752 equipment.

753 **5.3.5 Lubrication**

754 `Lubrication` is a `System` that represents the information for a system comprised of
755 all the parts involved in distribution and management of fluids used to lubricate portions
756 of the piece of equipment.

757 **5.3.6 Electric**

758 `Electric` is a `System` that represents the information for the main power supply for
759 device piece of equipment and the distribution of that power throughout the equipment.

760 The electric system will provide all the data with regard to electric current, voltage, fre-
 761 quency, etc. that applies to the piece of equipment as a functional unit. Data regarding
 762 electric power that is specific to a Component will be reported as *Data Entities* for that
 763 specific Component.

764 **5.3.7 Enclosure**

765 Enclosure is a System that represents the information for a structure used to contain or
 766 isolate a piece of equipment or area. The Enclosure system may provide information
 767 regarding access to the internal components of a piece of equipment or the conditions
 768 within the enclosure. For example, Door may be defined as a *Lower Level* Component
 769 or Composition element of the Enclosure system.

770 **5.3.8 Protective**

771 Protective is a System that represents the information for those functions that detect
 772 or prevent harm or damage to equipment or personnel. Protective does not include
 773 the information relating to the Enclosure system.

774 **5.3.9 ProcessPower**

775 ProcessPower is a System that represents the information for a power source associ-
 776 ated with a piece of equipment that supplies energy to the manufacturing process separate
 777 from the Electric system. For example, this could be the power source for an EDM
 778 machining process, an electroplating line, or a welding system.

779 **5.3.10 Feeder**

780 Feeder is a System that represents the information for a system that manages the de-
 781 livery of materials within a piece of equipment. For example, this could describe the wire
 782 delivery system for an EDM or welding process; conveying system or pump and valve sys-
 783 tem distributing material to a blending station; or a fuel delivery system feeding a furnace.

784 **5.3.11 Dielectric**

785 `Dielectric` is a `System` that represents the information for a system that manages a
786 chemical mixture used in a manufacturing process being performed at that piece of equip-
787 ment. For example, this could describe the dielectric system for an EDM process or the
788 chemical bath used in a plating process.

789 **5.3.12 EndEffector**

790 `EndEffector` is a `System` that represents the information for those functions that form
791 the last link segment of a piece of equipment. It is the part of a piece of equipment that
792 interacts with the manufacturing process.

793 **5.3.13 WorkEnvelope**

794 `WorkEnvelope` is a `System` that organizes information about the physical process ex-
795 ecution space within a piece of equipment. The `WorkEnvelope` **MAY** provide informa-
796 tion regarding the physical workspace and the conditions within that workspace.

797 **5.3.14 Heating**

798 `Heating` is a `System` used to deliver controlled amounts of heat to achieve a target
799 temperature at a specified heating rate.

800 Note: As an example, the energy delivery method can be either through electric heaters
801 or gas burners.

802 **5.3.15 Cooling**

803 `Cooling` is a `System` used to to extract controlled amounts of heat to achieve a target
804 temperature at a specified cooling rate.

805 Note: As an example, the energy extraction method can be via cooling water pipes
806 running through the chamber.

807 **5.3.16 Pressure**

808 Pressure is a System that delivers compressed gas or fluid and controls the pressure
809 and rate of pressure change to a desired target set-point.

810 Note: For example, the delivery method can be a Compressed Air or N2 tank that is piped
811 via an inlet valve to the chamber.

812 **5.3.17 Vacuum**

813 Vacuum is a System that evacuates gases and liquids from an enclosed and sealed space
814 to a controlled negative pressure or a molecular density below the prevailing atmospheric
815 level.

816 **5.4 Auxiliaries**

817 Auxiliaries *organizes* Auxiliary component types.

818 **5.4.1 Auxiliary**

819 Auxiliary is an abstract Component that represents removable part(s) of a piece of
820 equipment providing supplementary or extended functionality.

821 **5.4.2 Loader**

822 Loader is an Auxiliary comprised of all the parts involved in moving and distributing
823 materials, parts, tooling, and other items to or from a piece of equipment.

824 **5.4.2.1 BarFeeder**

825 BarFeeder is a Loader involved in delivering bar stock to a piece of equipment.

826 **5.4.3 WasteDisposal**

827 WasteDisposal is an Auxiliary that represents the information for a unit comprised
828 of all the parts involved in removing manufacturing byproducts from a piece of equipment.

829 **5.4.4 ToolingDelivery**

830 ToolingDelivery is an Auxiliary that represents the information for a unit in-
831 volved in managing, positioning, storing, and delivering tooling within a piece of equip-
832 ment.

833 **5.4.4.1 AutomaticToolChanger**

834 AutomaticToolChanger is a ToolingDelivery that represents a tool delivery
835 mechanism that moves tools between a ToolMagazine and a *Spindle* or a Turret.
836 An AutomaticToolChanger may also transfer tools between a location outside of a
837 piece of equipment and a ToolMagazine or Turret.

838 **5.4.4.2 ToolMagazine**

839 ToolMagazine is a ToolingDelivery that represents a tool storage mechanism that
840 holds any number of tools. Tools are located in POTS. POTS are moved into position to
841 transfer tools into or out of the ToolMagazine by an AutomaticToolChanger.

842 **5.4.4.3 Turret**

843 Turret is a ToolingDelivery that represents a tool mounting mechanism that holds
844 any number of tools. Tools are located in STATIONS . Tools are positioned for use in the
845 manufacturing process by rotating the Turret.

846 **5.4.4.4 GangToolBar**

847 GangToolBar is a ToolingDelivery that represents a tool mounting mechanism
848 that holds any number of tools. Tools are located in STATIONS. Tools are positioned for
849 use in the manufacturing process by linearly positioning the GangToolBar.

850 **5.4.4.5 ToolRack**

851 ToolRack is a ToolingDelivery that represents a linear or matrixed tool storage
852 mechanism that holds any number of tools. Tools are located in STATIONS.

853 **5.4.5 Environmental**

854 Environmental is an Auxiliary that represents the information for a unit or func-
855 tion involved in monitoring, managing, or conditioning the environment around or within
856 a piece of equipment.

857 **5.4.6 Sensor**

858 Sensor is is an Auxiliary that represents the information for a piece of equipment that
859 responds to a physical stimulus and transmits a resulting impulse or value from a sensing
860 unit. When modeled as a component of Auxiliaries, sensor **SHOULD** represent an
861 integrated *sensor unit* system that provides signal processing, conversion, and communi-
862 cations. A *sensor unit* may have multiple *sensing elements*; each representing the data for
863 a variety of measured values. See *Section 9.1.2 - Sensor Unit* for more details on *sensor*
864 *unit*.

865 Note: If modeling an individual sensor, then sensor should be associated with the
866 component that the measured value is most closely associated. See *Section 5.9.3*
867 - *Sensor*.

868 **5.4.7 Deposition**

869 Deposition is an Auxiliary that represents the information for a system that man-
870 ages the addition of material or state change of material being performed in an additive
871 manufacturing process. For example, this could describe the portion of a piece of equip-
872 ment that manages a material extrusion process or a vat polymerization process.

873 **5.5 Resources**

874 Resources *organizes* Resource component types.

875 **5.5.1 Resource**

876 `Resource` is an abstract `Component` that represents materials or personnel involved in
877 a manufacturing process.

878 **5.5.2 Materials**

879 `Materials` provides information about materials or other items consumed or used by the
880 piece of equipment for production of parts, materials, or other types of goods. `Materi-`
881 `als` also represents parts or part stock that are present at a piece of equipment or location
882 to which work is applied to transform the part or stock material into a more finished state.

883 **5.5.2.1 Stock**

884 `Stock` is a `Resource` that represents the information for the material that is used in a
885 manufacturing process and to which work is applied in a machine or piece of equipment
886 to produce parts.

887 `Stock` may be either a continuous piece of material from which multiple parts may be
888 produced or it may be a discrete piece of material that will be made into a part or a set of
889 parts.

890 **5.5.3 Personnel**

891 `Personnel` is a `Resource` that provides information about an individual or individuals
892 who either control, support, or otherwise interface with a piece of equipment.

893 **5.6 Interfaces**

894 `Interfaces` *organizes* `Interface` component types.

895 **5.6.1 Interface**

896 `Interface` is a `Component` that coordinates actions and activities between pieces of
897 equipment.

898 See *MTConnect Standard: Part 5.0 - Interfaces* for detailed information on `Interface`.

899 5.7 Adapters

900 `Adapters` *organizes* `Adapter` component types.

901 5.7.1 Adapter

902 `Adapter` is a `Component` that represents the connectivity state of a data source for the
903 *MTConnect Agent*.

904 It **MAY** contain additional telemetry about the data source and source-specific informa-
905 tion.

906 5.8 Structures

907 `Structures` *organizes* `Structure` component types.

908 5.8.1 Structure

909 `Structure` is a `Component` that represents the part(s) comprising the rigid bodies of
910 the piece of equipment.

911 5.8.2 Link

912 `Link` is a `Structure` providing a connection between `Components`.

913 5.9 Other Components

914 While most component elements **SHOULD** be modeled in a specific manner, there are
915 some types of component elements that are used ubiquitously in equipment and **MAY** be
916 associated with any number of different types of parent component elements.

917 These components **MAY** be modeled as *Lower Level* components of the Parent Element.

918 5.9.1 Actuator

919 Actuator is a Component that represents the information for an apparatus for moving
 920 or controlling a mechanism or system. It takes energy usually provided by air, electric
 921 current, or liquid and converts the energy into some kind of motion.

922 5.9.2 Door

923 Door is a Component that represents the information for a mechanical mechanism or
 924 closure that can cover, for example, a physical access portal into a piece of equipment. The
 925 closure can be opened or closed to allow or restrict access to other parts of the equipment.

926 When Door is represented as a Component, it **MUST** have a data item called DOOR_
 927 STATE to indicate if the door is OPEN, CLOSED, or UNLATCHED. A Component **MAY**
 928 contain multiple Door components.

929 5.9.3 Sensor

930 Sensor is a Component that represents the information for a piece of equipment that
 931 responds to a physical stimulus and transmits a resulting impulse or value. If modeling
 932 individual sensors, then sensor should be associated with the component that the measured
 933 value is most closely associated.

934 See *Section 9.1 - Sensor* for more details on the use of Sensor.

935 5.9.4 Processes

936 Processes *organizes* information describing the manufacturing process being executed
 937 on a piece of equipment.

938 5.9.4.1 ProcessOccurrence

939 ProcessOccurrence is a Component that *organizes* information about the execution
 940 of a specific process that takes place at a specific place and time, such as a specific instance
 941 of part-milling occurring at a specific timestamp.

942 PROCESS_OCCURRENCE_ID **MUST** be defined for PartOccurrence.

943 Suggested DataItem types for ProcessOccurrence are: PROCESS_AGGREGATE_
944 ID, PROCESS_KIND_ID, PROCESS_TIME, USER, PROGRAM, and PART_UNIQUE_
945 ID.

946 5.9.5 Parts

947 Parts *organizes* information for *Parts* being processed by a piece of equipment.

948 5.9.5.1 PartOccurrence

949 PartOccurrence is a Component that *organizes* information about a specific part as
950 it exists at a specific place and time, such as a specific instance of a bracket at a specific
951 timestamp.

952 *Part* is defined as a discrete item that has both defined and measurable physical charac-
953 teristics including mass, material and features and is created by applying one or more
954 manufacturing process steps to a workpiece.

955 PART_ID **MUST** be defined for PartOccurrence.

956 Suggested DataItem types for PartOccurrence are: PART_UNIQUE_ID, PART_
957 GROUP_ID, PART_KIND_ID, PART_COUNT, PART_STATUS, PROCESS_TIME, PRO-
958 CESS_OCCURRENCE_ID, and USER.

959 5.9.6 Lock

960 Lock is a Component that represents a mechanism which physically prohibits a device
961 or component from opening or operating.

962 6 Composition Type Structural Elements

963 Composition *Structural Elements* are used to describe the lowest level physical build-
 964 ing blocks of a piece of equipment contained within a Component. By referencing a spe-
 965 cific Composition element, further clarification and meaning to data associated with a
 966 specific Component can be achieved.

967 Both Component and Composition elements are *Lower Level* child Component
 968 XML elements representing the sub-parts of the parent Component. However, there are
 969 distinct differences between Component and Composition type elements.

970 Component elements may be further defined with *Lower Level* Component elements
 971 and may have associated *Data Entities*.

972 Composition elements represent the lowest level physical part of a piece of equipment.
 973 They **MUST NOT** be further defined with *Lower Level* Component elements and they
 974 **MUST NOT** have *Data Entities* directly associated with them. They do provide additional
 975 information that can be used to enhance the specificity of *Data Entities* associated with the
 976 parent Component.

977 *Table 18* defines Composition type elements that are currently available to describe
 978 sub-parts of a Component element.

Table 18: 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 18	
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.
COOLING_TOWER	A heat exchange system that uses a fluid to transfer heat to the atmosphere.
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 to measure position.
EXPIRED_POT	A POT for a tool that is no longer useable for removal from a ToolMagazine or Turret.
EXPOSURE_UNIT	A mechanism for emitting a type of radiation

Continuation of Table 18	
Element Type	Description
EXTRUSION_UNIT	A mechanism for dispensing liquid or powered materials
FAN	Any mechanism for producing a current of air.
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	<p>A mechanism that measures linear motion or position.</p> <p>DEPRECATION WARNING : May be deprecated in the future. Recommend using ENCODER.</p>
MOTOR	A mechanism that converts electrical, pneumatic, or hydraulic energy into mechanical energy.
OIL	A viscous liquid.
POT	A tool storage location associated with a ToolMagazine or AutomaticToolChanger.
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.

Continuation of Table 18	
Element Type	Description
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
REMOVAL_POT	A POT for a tool to be removed from a ToolMagazine or Turret to a location outside of the piece of equipment.
RETURN_POT	A POT for a tool removed from <i>Spindle</i> or Turret and awaiting for return to a ToolMagazine.
SENSING_ELEMENT	A mechanism that provides a signal or measured value.
SPREADER	A mechanism for flattening or spreading materials
STAGING_POT	A POT for a tool awaiting transfer to a ToolMagazine or Turret from outside of the piece of equipment.
STATION	A storage or mounting location for a tool associated with a Turret, GangToolBar, or ToolRack.
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.
TRANSFER_ARM	A mechanism for physically moving a tool from one location to another.
TRANSFER_POT	A POT for a tool awaiting transfer from a ToolMagazine to <i>Spindle</i> or Turret.

Continuation of Table 18	
Element Type	Description
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.
WORKPIECE	An object or material on which a form of work is performed.

979 Note: As the MTConnect Standard evolves, more Composition types will be
980 added.

981 7 Data Entities for Device

982 In the MTConnectDevices XML document, *Data Entities* are XML elements that de-
 983 scribe data that can be reported by a piece of equipment and are associated with *Device*
 984 and *Component Structural Elements*. While the *Data Entities* describe the data that can
 985 be reported by a piece of equipment in the MTConnectDevices document, the actual
 986 data values are provided in the *Streams Information Model*. See *MTConnect Standard:*
 987 *Part 3.0 - Streams Information Model* for detail on the reported values.

988 Each *Data Entity* **SHOULD** be modeled in the MTConnectDevices document such
 989 that it is associated with the *Structural Element* that the reported data directly applies.

990 When *Data Entities* are associated with a *Structural Element*, they are organized in a
 991 *DataItems* XML element. *DataItems* is a container type XML element. *DataItems*
 992 provides the structure for organizing individual *DataItem* elements that represent each
 993 *Data Entity*. The *DataItems* container is comprised of one or more *DataItem* type
 994 XML element(s).

995 *DataItem* describes specific types of *Data Entities* that represent a numeric value, a
 996 functioning state, or a health status reported by a piece of equipment. *DataItem* provides
 997 a detailed description for each *Data Entity* that is reported; it defines the type of data being
 998 reported and an array of optional attributes that further describe that data. The different
 999 types of *DataItem* elements are defined in *Section 8 - Listing of Data Items*.

1000 *Figure 11* demonstrates the relationship between *Data Entities* (*DataItem*) and the var-
 1001 ious *Structural Elements* in the MTConnectDevices XML document.

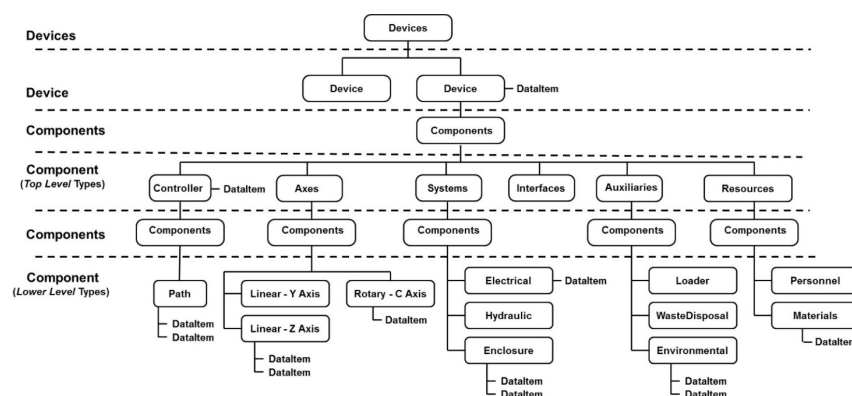


Figure 11: Example Data Entities for Device (*DataItem*)

1002 7.1 DataItems

1003 The DataItems XML element is the first, or highest, level container for the *Data Entities*
 1004 associated with a Device or Component XML element. DataItems **MUST** contain
 1005 only DataItem type elements. DataItems **MUST** contain at least one DataItem
 1006 type element, but **MAY** contain multiple DataItem type elements.

Table 19: 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

1007 7.2 DataItem

1008 A DataItem XML element represents each *Data Entity* that **MAY** be reported by a piece
 1009 of equipment through an *Agent*. DataItem provides a detailed description for each *Data*
 1010 *Entity* that is reported and defines the type of data being reported along with an array of
 1011 optional attributes that further define that data. XML elements representing DataItem
 1012 will include elements such as TEMPERATURE, PRESSURE, and VELOCITY.

Table 20: MTConnect DataItem Element

Element	Description	Occurrence
DataItem	<i>Data Entity</i> describing a piece of information reported about a piece of equipment.	1..*

1013 7.2.1 XML Schema Structure for DataItem

1014 *Figure 12* represents the structure of a DataItem XML element showing the attributes
 1015 defined for DataItem and the elements that may be associated with DataItem type
 1016 XML elements.

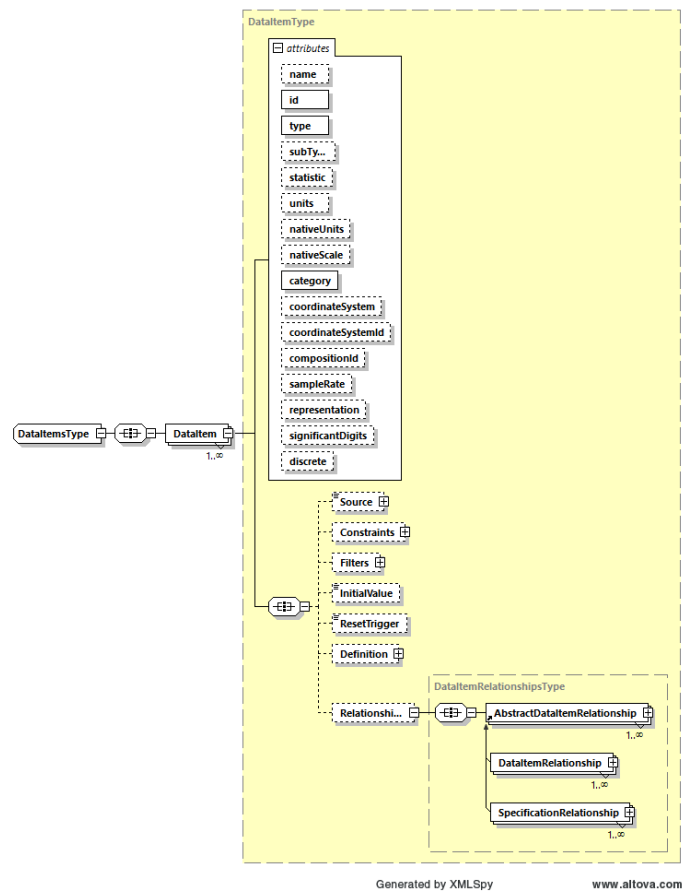


Figure 12: DataItem Diagram

1017 7.2.2 Attributes for DataItem

1018 *Table 21* lists the attributes defined to provide information for a DataItem type XML
 1019 element.

1020 DataItem **MUST** specify the type of data being reported, the id of the DataItem, and
 1021 the category of the DataItem.

Table 21: 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 21		
Attribute	Description	Occurrence
<code>statistic</code>	<p>Describes the type of statistical calculation performed on a series of data samples to provide the reported data value.</p> <p><code>statistic</code> is an optional attribute.</p> <p>Examples of <code>statistic</code> are AVERAGE, MINIMUM, MAXIMUM, ROOT_MEAN_SQUARE, RANGE, MEDIAN, MODE, and STANDARD_DEVIATION.</p>	0..1
<code>units</code>	<p>The unit of measurement for the reported value of the data item.</p> <p><code>units</code> is an optional attribute.</p> <p>Data items in the <code>Sample</code> 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
<code>nativeUnits</code>	<p>The native units of measurement for the reported value of the data item.</p> <p><code>nativeUnits</code> 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 21		
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

Continuation of Table 21		
Attribute	Description	Occurrence
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
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 Version 1.5), DATA_SET, TABLE, and VALUE are defined in Section 7.2.2.12 - <i>representation Attribute for DataItem</i>.</p> <p>If representation is not specified, it MUST be determined to be VALUE.</p>	0..1

Continuation of Table 21		
Attribute	Description	Occurrence
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
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
coordinateSystemIdRef	The associated <code>CoordinateSystem</code> context for the <code>DataItem</code> .	0..1

1022 7.2.2.1 name Attribute for DataItem

1023 The attribute `name` is provided as an additional human readable identifier for a data item.
 1024 It is not required and is implementation dependent.

1025 7.2.2.2 id Attribute for DataItem

1026 Each `DataItem` element **MUST** be identified with an `id`. The `id` attribute **MUST** be
 1027 unique across the entire `MTConnectDevices` document for a piece of equipment, in-
 1028 cluding the identifiers for all *Structural Elements*. This unique `id` provides the information

1029 required by a client software application to uniquely identify each *Data Entity*.

1030 For example, an XML document may provide three different *Data Entities* representing
1031 the position of the axes on a machine (x axis position, y axis position, and z axis position).
1032 All three may be modeled in the XML document as POSITION type data items for the
1033 Axes components. The unique id allows the client software application to distinguish
1034 the data for each of the axes.

1035 7.2.2.3 type and subType Attributes for DataItem

1036 The attribute `type` specifies the kind of data that is represented by the data item.

1037 The attribute `type` **MUST** be specified for every data item.

1038 A data item **MAY** further qualify the data being reported by specifying a `subType`.
 1039 `subType` is required for certain data item types. For example, `POSITION` has the
 1040 `subType` of `ACTUAL` and `PROGRAMMED`. Both data values can be represented in the
 1041 document as two separate and different `DataItem` XML elements – `POSITION` with
 1042 `subType` `ACTUAL` and `POSITION` with `subType` `PROGRAMMED`.

1043 The `type` and `subType` **SHOULD** be used to further identify the meaning of the `DataItem`
 1044 associated with a `Component` element when a `subType` is applicable. There **SHOULD**
 1045 **NOT** be more than one `DataItem` with the same `type`, `subType`, and `composi-`
 1046 `tionId` within a `Component` element.

1047 *Section 8 - Listing of Data Items* provides a detailed listing of the data item `type` and
 1048 `subType` elements defined for each category of data item available for a piece of
 1049 equipment: `SAMPLE`, `EVENT`, and `CONDITION`.

1050 7.2.2.4 statistic Attribute for DataItem

1051 A piece of equipment may further process some data types using a statistical calculation
 1052 like average, mean, or square root. In this case, the `statistic` attribute **MAY** be used
 1053 to indicate how the data was processed.

1054 `statistic` may be defined for any `SAMPLE` type `DataItem`. All statistic data is re-
 1055 ported in the standard units of the `DataItem`.

1056 `statistic` data is always the result of a calculation using data that has been measured
 1057 over a specified period of time.

1058 The value of `statistic` may be periodically reset. When a piece of equipment reports
 1059 a `DataItem` with a value that is a `statistic`, the information provided in the XML
 1060 document for that *Data Entity* **MUST** include an additional attribute called `duration`.
 1061 The attribute `duration` defines the period of time over which the `statistic` has been
 1062 calculated. See *MTConnect Standard: Part 3.0 - Streams Information Model* for more
 1063 information about `duration`.

1064 *Table 22* shows the `statistic` calculations that can be defined for a `DataItem`.

Table 22: DataItem attribute statistic type

Statistic	Description
AVERAGE	Mathematical Average value calculated for the data item during the calculation period.
KURTOSIS	DEPRECATED in <i>Version 1.6</i> . 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.

1065 7.2.2.5 units Attribute for DataItem

1066 *Table 23* lists the units that are defined as the standard unit of measure for each type of
 1067 DataItem. All SAMPLE type data items **MUST** report data values in standard units.

Table 23: DataItem attribute units type

Units	Description
AMPERE	Amps
CELSIUS	Degrees Celsius
COUNT	A count of something.
COUNT/SECOND	Count per second.
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
DEGREE_3D	A space-delimited, floating-point representation of the angular rotation in degrees around the X, Y, and Z axes relative to a cartesian coordinate system respectively in order as A, B, and C. If any of the rotations is not known, it MUST be zero (0).
GRAM/CUBIC_METER	Gram per cubic meter.
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

Continuation of Table 23	
Units	Description
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
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	Pascal per second.
PASCAL_SECOND	Measurement of Viscosity
PERCENT	Percentage
PH	A measure of the acidity or alkalinity of a solution.
REVOLUTION/MINUTE	Revolutions per minute
REVOLUTION/SECOND	Revolutions per second.
REVOLUTION/SECOND ²	Revolutions per second squared.
SECOND	A measurement of time.
SIEMENS/METER	A measurement of Electrical Conductivity

Continuation of Table 23	
Units	Description
UNIT_VECTOR_3D	A 3D Unit Vector. Space delimited list of three floating point numbers.
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

1068 7.2.2.6 nativeUnits Attribute for DataItem

1069 The DataItem **MAY** specify the *engineering units* used by the information source using
 1070 the optional attribute nativeUnits. The nativeUnits are inclusive of the *engi-*
 1071 *neering units* for the units attribute (See Table 23). One **MAY** use a prefixed value,
 1072 for example nativeUnits="x:MILE", to extend the *Controlled Vocabulary* with a
 1073 namespace.

1074 *MTConnect* specifies the following *Controlled Vocabulary* for nativeUnits in Ta-
 1075 ble 24:

Table 24: DataItem attribute nativeunits type

Native Units	Description
BAR	Pressure in Bar.
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

Continuation of Table 24	
Native Units	Description
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.
HOURL	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
MILLIMETER_MERCURY	Pressure in Millimeter of Mercury (mmHg).
MINUTE	A measurement of time in minutes
OTHER	Unsupported units
PASCAL/MINUTE	Pascal per minute.
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

Continuation of Table 24	
Native Units	Description
RADIAN/SECOND ²	Rotational acceleration in radian per second squared
REVOLUTION/SECOND	Rotational velocity in revolution per second
TORR	Pressure in Torr.

1076 7.2.2.7 nativeScale Attribute for DataItem

1077 The units of measure for some measured values may be different from the `nativeUnits`
 1078 defined in *Section 7.2.2.8 - category Attribute for DataItem*. In the cases where the units
 1079 of measure use a different weighting or range than is provided by `nativeUnits`, the
 1080 `nativeScale` attribute can be used to define the original units of measure.

1081 As an example, a velocity measured in units of 100 ft/min can be represented as `native-`
 1082 `Units="FEET/MINUTE"` and `nativeScale="100"`.

1083 7.2.2.8 category Attribute for DataItem

1084 Many `DataItem` types provide two forms of data, a value (reported as either a `SAMPLE`
 1085 or `EVENT` category) and a health status (reported as a `CONDITION` category). Therefore,
 1086 each occurrence of a `DataItem` in the XML document **MUST** report a `category` at-
 1087 tribute. This `category` attribute provides the information required by a client software
 1088 application to determine the specific meaning of the data provided.

1089 Each *Data Entity* provided by a piece of equipment **MUST** be identified with one of the
 1090 following: `SAMPLE`, `EVENT`, `CONDITION`.

1091 A `SAMPLE` is the reading of the value of a continuously variable or analog data value. A
 1092 continuous value can be measured at any point-in-time and will always produce a result.
 1093 An example of a continuous data value is the position of a linear axis called X.

1094 The data provided for a `SAMPLE` category data item is always a floating point number
 1095 or integers that have an infinite number of possible values. This is different from a state
 1096 or discrete type data item that has a limited number of possible values. A data item of
 1097 category `SAMPLE` **MUST** also provide the `units` attribute.

1098 An `EVENT` is a data item representing a discrete piece of information from the piece of
 1099 equipment. `EVENT` does not have intermediate values that vary over time, as does `SAM-`
 1100 `PLE`. An `EVENT` is information that, when provided at any specific point in time, repre-

1101 sends the current state of the piece of equipment.

1102 There are two types of `EVENT`: those representing state, with two or more discrete values,
1103 and those representing messages that contain plain text data.

1104 An example of a state type `EVENT` is the value of the data item `DOOR_STATE`, which
1105 can be `OPEN`, `CLOSED`, or `UNLATCHED`. (Note: No other values are valid to represent the
1106 value of `DOOR_STATE`.)

1107 An example of a message type `EVENT` is the value for a data item `PROGRAM`. The value
1108 representing `PROGRAM` can be any valid string of characters.

1109 A `CONDITION` is a data item that communicates information about the health of a piece
1110 of equipment and its ability to function. A valid value for a data item in the category
1111 `CONDITION` can be one of `Normal`, `Warning`, or `Fault`.

1112 A data item of category `CONDITION` **MAY** report multiple values (`CONDITION`) at one
1113 time whereas a data item of category `SAMPLE` or `EVENT` can only have a single value at
1114 any one point in time.

1115 **7.2.2.9 coordinateSystem Attribute for DataItem**

1116 The values reported by a piece of equipment for some types of data will be associated
1117 to a specific positioning measurement system used by the equipment. The `coordi-`
1118 `nateSystem` attribute **MAY** be used to specify the coordinate system used for the mea-
1119 sured value.

1120 The `coordinateSystem` attribute is used by a client software application to interpret
1121 the spatial relationship between values reported by a piece of equipment.

1122 If `coordinateSystem` is not provided, all values representing positional data for `Axes`
1123 **MUST** be interpreted using the `MACHINE` coordinate system and all values representing
1124 positional data for `Path` **MUST** be interpreted using the `WORK` coordinate system.

1125 *Table 25* defines the types of `coordinateSystem` currently supported by the `MTCon-`
1126 `nectDevices` XML document:

Table 25: DataItem attribute `coordinateSystem` type

Coordinate System	Description
<code>MACHINE</code>	An unchangeable coordinate system that has machine zero as its origin.

Continuation of Table 25	
Coordinate System	Description
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.

1127 **7.2.2.10 compositionId Attribute for DataItem**

1128 `compositionId` attribute identifies the id of the `Composition` element where the
1129 reported data is most closely associated.

1130 An example would be a `TEMPERATURE` associated with a `Linear` type axis may be
1131 further clarified by referencing the `MOTOR` or `AMPLIFIER` type `Composition` element
1132 associated with that axis, which differentiates the temperature of the motor from the tem-
1133 perature of the amplifier.

1134 The `compositionId` attribute provides the information required by a client software
1135 application to interpret the data with a greater specificity and to disambiguate between
1136 multiple *Data Entities* of the same data type associated with a `Component` element.

1137 **7.2.2.11 sampleRate Attribute for DataItem**

1138 The value for some data types provided by a piece of equipment may be reported as a
1139 single set of data containing a series of values that have been recorded at a fixed sample
1140 rate. When such data is reported, the `sampleRate` defines the rate at which successive
1141 samples of data were recorded.

1142 The `sampleRate` attribute provides the information required by a client software appli-
1143 cation to interpret the data and the sampling time relationship between successive values
1144 contained in the set of data.

1145 `sampleRate` is expressed in terms of samples per second. If the sample rate is smaller
1146 than one, the number can be represented as a floating point number. For example, a rate 1
1147 per 10 seconds would be 0.1

1148 **7.2.2.12 representation Attribute for DataItem**

1149 Some data types provide data that may consist of a series of values or a file of data, not a
 1150 single value. Other data types provide a series of data values that may require additional
 1151 information so that the data may be correctly understood by a client software application.

1152 When such data is provided, the `representation` attribute **MUST** be used to define
 1153 the format for the data provided.

1154 The types of `representation` defined are provided in *Table 26*.

1155 Note: See *MTConnect Standard: Part 3.0 - Streams Information Model* for more
 1156 information on the structure and format of each `representation`.

Table 26: DataItem attribute representation type

Representation	Description
DATA_SET	<p>The reported value(s) are represented as a set of <i>key-value pairs</i>.</p> <p>Each reported value in the <i>Data Set</i> MUST have a unique key.</p>
DISCRETE DEPRECATED in <i>Version 1.5</i>	<p>DEPRECATED as a representation in MTConnect Version. 1.5. Replaced by the <code>discrete</code> attribute for a <i>Data Entity</i> – <i>Section 7.2.2.14 - discrete Attribute for DataItem</i>.</p> <p>A Data Entity where each discrete occurrence of the data may have the same value as the previous occurrence of the data. There is no reported state change between occurrences of the data. In this case, duplicate occurrences of the same data value SHOULD NOT be suppressed. 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>

Continuation of Table 26	
Representation	Description
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>
TABLE	<p>A <i>Table</i> is a two dimensional set of <i>key-value pairs</i> where the <i>Entry</i> represents a row, and the value is a set of <i>key-value pair</i> <i>Cell</i> elements. The <i>Table</i> follows the same behavior as the <i>Data Set</i> for change tracking, clearing, and history. When an <i>Entry</i> changes, all <i>Cell</i> elements update as a single unit following the behavior of a <i>Data Set</i>.</p> <p>Note: It is best to use the VARIABLE <i>DataItem</i> type if the <i>Cell</i> elements represent multiple semantic types.</p> <p>Each <i>Entry</i> in the <i>Table</i> MUST have a unique key. Each <i>Cell</i> of each <i>Entry</i> in the <i>Table</i> MUST have a unique key.</p> <p>See Section 5.6.5 of <i>MTConnect Standard: Part 3.0 - Streams Information Model</i>, for a description of <i>Entry</i> and <i>Cell</i> elements.</p>

1157 7.2.2.13 significantDigits Attribute for DataItem

1158 significantDigits is used to specify the level of precision (number of significant
1159 digits) for the value provided for a data item.

1160 significantDigits attribute is not required for a data item, but it is recommended
1161 and **SHOULD** be used for any data item reporting a numeric value.

1162 7.2.2.14 discrete Attribute for DataItem

1163 An indication signifying whether each value reported for the *Data Entity* is significant and
 1164 whether duplicate values are to be suppressed.

1165 The value defined **MUST** be either `true` or `false` - an XML boolean type.

1166 `true` indicates that each update to the *Data Entity*'s value is significant and duplicate
 1167 values **MUST NOT** be suppressed.

1168 `false` indicates that duplicated values **MUST** be suppressed.

1169 If a value is not defined for `discrete`, the default value **MUST** be `false`.

1170 7.2.3 Elements for DataItem

1171 *Table 27* lists the elements defined to provide additional information for a `DataItem`
 1172 type XML element.

Table 27: Elements for DataItem

Element	Description	Occurrence
Source	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. 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)	0..1
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

Continuation of Table 27		
Element	Description	Occurrence
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
Definition	The Definition defines the meaning of Entry and Cell elements associated with the DataItem when the representation is either DATA_SET or TABLE.	0..1
Relationships	Relationships <i>organizes</i> one or more DataItemRelationship and SpecificationRelationship.	0..1

1173 7.2.3.1 Source Element for DataItem

1174 Source is an optional XML element that may be used to identify the physical part of a
 1175 piece of equipment where the data represented by DataItem originated and/or it may be
 1176 used to identify a complex name or an alternate name used to identify the data where it
 1177 originated (e.g. a PLC tag name).

1178 As an example, data related to a servo motor on an Axes component may actually origi-
 1179 nate from a measurement made in the Controller element.

1180 In the case where the real name associated with a `DataItem` element is either complex
 1181 or does not meet the format requirements of a NMTOKEN XML type, the real name of
 1182 the element may not be able to be expressed in the `name` attribute. Additionally, a second
 1183 or alternate name may be required to describe a piece of data. An example of this case
 1184 would be the identity of the bit address in a PLC that represents this piece of data (PLC
 1185 address I0015.4). When these cases occur, the alternate name can be provided as the value
 1186 for the `CDATA` for `Source`.

1187 The XML schema in *Figure 13* represents the structure of the `Source` XML element
 1188 showing the attributes defined for `Source`.

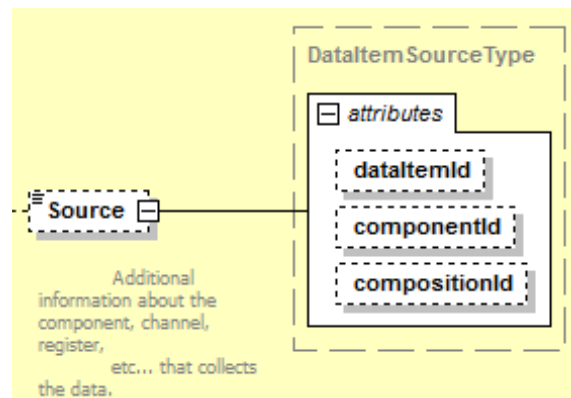


Figure 13: Source Diagram

1189 7.2.3.1.1 Attributes for Source

1190 *Table 28* identifies the attributes available to identify `Source` for a measured value:

Table 28: Attributes for Source

Attribute	Description	Occurrence
<code>componentId</code>	<p>The identifier attribute of the <code>Component</code> element that represents the physical part of a piece of equipment where the data represented by the <code>DataItem</code> element originated.</p> <p>A <i>Valid Data Value</i> reported for <code>componentId</code> MUST be the value of the <code>id</code> attribute for the <code>Component</code> element identified.</p> <p><code>componentId</code> is an optional attribute.</p>	0..1

Continuation of Table 28		
Attribute	Description	Occurrence
dataItemId	<p>The identifier attribute of the <code>DataItem</code> that represents the originally measured value of the data referenced by this data item.</p> <p>A <i>Valid Data Value</i> reported for <code>dataItemId</code> MUST be the value of the <code>id</code> attribute for the <code>DataItem</code> element identified.</p> <p><code>dataItemId</code> is an optional attribute.</p>	0..1
compositionId	<p>The identifier attribute of the <code>Composition</code> element that represents the physical part of a piece of equipment where the data represented by the <code>DataItem</code> element originated.</p> <p>A <i>Valid Data Value</i> reported for <code>compositionId</code> MUST be the value of the <code>id</code> attribute for the <code>Composition</code> element identified.</p> <p><code>compositionId</code> is an optional attribute.</p>	0..1

1191 Note: †One of `componentID`, `composnitionId` , or `dataItemId` **MUST** be provided.

1192 7.2.3.2 Constraints Element for `DataItem`

1193 For some types of `DataItem` elements, the expected value(s) for the data reported for the
 1194 `DataItem` **MAY** be restricted to specific values or a range of values.

1195 `Constraints` is an optional XML element that provides a way to define the expected
 1196 value(s) or the upper and lower limits for the range of values that are expected to be
 1197 reported in response to a *Current Request* or *Sample Request*.

1198 `Constraints` are used by a software application to evaluate the validity of the data
 1199 reported.

1200 The value associated with each `Constraint` element is reported in the CDATA for that
 1201 element.

1202 7.2.3.2.1 Schema for `Constraints`

1203 The XML schema in *Figure 14* represents the structure of the Constraints XML
 1204 element and the elements defined for Constraints.

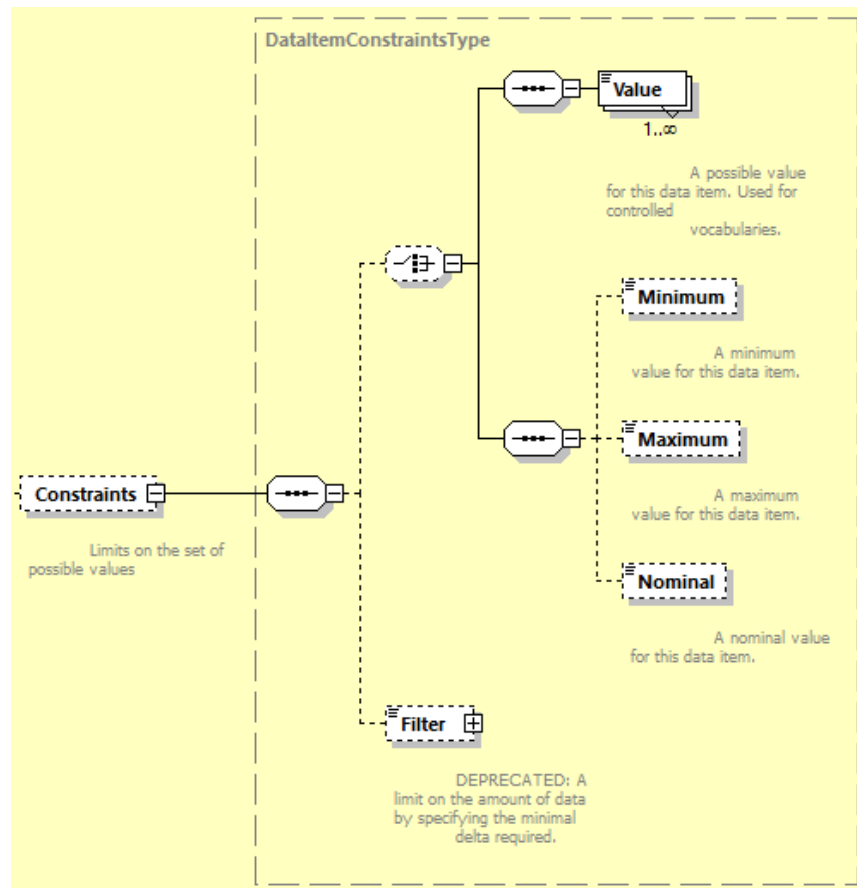


Figure 14: Constraints Diagram

1205 *Table 29* identifies the elements available to identify Constraints for a measured value:

Table 29: Elements for Constraints

Element	Description	Occurrence
Value	<p>Value represents a single data value that is expected to be reported for a <code>DataItem</code> 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 <code>DataItem</code> element, multiple <code>Value</code> elements may be defined.</p> <p>In the case where only one <code>Value</code> 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 <code>Value</code> element.</p> <p>Value MUST NOT be used in conjunction with any other <code>Constraint</code> 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 29		
Element	Description	Occurrence
<code>Filter</code>	<p>DEPRECATED in Version 1.4 – Moved to the <code>Filters</code> element of a <code>DataItem</code>.</p> <p>If the data reported for a <code>DataItem</code> 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 <code>CDATA</code> of this element. <code>Filter</code> is an abstract type XML element. As such, <code>Filter</code> will never appear in the XML document, but will be replaced by a <code>Filter</code> type. The only currently supported <code>Filter</code> type is <code>MINIMUM_DELTA</code>. The <code>CDATA</code> 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 [†]

1206 Note: [†]Remains in schema for backwards compatibility.

1207 7.2.3.3 Filters Element for `DataItem`

1208 `Filters` is an optional XML container that organizes the `Filter` elements for `DataItem`.

1209 `Filters` contains one or more `Filter` XML elements.

Table 30: MTConnect Filters Element

Element	Description	Occurrence
<code>Filters</code>	An XML container consisting of one or more types of <code>Filter</code> XML elements. Only one <code>Filters</code> container MAY appear for a <code>DataItem</code> element.	0..1

1210 7.2.3.3.1 Filter

1211 *Filter* provides a means to control when an *Agent* records updated information for a
 1212 data item. Currently, there are two types of *Filter* elements defined in the MTConnect
 1213 Standard - `MINIMUM_DELTA` and `PERIOD`. More *Filter* types may be added in the
 1214 future.

1215 The value associated with each *Filter* element is reported in the CDATA for that ele-
 1216 ment.

1217 *Figure 15* represents the structure for *Filter* XML element.

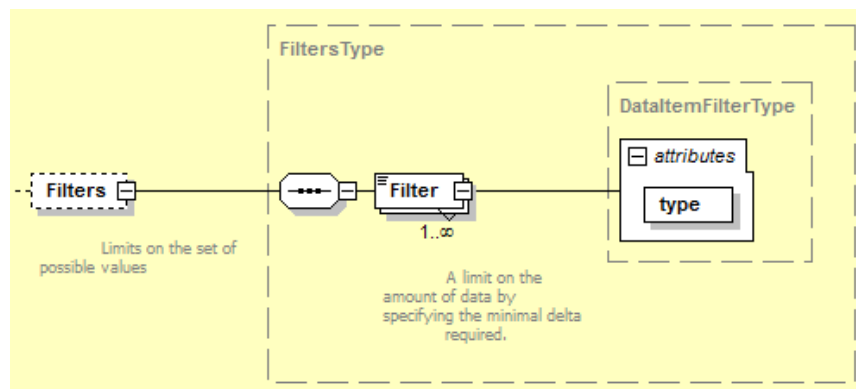


Figure 15: Filter Diagram

1218 *Table 31* describes the types of *Filter* defined for a *DataItem* element and the ex-
 1219 pected behavior of an *Agent* when a *Filter* is applied to *DataItem* element.

Table 31: DataItem Element Filter type

type	Description	Occurrence
MINIMUM_DELTA	<p>For a <code>MINIMUM_DELTA</code> type <i>Filter</i>, 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 31		
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 [†]

1220 [†]Note: Either MINIMUM_DELTA or PERIOD can be defined, not both.

1221 7.2.3.4 InitialValue Element for DataItem

1222 InitialValue is an XML element that defines the value to be set for the data item after
1223 a reset event.

1224 The value associated with the InitialValue element is reported in the CDATA for this
1225 element and **MUST** be an absolute value using the same units as the reported data.

1226 7.2.3.5 ResetTrigger Element for DataItem

1227 The value of some data types is periodically reset to the value of the InitialValue ele-
1228 ment. These reset events may be based upon a specific elapsed time or may be triggered by
1229 a physical or logical reset action that causes the reset to occur. ResetTrigger provides
1230 additional information regarding the meaning of the data – establishing an understanding
1231 of the time frame that the data represents so that the data may be correctly understood by
1232 a client software application.

Table 32: 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

1233 The reset action that **MAY** cause a reset to occur is provided in the CDATA for this ele-
 1234 ment.

1235 The reset actions that may cause a reset to occur are described in *Table 33*.

Table 33: 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 33	
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.

1236 7.2.3.6 Definition Element for DataItem

1237 *Figure 16* represents the *XML Schema* structure for `Definition` element.

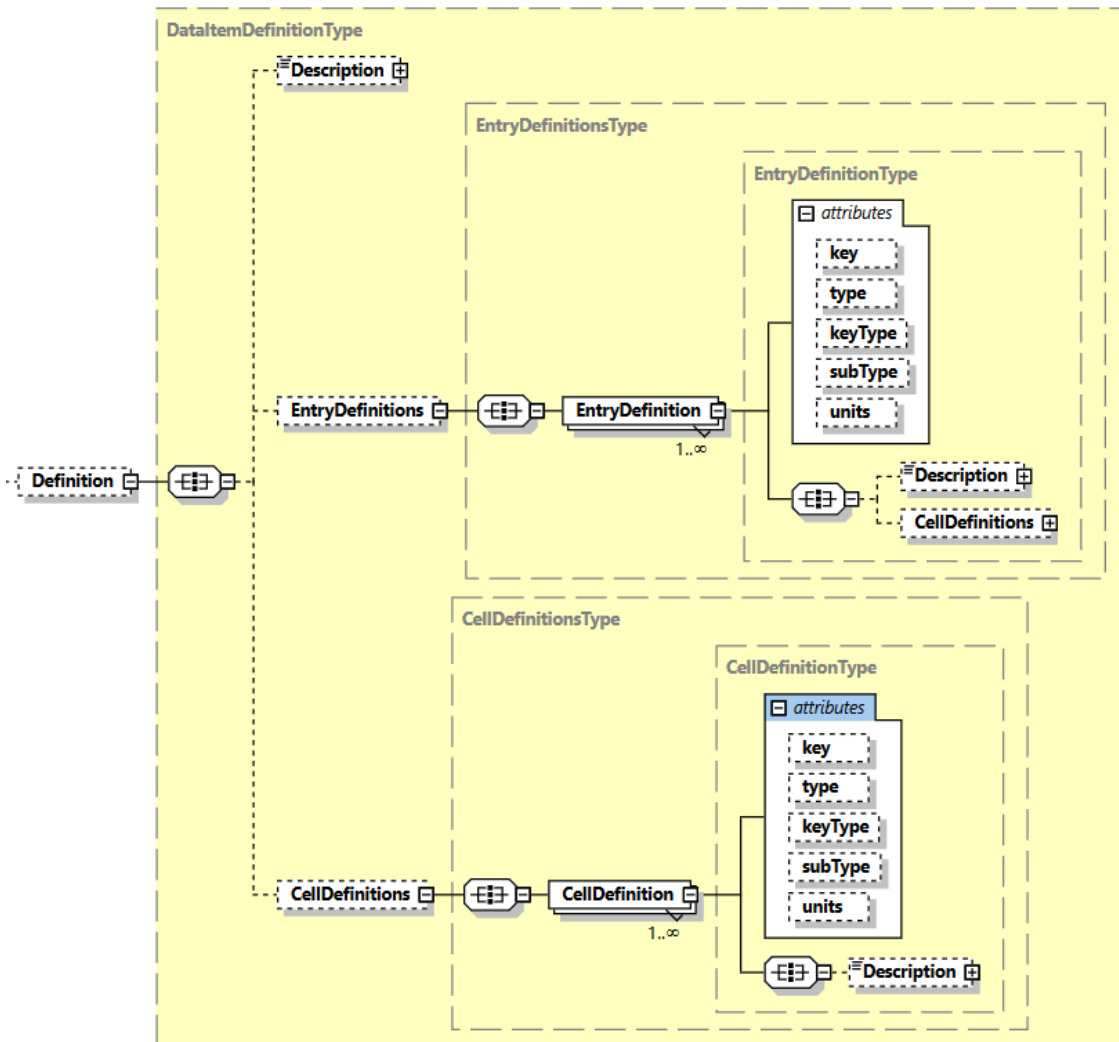


Figure 16: Definition Diagram

1238 The Definition provides additional descriptive information for any DataItem rep-
 1239 resentations. When the representation is either DATA_SET or TABLE, it gives the
 1240 specific meaning of a key and **MAY** provide a Description, type, and units for
 1241 semantic interpretation of data.

Table 34: Elements for Definition

Element	Description	Occurrence
Description	The Description of the Definition. See Component Description	0..1

Continuation of Table 34		
Element	Description	Occurrence
EntryDefinitions	The EntryDefinitions aggregates EntryDefinition .	0..1
CellDefinitions	The CellDefinitions aggregates CellDefinition.	0..1

1242 **7.2.3.6.1 EntryDefinitions Element for Definition**

1243 The EntryDefinitions aggregates EntryDefinition for Definition.

1244 Elements for EntryDefinitions

Table 35: Elements for EntryDefinitions

Element	Description	Occurrence
EntryDefinition	The semantic definition of an Entry	1..*

1245 **7.2.3.6.2 EntryDefinition Element for Definition**

1246 When the representation is DATA_SET, the EntryDefinition provides the
1247 Description, units, and type of each Entry identified by a unique key.

1248 When the representation is TABLE, the EntryDefinition provides a Descrip-
1249 tion and a set of CellDefinitions for an Entry identified by a unique key.

1250 The key for the EntryDefinion **MUST** be unique for a given DataItem Defini-
1251 tion.

1252 Attributes for EntryDefinition**Table 36:** Attributes for EntryDefinition

Attribute	Description	Occurrence
key	The unique identification of the Entry in the Definition. The description applies to all Entry observations having this key.	0..1
keyType	The DataItem type that defines the meaning of the key.	0..1
units	Same as DataItem units. See <i>Section 7.2.2.5 - units Attribute for DataItem</i> . Only valid for representation of DATA_SET.	0..1
type	Same as DataItem type. See <i>Section 8 - Listing of Data Items</i> .	0..1
subType	Same as DataItem subType. See <i>Section 8 - Listing of Data Items</i> .	0..1

1253 Elements for EntryDefinition**Table 37:** Elements for EntryDefinition

Element	Description	Occurrence
Description	The Description of the EntryDefinition. See Component Description	0..1
CellDefinitions	The CellDefinitions aggregates CellDefinition if the representation is TABLE.	0..1

1254 **7.2.3.6.3 CellDefinitions Element for Definition**

1255 The CellDefinitions aggregates CellDefinition declarations.

1256 Elements for CellDefinitions**Table 38:** Elements for CellDefinitions

Element	Description	Occurrence
CellDefinition	The semantic definition of a Cell.	1..*

1257 **7.2.3.6.4 CellDefinition Element for CellDefinitions**

1258 When the representation is TABLE, the CellDefinition provides the De-
 1259 scription and the units associated each Cell by key.

1260 The key for the CellDefinion **MUST** be unique for a given Definition or En-
 1261 tryDefinition.

1262 Attributes for CellDefinition**Table 39:** Attributes for CellDefinition

Attribute	Description	Occurrence
key	The unique identification of the Entry in the Definition. The description applies to all Entry <i>observations</i> having this key.	0..1
keyType	The DataItem type that defines the meaning of the key.	0..1
units	Same as DataItem units. See <i>Section 7.2.2.5 - units Attribute for DataItem</i> .	0..1
type	Same as DataItem type. See <i>Section 8 - Listing of Data Items</i> .	0..1
subType	Same as DataItem subType. See <i>Section 8 - Listing of Data Items</i> .	0..1

1263 Elements for CellDefinition

Table 40: Elements for CellDefinition

Element	Description	Occurrence
Description	The Description of the CellDefinition. See Component Description	0..1

1264 7.2.3.7 Relationships Element for DataItem

1265 Relationships *organizes* DataItemRelationship and SpecificationRe-
1266 lationship.

1267 See Section 9.2 - Relationships for definitions of Relationships and Relation-
1268 ship.

1269 7.2.3.7.1 DataItemRelationship

1270 A Relationship providing a semantic reference to another DataItem described by
1271 the type property.

Table 41: Attributes for DataItemRelationship

Attribute	Description	Occurrence
name	A descriptive name associated with this Relationship. An NMTOKEN XML type.	0..1
type	Specifies how the DataItem is related. The value provided for type MUST be one of the following values: ATTACHMENT: A reference to a DataItem that associates the values with an external entity. COORDINATE_SYSTEM: The referenced DataItem provides the id of the effective Coordinate System. LIMIT: The referenced DataItem provides process limits. OBSERVATION: The referenced DataItem provides the observed values.	1

Continuation of Table 41		
Attribute	Description	Occurrence
idRef	A reference to the related <code>DataItem</code> id. An NMTOKEN XML type.	1

1272 7.2.3.7.2 SpecificationRelationship

1273 A Relationship providing a semantic reference to a `Specification` described by
 1274 the `type` property.

Table 42: Attributes for `SpecificationRelationship`

Attribute	Description	Occurrence
name	A descriptive name associated with this <code>Relationship</code> . An NMTOKEN XML type.	0..1
type	Specifies how the <code>Specification</code> is related. The value provided for <code>type</code> MUST be one of the following values: LIMIT: The referenced <code>Specification</code> provides process limits.	1
idRef	A reference to the related <code>Specification</code> id. An NMTOKEN XML type.	1

1275 8 Listing of Data Items

1276 In the MTConnect Standard, `DataItem` elements are defined and organized based upon
1277 the `category` and `type` attributes. The `category` attribute provides a high level
1278 grouping for `DataItem` elements based on the kind of information that is reported by
1279 the data item.

1280 These categories are:

1281 • `SAMPLE`

1282 A `SAMPLE` reports a continuously variable or analog data value.

1283 • `EVENT`

1284 An `EVENT` reports information representing a functional state, with two or more
1285 discrete values, associated with a component or it contains a message. The data
1286 provided may be a numeric value or text.

1287 • `CONDITION`

1288 A `CONDITION` reports information about the health of a piece of equipment and its
1289 ability to function.

1290 The `type` attribute specifies the specific kind of data that is reported. For some types of
1291 data items, a `subType` attribute may also be used to differentiate between multiple data
1292 items of the same `type` where the information reported by the data item has a different,
1293 but related, meaning.

1294 Many types of data items provide two forms of data: a value (reported as either a `SAMPLE`
1295 or `EVENT`) and a health status (reported as a `CONDITION`). These `DataItem` types **MAY**
1296 be defined in more than one `category` based on the data that they report.

1297 8.1 Data Items in category SAMPLE

1298 The types of `DataItem` elements in the `SAMPLE` category report data representing a
 1299 continuously changing or analog data value. This data can be measured at any point-in-
 1300 time and will always produce a result. The data provided may be a scalar floating point
 1301 number or integers that have an infinite number of possible values. The `units` attribute
 1302 **MUST** be defined and reported for each `DataItem` in this category.

1303 *Table 43* defines the types and subtypes of `DataItem` elements defined for the `SAMPLE`
 1304 category. The subtypes are indented below their associated types.

Table 43: `DataItem` type subType for category `SAMPLE`

DataItem type/subType	Description	Units
ACCELERATION	The positive rate of change of velocity. If a subType is not specified, the reported value for the data MUST default to the subType of <code>ACTUAL</code> .	MILLIMETER/SECOND ²
ACTUAL	The measured or reported value of an <i>observation</i> .	MILLIMETER/SECOND ²
COMMANDED	Directive value including adjustments such as an offset or overrides.	MILLIMETER/SECOND ²
PROGRAMMED	Directive value without offsets and adjustments.	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 <code>PROCESS_TIMER</code> and <code>EQUIPMENT_TIMER</code> .	SECOND

Continuation of Table 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
AMPERAGE	DEPRECATED in <i>Version 1.6</i> . Replaced by AMPERAGE_AC and AMPERAGE_DC.	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
TARGET	The desired or preset amperage to be delivered from a power source.	AMPERE
AMPERAGE_AC	The measurement of an electrical current that reverses direction at regular short intervals. A subType MUST always be specified. If not specified further in statistic, defaults to RMS amperage.	AMPERE
ACTUAL	The measured or reported value of an <i>observation</i> .	AMPERE
COMMANDED	Directive value including adjustments such as an offset or overrides.	AMPERE
PROGRAMMED	Directive value without offsets and adjustments.	AMPERE

Continuation of Table 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
AMPERAGE_DC	The measurement of an electric current flowing in one direction only. A subType MUST always be specified.	AMPERE
ACTUAL	The measured or reported value of an <i>observation</i> .	AMPERE
COMMANDED	Directive value including adjustments such as an offset or overrides.	AMPERE
PROGRAMMED	Directive value without offsets and adjustments.	AMPERE
ANGLE	The measurement of angular position.	DEGREE
ACTUAL	The measured or reported value of an <i>observation</i> .	DEGREE
COMMANDED	Directive value including adjustments such as an offset or overrides.	DEGREE
ANGULAR_– ACCELERATION	The positive rate of change of angular velocity. If a subType is not specified, the reported value for the data MUST default to the subType of ACTUAL.	DEGREE / SECOND ²
ACTUAL	The measured or reported value of an <i>observation</i> .	DEGREE / SECOND ²
COMMANDED	Directive value including adjustments such as an offset or overrides.	DEGREE / SECOND ²
PROGRAMMED	Directive value without offsets and adjustments.	DEGREE / SECOND ²

Continuation of Table 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
ANGULAR_ DECELERATION	Negative rate of change of angular velocity.	DEGREE/SECOND ²
ACTUAL	The measured or reported value of an <i>observation</i> .	DEGREE/SECOND ²
COMMANDED	Directive value including adjustments such as an offset or overrides.	DEGREE/SECOND ²
PROGRAMMED	Directive value without offsets and adjustments.	DEGREE/SECOND ²
ANGULAR_VELOCITY	Rate of change of angular position.	DEGREE/SECOND
ASSET_UPDATE_RATE	The average rate of change of values for assets in the MTConnect streams. The average is computed over a rolling window defined by the implementation.	COUNT/SECOND
AXIS_FEEDRATE	The feedrate of a linear axis.	MILLIMETER/SECOND
ACTUAL	The measured or reported value of an <i>observation</i> .	MILLIMETER/SECOND
COMMANDED	Directive value including adjustments such as an offset or overrides.	MILLIMETER/SECOND
JOG	The feedrate specified by a logic or motion program, by a pre-set value, or set by a switch as the feedrate for a linear axis when operating in a manual state or method (jogging).	MILLIMETER/SECOND

Continuation of Table 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
OVERRIDE	The operator's overridden value. Percent of commanded. DEPRECATED in Version 1.3. See EVENT category data items.	PERCENT
PROGRAMMED	Directive value without offsets and adjustments.	MILLIMETER/SECOND
RAPID	Performing an operation faster or in less time than nominal rate.	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
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 or reported value of an <i>observation</i> .	MILLIMETER/SECOND
COMMANDED	Directive value including adjustments such as an offset or overrides.	MILLIMETER/SECOND
PROGRAMMED	Directive value without offsets and adjustments.	MILLIMETER/SECOND

Continuation of Table 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
DECELERATION	Negative rate of change of velocity.	MILLIMETER/SECOND ²
ACTUAL	The measured or reported value of an <i>observation</i> .	MILLIMETER/SECOND ²
COMMANDED	Directive value including adjustments such as an offset or overrides.	MILLIMETER/SECOND ²
PROGRAMMED	Directive value without offsets and adjustments.	MILLIMETER/SECOND ²
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
ACTUAL	The measured or reported value of an <i>observation</i> .	CUBIC_ MILLIMETER/SECOND ²
COMMANDED	Directive value including adjustments such as an offset or overrides.	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 or reported value of an <i>observation</i> .	MILLIGRAM/CUBIC_ MILLIMETER
COMMANDED	Directive value including adjustments such as an offset or overrides.	MILLIGRAM/CUBIC_ MILLIMETER
DEPOSITION_MASS	The mass of the material deposited in an additive manufacturing process.	MILLIGRAM

Continuation of Table 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
ACTUAL	The measured or reported value of an <i>observation</i> .	MILLIGRAM
COMMANDED	Directive value including adjustments such as an offset or overrides.	MILLIGRAM
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 or reported value of an <i>observation</i> .	CUBIC_– MILLIMETER/SECOND
COMMANDED	Directive value including adjustments such as an offset or overrides.	CUBIC_– MILLIMETER/SECOND
DEPOSITION_VOLUME	The spatial volume of material to be deposited in an additive manufacturing process.	CUBIC_MILLIMETER
ACTUAL	The measured or reported value of an <i>observation</i> .	CUBIC_MILLIMETER
COMMANDED	Directive value including adjustments such as an offset or overrides.	CUBIC_MILLIMETER
DIAMETER	The measured dimension of a diameter.	MILLIMETER
DISPLACEMENT	The change in position of an object.	MILLIMETER
ELECTRICAL_ENERGY	The value of Wattage used or generated by a component over an interval of time.	WATT_SECOND

Continuation of Table 43: 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	The elapsed time of a temporary halt of action.	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
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

Continuation of Table 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
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
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

Continuation of Table 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
GLOBAL_POSITION	DEPRECATED in Version 1.1	None
HUMIDITY_ABSOLUTE	The amount of water vapor expressed in grams per cubic meter.	GRAM/CUBIC_METER
ACTUAL	The measured or reported value of an <i>observation</i> .	GRAM/CUBIC_METER
COMMANDED	Directive value including adjustments such as an offset or overrides.	GRAM/CUBIC_METER
HUMIDITY_RELATIVE	The amount of water vapor present expressed as a percent to reach saturation at the same temperature.	PERCENT
ACTUAL	The measured or reported value of an <i>observation</i> .	PERCENT
COMMANDED	Directive value including adjustments such as an offset or overrides.	PERCENT
HUMIDITY_SPECIFIC	The ratio of the water vapor present over the total weight of the water vapor and air present expressed as a percent.	PERCENT
ACTUAL	The measured or reported value of an <i>observation</i> .	PERCENT
COMMANDED	Directive value including adjustments such as an offset or overrides.	PERCENT
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

Continuation of Table 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
USEABLE	The remaining useable length of an object.	MILLIMETER
LEVEL	DEPRECATED in Version 1.2. See <code>FILL_LEVEL</code>	None
LINEAR_FORCE	A <i>Force</i> applied to a mass in one direction only.	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
OBSERVATION_UPDATE_RATE	The average rate of change of values for data items in the MTConnect streams. The average is computed over a rolling window defined by the implementation.	COUNT/SECOND
ORIENTATION	A measured or calculated orientation of a plane or vector relative to a cartesian coordinate system. ORIENTATION SHOULD have a <code>coordinateSystemIdRef</code> or a <code>coordinateSystem</code> attribute, otherwise the <code>coordinateSystem</code> attribute MUST default to <code>WORK</code> coordinates.	DEGREE_3D
ACTUAL	The measured or reported value of an <i>observation</i> .	DEGREE_3D

Continuation of Table 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
COMMANDED	Directive value including adjustments such as an offset or overrides.	DEGREE_3D
PATH_FEEDRATE	The feedrate for the axes, or a single axis, associated with a Path component– a vector.	MILLIMETER/SECOND
ACTUAL	The measured or reported value of an <i>observation</i> .	MILLIMETER/SECOND
COMMANDED	Directive value including adjustments such as an offset or overrides.	MILLIMETER/SECOND
JOG	The feedrate specified by a logic or motion program, by a pre-set value, or set by a switch as the feedrate for the axes, or a single axis, associated with a Path 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	Directive value without offsets and adjustments.	MILLIMETER/SECOND
RAPID	Performing an operation faster or in less time than nominal rate.	MILLIMETER/SECOND
PATH_FEEDRATE_– PER_REVOLUTION	The feedrate for the axes, or a single axis.	MILLIMETER/REVO– LUTION
ACTUAL	The measured or reported value of an <i>observation</i> .	MILLIMETER/REVO– LUTION

Continuation of Table 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
COMMANDED	Directive value including adjustments such as an offset or overrides.	MILLIMETER/REVO- LUTION
PROGRAMMED	Directive value without offsets and adjustments.	MILLIMETER/REVO- LUTION
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

Continuation of Table 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
ACTUAL	The measured or reported value of an <i>observation</i> .	MILLIMETER_3D
PROGRAMMED	Directive value without offsets and adjustments.	MILLIMETER_3D
COMMANDED	Directive value including adjustments such as an offset or overrides.	MILLIMETER_3D
PROBE	The position provided by a measurement probe. DEPRECATION WARNING: May be deprecated in the future.	MILLIMETER_3D
TARGET	The goal of the operation or process.	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 measured or reported value of an <i>observation</i> .	MILLIMETER
COMMANDED	Directive value including adjustments such as an offset or overrides.	MILLIMETER

Continuation of Table 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
PROGRAMMED	Directive value without offsets and adjustments.	MILLIMETER
TARGET	The goal of the operation or process.	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 measured relative to atmospheric pressure. Commonly referred to as gauge pressure.	PASCAL
PRESSURE_ABSOLUTE	The force per unit area measured relative to a vacuum.	PASCAL

Continuation of Table 43: 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	The elapsed time of a temporary halt of action.	SECOND
PROCESS	<p>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.</p>	SECOND

Continuation of Table 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
PRESSURIZATION_ RATE	The change of pressure per unit time.	PASCAL/SECOND
ACTUAL	The measured or reported value of an <i>observation</i> .	PASCAL/SECOND
COMMANDED	Directive value including adjustments such as an offset or overrides.	PASCAL/SECOND
PROGRAMMED	Directive value without offsets and adjustments.	PASCAL/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 or reported value of an <i>observation</i> .	REVOLUTION/MINUTE
COMMANDED	Directive value including adjustments such as an offset or overrides.	REVOLUTION/MINUTE
OVERRIDE	The operator's overridden value. Percent of commanded. DEPRECATED in Version 1.3. See EVENT category data items.	PERCENT
PROGRAMMED	Directive value without offsets and adjustments.	REVOLUTION/MINUTE
SOUND_LEVEL	The measurement of a sound level or sound pressure level relative to atmospheric pressure.	DECIBEL

Continuation of Table 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
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
TEMPERATURE	The measurement of temperature.	CELSIUS
ACTUAL	The measured or reported value of an <i>observation</i> .	CELSIUS
COMMANDED	Directive value including adjustments such as an offset or overrides.	CELSIUS

Continuation of Table 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
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	DEPRECATED in <i>Version 1.6</i> . Replaced by VOLTAGE_AC and VOLTAGE_DC.	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 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
VOLTAGE_AC	<p>The measurement of the electrical potential between two points in an electrical circuit in which the current periodically reverses direction.</p> <p>A subType MUST be specified.</p> <p>If not specified further in statistic, defaults to RMS voltage.</p>	VOLT
ACTUAL	The measured or reported value of an <i>observation</i> .	VOLT
COMMANDED	Directive value including adjustments such as an offset or overrides.	VOLT
PROGRAMMED	Directive value without offsets and adjustments.	VOLT
VOLTAGE_DC	<p>The measurement of the electrical potential between two points in an electrical circuit in which the current is unidirectional.</p> <p>A subType MUST be specified.</p>	VOLT
ACTUAL	The measured or reported value of an <i>observation</i> .	VOLT
COMMANDED	Directive value including adjustments such as an offset or overrides.	VOLT
PROGRAMMED	Directive value without offsets and adjustments.	VOLT

Continuation of Table 43: 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 measured or reported value of an <i>observation</i> .	MILLILITER
START	Boundary when an activity or an event commences.	MILLILITER
ENDED	Boundary when an activity or an event terminates.	MILLILITER
CONSUMED	Reported or measured value of the amount used in the manufacturing process.	MILLILITER
WASTE	Reported or measured value of the amount discarded.	MILLILITER
PART	Reported or measured value of amount included in the <i>Part</i> .	MILLILITER
VOLUME_SPATIAL	The geometric volume of an object or container.	CUBIC_MILLIMETER
ACTUAL	The measured or reported value of an <i>observation</i> .	CUBIC_MILLIMETER

Continuation of Table 43: DataItem type subType for category SAMPLE		
DataItem type/subType	Description	Units
START	Boundary when an activity or an event commences.	CUBIC_MILLIMETER
ENDED	Boundary when an activity or an event terminates.	CUBIC_MILLIMETER
CONSUMED	Reported or measured value of the amount used in the manufacturing process.	CUBIC_MILLIMETER
WASTE	Reported or measured value of the amount discarded.	CUBIC_MILLIMETER
PART	Reported or measured value of amount included in the <i>Part</i> .	CUBIC_MILLIMETER
WATTAGE	The measurement of power flowing through or dissipated by an electrical circuit or piece of equipment.	WATT
ACTUAL	The measured or reported value of an <i>observation</i> .	WATT
TARGET	The goal of the operation or process.	WATT
X_DIMENSION	Measured dimension of an entity relative to the X direction of the referenced coordinate system.	MILLIMETER
Y_DIMENSION	Measured dimension of an entity relative to the Y direction of the referenced coordinate system.	MILLIMETER
Z_DIMENSION	Measured dimension of an entity relative to the Z direction of the referenced coordinate system.	MILLIMETER

1305 8.2 Data Items in category EVENT

1306 DataItem types in the EVENT category represent a discrete piece of information from a
 1307 piece of equipment. EVENT does not have intermediate values that vary over time.

1308 An EVENT is information that, when provided at any specific point in time, represents the
 1309 current state of the piece of equipment.

1310 There are two types of EVENT: those representing state, with two or more discrete values,
 1311 and those representing messages that contain plain text data.

1312 Table 44 defines the DataItem types and subtypes defined for the EVENT category. The
 1313 subtypes are indented below their associated types.

Table 44: DataItem type subType for category EVENT

DataItem type subType	Description
ACTIVATION_COUNT	<p>Accumulation of the number of times a function has attempted to, or is planned to attempt to, activate or be performed.</p> <p>Use the <code>discrete</code> attribute with value <code>true</code> to report non-aggregate count.</p> <p>See <i>Section 7.2.3.5 - ResetTrigger Element for DataItem</i> to reset the count.</p> <p>The <i>Valid Data Value</i> MUST be numeric.</p>
ALL	An accumulation representing all actions, items, or activities being counted independent of the outcome. ALL is the default subType.
BAD	An accumulation representing actions, items, or activities being counted that do not conform to specification or expectation.
GOOD	An accumulation representing actions, items, or activities being counted that conform to specification or expectation.
TARGET	The goal of the operation or process.
REMAINING	An accumulation representing actions, items, or activities yet to be counted.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
COMPLETE	An accumulation representing actions, items, or activities that have been completed, independent of the outcome.
FAILED	An accumulation representing actions or activities that were attempted, but failed to complete or resulted in an unexpected or unacceptable outcome.
ABORTED	An accumulation representing actions or activities that were attempted, but terminated before they could be completed.
ACTIVE_AXES	<p>The set of axes currently associated with a Path or Controller <i>Structural Element</i>.</p> <p>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>
ADAPTER_SOFTWARE_VERSION	<p>The originator's software version of the Adapter.</p> <p>The <i>Valid Data Value</i> MUST be a string.</p>
ADAPTER_URI	<p>The URI of the Adapter.</p> <p>The <i>Valid Data Value</i> MUST be a string.</p>

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
ALARM	DEPRECATED in Version 1.1. Replaced with CONDITION category.
ALARM_LIMIT	<p>A set of limits used to trigger warning or alarm indicators.</p> <p>The <i>Valid Data Value</i> MUST be a float.</p> <p>The representation attribute MUST be DATA_SET.</p> <p>The EntryDefinition key MUST be from the following:</p> <p> UPPER_LIMIT: The upper conformance boundary for a variable.</p> <p>Note: immediate concern or action may be required.</p> <p> UPPER_WARNING: The upper boundary indicating increased concern and supervision may be required.</p> <p> LOWER_WARNING: The lower boundary indicating increased concern and supervision may be required.</p> <p> LOWER_LIMIT: The lower conformance boundary for a variable.</p> <p>Note: immediate concern or action may be required.</p>
APPLICATION	<p>The application on a component.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p> <p>A subType MUST always be specified.</p>
LICENSE	The license code to validate or activate the hardware or software.
VERSION	The version of the hardware or software.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
RELEASE_DATE	<p>The date the hardware or software was released for general use.</p> <p>It MUST be reported in ISO 8601 format.</p>
INSTALL_DATE	<p>The date the hardware or software was installed.</p> <p>It MUST be reported in ISO 8601 format.</p>
MANUFACTURER	<p>The corporate identity for the maker of the hardware or software.</p>
AVAILABILITY	<p>Represents the <i>Agent's</i> ability to communicate with the data source.</p> <p>This MUST be provided for a <i>Device Element</i> and MAY be provided for any other <i>Structural Element</i>. The <i>Valid Data Value</i> MUST be AVAILABLE or UNAVAILABLE.</p>
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>

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
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 <code>AXIS_FEEDRATE_OVERRIDE</code> is expressed as a percentage of the designated feedrate for the axis.</p> <p>When <code>AXIS_FEEDRATE_OVERRIDE</code> is applied, the resulting commanded feedrate for the axis is limited to the value of the original feedrate multiplied by the value of the <code>AXIS_FEEDRATE_OVERRIDE</code>.</p> <p>There MAY be different subtypes of <code>AXIS_FEEDRATE_OVERRIDE</code>; 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 <code>PROGRAMMED</code>, <code>JOG</code>, and <code>RAPID</code>.</p>
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 <code>JOG</code> subtype of <code>AXIS_FEEDRATE_OVERRIDE</code> is applied, the resulting commanded feedrate for the axis is limited to the value of the original <code>JOG</code> subtype of the <code>AXIS_FEEDRATE</code> multiplied by the value of the <code>JOG</code> subtype of <code>AXIS_FEEDRATE_OVERRIDE</code>.</p>
PROGRAMMED	Directive value without offsets and adjustments.
RAPID	Performing an operation faster or in less time than nominal rate.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
AXIS_INTERLOCK	<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>The <i>Valid Data Value</i> MUST be ACTIVE or INACTIVE.</p>
AXIS_STATE	<p>An indicator of the controlled state of a Linear or Rotary component representing an axis.</p> <p>The <i>Valid Data Value</i> MUST be HOME, TRAVEL, PARKED, or STOPPED.</p>
BLOCK	<p>The line of code or command being executed by a Controller <i>Structural Element</i>.</p> <p>The value reported for Block MUST include the entire expression for a line of program code, including all parameters.</p>
BLOCK_COUNT	<p>The total count of the number of blocks of program code that have been executed since execution started.</p> <p>BLOCK_COUNT counts blocks of program code executed regardless of program structure (e.g., looping or branching within the program).</p> <p>The starting value for BLOCK_COUNT MAY be established by an initial value provided in the Constraint element defined for the data item.</p>
CHUCK_INTERLOCK	<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>The <i>Valid Data Value</i> MUST be ACTIVE or INACTIVE.</p>

Continuation of Table 44: 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>
CLOCK_TIME	<p>The value provided by a timing device at a specific point in time.</p> <p>CLOCK_TIME MUST be reported in ISO 8601 format.</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>

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
ACTION	<p>An indication of the operating state of a mechanism represented by a <code>Composition</code> type component.</p> <p>The operating state indicates whether the <code>Composition</code> element is activated or disabled.</p> <p>The <i>Valid Data Value</i> MUST be ACTIVE or INACTIVE.</p>
LATERAL	<p>An indication of the position of a mechanism that may move in a lateral direction. The mechanism is represented by a <code>Composition</code> type component.</p> <p>The position information indicates whether the <code>Composition</code> 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 <code>Composition</code> type component.</p> <p>The operating state indicates whether the state of the <code>Composition</code> 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 <code>Composition</code> type component.</p> <p>The activation state indicates whether the <code>Composition</code> element is activated or not.</p> <p>The <i>Valid Data Value</i> MUST be ON or OFF.</p>

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
VERTICAL	<p>An indication of the position of a mechanism that may move in a vertical direction. The mechanism is represented by a <code>Composition</code> type component.</p> <p>The position information indicates whether the <code>Composition</code> 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>
CONNECTION_STATUS	<p>The status of the connection between an <i>Adapter</i> and an <i>Agent</i>.</p> <p>The <i>Valid Data Value</i> MUST be CLOSED, LISTEN, or ESTABLISHED.</p>

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
CONTROL_LIMIT	<p>A set of limits used to indicate whether a process variable is stable and in control.</p> <p>The <i>Valid Data Value</i> MUST be a float.</p> <p>The <code>representation</code> attribute MUST be <code>DATA_SET</code>.</p> <p>The <code>EntryDefinition</code> key MUST be from the following:</p> <p> <code>UPPER_LIMIT</code>: The upper conformance boundary for a variable.</p> <p>Note: immediate concern or action may be required.</p> <p> <code>UPPER_WARNING</code>: The upper boundary indicating increased concern and supervision may be required.</p> <p> <code>NOMINAL</code>: The ideal or desired value for a variable.</p> <p> <code>LOWER_WARNING</code>: The lower boundary indicating increased concern and supervision may be required.</p> <p> <code>LOWER_LIMIT</code>: The lower conformance boundary for a variable.</p> <p>Note: immediate concern or action may be required.</p>
CONTROLLER_MODE	<p>The current mode of the <code>Controller</code> component. The <i>Valid Data Value</i> MUST be <code>AUTOMATIC</code>, <code>MANUAL</code>, <code>MANUAL_DATA_INPUT</code>, <code>SEMI_AUTOMATIC</code>, or <code>EDIT</code>.</p>
CONTROLLER_MODE_OVERRIDE	<p>A setting or operator selection that changes the behavior of a piece of equipment.</p> <p>A subType MUST always be specified.</p>

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
DRY_RUN	<p>A setting or operator selection used to execute a test mode to confirm the execution of machine functions.</p> <p>The <i>Valid Data Value</i> MUST be ON or OFF.</p> <p>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.</p>
MACHINE_AXIS_LOCK	<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>When MACHINE_AXIS_LOCK is ON, program execution continues normally, but no equipment motion occurs</p>
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>

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
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>
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>

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
CYCLE_COUNT	<p>Accumulation of the number of times a cyclic function has attempted to, or is planned to attempt to execute.</p> <p>Use the <code>discrete</code> attribute with value <code>true</code> to report non-aggregate count.</p> <p>See <i>Section 7.2.3.5 - ResetTrigger Element for DataItem</i> to reset the count.</p> <p>The <i>Valid Data Value</i> MUST be numeric.</p>
ALL	An accumulation representing all actions, items, or activities being counted independent of the outcome. ALL is the default subType.
BAD	An accumulation representing actions, items, or activities being counted that do not conform to specification or expectation.
GOOD	An accumulation representing actions, items, or activities being counted that conform to specification or expectation.
TARGET	The goal of the operation or process.
REMAINING	An accumulation representing actions, items, or activities yet to be counted.
COMPLETE	An accumulation representing actions, items, or activities that have been completed, independent of the outcome.
FAILED	An accumulation representing actions or activities that were attempted, but failed to complete or resulted in an unexpected or unacceptable outcome.
ABORTED	An accumulation representing actions or activities that were attempted, but terminated before they could be completed.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
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.
DEACTIVATION_COUNT	<p>Accumulation of the number of times a function has attempted to, or is planned to attempt to, deactivate or cease.</p> <p>Use the <code>discrete</code> attribute with value <code>true</code> to report non-aggregate count.</p> <p>See <i>Section 7.2.3.5 - ResetTrigger Element for DataItem</i> to reset the count.</p> <p>The <i>Valid Data Value</i> MUST be numeric.</p>
ALL	An accumulation representing all actions, items, or activities being counted independent of the outcome. ALL is the default subType.
BAD	An accumulation representing actions, items, or activities being counted that do not conform to specification or expectation.
GOOD	An accumulation representing actions, items, or activities being counted that conform to specification or expectation.
TARGET	The goal of the operation or process.
REMAINING	An accumulation representing actions, items, or activities yet to be counted.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
COMPLETE	An accumulation representing actions, items, or activities that have been completed, independent of the outcome.
FAILED	An accumulation representing actions or activities that were attempted, but failed to complete or resulted in an unexpected or unacceptable outcome.
ABORTED	An accumulation representing actions or activities that were attempted, but terminated before they could be completed.
DEVICE_ADDED	<p>DEVICE_ADDED is an Event that provides the UUID of a new device added to an <i>MTConnect Agent</i>.</p> <p><i>Valid Data Value</i> is the value of the Device's UUID that was added to the <i>MTConnect Agent</i>.</p>
DEVICE_CHANGED	<p>DEVICE_CHANGED is an Event that provides the UUID of the device whose <i>Metadata</i> has changed.</p> <p><i>Valid Data Value</i> is the value of the Device's UUID for which the metadata has changed.</p>
DEVICE_REMOVED	<p>DEVICE_REMOVED is an Event that provides the UUID of a device removed from an <i>MTConnect Agent</i>.</p> <p><i>Valid Data Value</i> is the value of the Device's UUID that was removed from the <i>MTConnect Agent</i>.</p>
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>

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
DIRECTION	<p>The direction of motion.</p> <p>A subType MUST always be specified</p>
LINEAR	<p>The direction of linear motion.</p> <p>The <i>Valid Data Value</i> MUST be POSTIVE, NEGATIVE, or NONE.</p>
ROTARY	<p>The direction of rotary motion using the right-hand rule convention.</p> <p>The <i>Valid Data Value</i> MUST be CLOCKWISE, COUNTER_CLOCKWISE, or NONE.</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>
AUXILIARY	<p>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.</p>

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
PRIMARY	<p>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.</p> <p>If no subType is specified, PRIMARY MUST be the default END_OF_BAR indication.</p>
EQUIPMENT_MODE	<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>EQUIPMENT_MODE MAY have more than one subtype defined.</p> <p>A subType MUST always be specified.</p>
DELAY	The elapsed time of a temporary halt of action.
LOADED	<p>An indication that the sub-parts of a piece of equipment are under load.</p> <p>Example: For traditional machine tools, this is an indication that the cutting tool is assumed to be engaged with the part.</p> <p>The <i>Valid Data Value</i> MUST be ON or OFF.</p>
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>

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
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 component.</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>
FIRMWARE	<p>The embedded software of a component.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p> <p>A subType MUST always be specified.</p>
LICENSE	<p>The license code to validate or activate the hardware or software.</p>
VERSION	<p>The version of the hardware or software.</p>

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
RELEASE_DATE	<p>The date the hardware or software was released for general use.</p> <p>It MUST be reported in ISO 8601 format.</p>
INSTALL_DATE	<p>The date the hardware or software was installed.</p> <p>It MUST be reported in ISO 8601 format.</p>
MANUFACTURER	The corporate identity for the maker of the hardware or software.
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.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
HARDWARE	<p>The hardware of a component.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p> <p>A subType MUST always be specified.</p>
LICENSE	The license code to validate or activate the hardware or software.
VERSION	The version of the hardware or software.
RELEASE_DATE	<p>The date the hardware or software was released for general use.</p> <p>It MUST be reported in ISO 8601 format.</p>
INSTALL_DATE	<p>The date the hardware or software was installed.</p> <p>It MUST be reported in ISO 8601 format.</p>
MANUFACTURER	The corporate identity for the maker of the hardware or software.
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>
LIBRARY	<p>The software library on a component.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p> <p>A subType MUST always be specified.</p>
LICENSE	The license code to validate or activate the hardware or software.
VERSION	The version of the hardware or software.
RELEASE_DATE	<p>The date the hardware or software was released for general use.</p> <p>It MUST be reported in ISO 8601 format.</p>
INSTALL_DATE	<p>The date the hardware or software was installed.</p> <p>It MUST be reported in ISO 8601 format.</p>

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
MANUFACTURER	The corporate identity for the maker of the hardware or software.
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.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
LOAD_COUNT	<p>Accumulation of the number of times an operation has attempted to, or is planned to attempt to, load materials, parts, or other items.</p> <p>Use the <code>discrete</code> attribute with value <code>true</code> to report non-aggregate count.</p> <p>See <i>Section 7.2.3.5 - ResetTrigger Element for DataItem</i> to reset the count.</p> <p>The <i>Valid Data Value</i> MUST be numeric.</p>
ALL	An accumulation representing all actions, items, or activities being counted independent of the outcome. <code>ALL</code> is the default subType.
BAD	An accumulation representing actions, items, or activities being counted that do not conform to specification or expectation.
GOOD	An accumulation representing actions, items, or activities being counted that conform to specification or expectation.
TARGET	The goal of the operation or process.
REMAINING	An accumulation representing actions, items, or activities yet to be counted.
COMPLETE	An accumulation representing actions, items, or activities that have been completed, independent of the outcome.
FAILED	An accumulation representing actions or activities that were attempted, but failed to complete or resulted in an unexpected or unacceptable outcome.
ABORTED	An accumulation representing actions or activities that were attempted, but terminated before they could be completed.
LOCK_STATE	The state or operating mode of a <code>Lock</code> .

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
MATERIAL	<p>The identifier of a material used or consumed in the manufacturing process.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p>
MATERIAL_LAYER	<p>Identifies the layers of material applied to a part or product as part of an additive manufacturing process.</p> <p>The <i>Valid Data Value</i> MUST be an integer.</p>
ACTUAL	The measured or reported value of an <i>observation</i> .
TARGET	The goal of the operation or process.
MESSAGE	Any text string of information to be transferred from a piece of equipment to a client software application.
MTCONNECT_VERSION	<p>The reference version of the MTConnect Standard supported by the <i>Adapter</i>.</p> <p>The <i>Valid Data Value</i> MUST be a string.</p>
NETWORK	<p>Network details of a component.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p> <p>A subType MUST always be specified.</p> <p>If the subType is WIRELESS, the <i>Valid Data Value</i> MUST be YES or NO.</p>
IPV4_ADDRESS	The IPV4 network address of the component.
IPV6_ADDRESS	The IPV6 network address of the component.
GATEWAY	The Gateway for the component network.
SUBNET_MASK	The SubNet mask for the component network.
VLAN_ID	The layer2 Virtual Local Network (VLAN) ID for the component network.
MAC_ADDRESS	Media Access Control Address. The unique physical address of the network hardware.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
WIRELESS	Identifies whether the connection type is wireless.
OPERATING_SYSTEM	The Operating System of a component. The <i>Valid Data Value</i> MUST be a text string. A subType MUST always be specified.
LICENSE	The license code to validate or activate the hardware or software.
VERSION	The version of the hardware or software.
RELEASE_DATE	The date the hardware or software was released for general use. It MUST be reported in ISO 8601 format.
INSTALL_DATE	The date the hardware or software was installed. It MUST be reported in ISO 8601 format.
MANUFACTURER	The corporate identity for the maker of the hardware or software.
OPERATOR_ID	The identifier of the person currently responsible for operating the piece of equipment. DEPRECATION WARNING : May be deprecated in the future. See USER below.
PALLET_ID	The identifier for a pallet. The <i>Valid Data Value</i> MUST be a text string.
PART_COUNT	The aggregate count of parts. Use the <code>discrete</code> attribute with value <code>true</code> to report non-aggregate part count. See Section 7.2.3.5 - <i>ResetTrigger Element for DataItem</i> to reset the count. The <i>Valid Data Value</i> MUST be numeric.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
ALL	An accumulation representing all actions, items, or activities being counted independent of the outcome. ALL is the default subType.
BAD	An accumulation representing actions, items, or activities being counted that do not conform to specification or expectation.
GOOD	An accumulation representing actions, items, or activities being counted that conform to specification or expectation.
TARGET	The goal of the operation or process.
REMAINING	An accumulation representing actions, items, or activities yet to be counted.
COMPLETE	An accumulation representing actions, items, or activities that have been completed, independent of the outcome.
FAILED	An accumulation representing actions or activities that were attempted, but failed to complete or resulted in an unexpected or unacceptable outcome.
ABORTED	An accumulation representing actions or activities that were attempted, but terminated before they could be completed.
PART_DETECT	An indication designating whether a part or work piece has been detected or is present. The <i>Valid Data Value</i> MUST be PRESENT or NOT_PRESENT.
PART_GROUP_ID	Identifier given to a collection of individual parts. If no subType is specified, UUID is default. The <i>Valid Data Value</i> MUST be a string.
UUID	The globally unique identifier as specified in ISO 11578 or RFC 4122.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
LOT	An identifier that references a group of parts tracked as a lot.
BATCH	An identifier that references a group of parts produced in a batch.
RAW_MATERIAL	Material that is used to produce parts.
HEAT_TREAT	An identifier used to reference a material heat number.
PART_ID	An identifier of a part in a manufacturing operation. The <i>Valid Data Value</i> MUST be a text string.
PART_KIND_ID	Identifier given to link the individual occurrence to a class of parts, typically distinguished by a particular part design. If no subType is specified, UUID is default. The <i>Valid Data Value</i> MUST be a string.
UUID	The globally unique identifier as specified in ISO 11578 or RFC 4122.
PART_NUMBER	Identifier of a particular part design or model.
PART_FAMILY	An identifier given to a group of parts having similarities in geometry, manufacturing process, and/or functions.
PART_NAME	A word or set of words by which a part is known, addressed, or referred to.
PART_NUMBER	DEPRECATED in <i>Version 1.7</i> . PART_NUMBER is now a subType of PART_KIND_ID.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
PART_PROCESSING_STATE	<p>The particular condition of the part occurrence at a specific time.</p> <p>The <i>Valid Data Value</i> MUST be NEEDS_PROCESSING, IN_PROCESS, PROCESSING_ENDED, PROCESSING_ENDED_COMPLETE, PROCESSING_ENDED_STOPPED, PROCESSING_ENDED_ABORTED, PROCESSING_ENDED_LOST, PROCESSING_ENDED_SKIPPED, PROCESSING_ENDED_REJECTED, WAITING_FOR_TRANSIT, IN_TRANSIT, or TRANSIT_COMPLETE.</p>
PART_STATUS	<p>State or condition of a part.</p> <p>If unique identifier is given, part status is for that individual. If group identifier is given without a unique identifier, then the status is assumed to be for the whole group.</p> <p>The <i>Valid Data Value</i> MUST be PASS or FAIL.</p>
PART_UNIQUE_ID	<p>Identifier given to a distinguishable, individual part. If no subType is specified, UUID is default.</p> <p>The <i>Valid Data Value</i> MUST be a string.</p>
UUID	The globally unique identifier as specified in ISO 11578 or RFC 4122.
SERIAL_NUMBER	A serial number that uniquely identifies a specific part.
RAW_MATERIAL	Material that is used to produce parts.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
PATH_FEEDRATE_OVERRIDE	<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 PATH_FEEDRATE_OVERRIDE is expressed as a percentage of the designated feedrate for the path.</p> <p>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.</p> <p>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.</p>
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	Directive value without offsets and adjustments.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
RAPID	Performing an operation faster or in less time than nominal rate.
PATH_MODE	<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>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> .
LINE	The state of the power source for the <i>Structural Element</i> .
POWER_STATUS	DEPRECATED in Version 1.1.0.
PROCESS_AGGREGATE_ID	<p>Identifier given to link the individual occurrence to a group of related occurrences, such as a process step in a process plan.</p> <p>The <i>Valid Data Value</i> MUST be a string.</p>
PROCESS_STEP	Identifier of the step in the process plan that this occurrence corresponds to. Synonyms include "operation id".

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
PROCESS_PLAN	Identifier of the process plan that this occurrence belongs to. Synonyms include "routing id", "job id".
ORDER_NUMBER	Identifier of the authorization of the process occurrence. Synonyms include "job id", "work order".
PROCESS_KIND_ID	Identifier given to link the individual occurrence to a class of processes or process definition. The <i>Valid Data Value</i> MUST be a string.
UUID	The globally unique identifier as specified in ISO 11578 or RFC 4122.
PROCESS_NAME	A word or set of words by which a process being executed (process occurrence) by the device is known, addressed, or referred to.
ISO_STEP_EXECUTABLE	A reference to a ISO 10303 Executable.
PROCESS_OCCURRENCE_ID	An identifier of a process being executed by the device. The <i>Valid Data Value</i> MUST be a string.
PROCESS_STATE	The particular condition of the process occurrence at a specific time. The <i>Valid Data Value</i> MUST be INITIALIZING, READY, ACTIVE, COMPLETE, INTERRUPTED, or ABORTED.
PROCESS_TIME	The time and date associated with an activity or event. PROCESS_TIME MUST be reported in ISO 8601 format.
START	Boundary when an activity or an event commences.
COMPLETE	The time and date associated with the completion of an activity or event.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
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.
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.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
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>If not specified, the default subType is MAIN.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p>
SCHEDULE	<p>The identity of a control program that is used to specify the order of execution of other programs.</p>
MAIN	<p>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.</p>
ACTIVE	<p>The identity of the logic or motion program currently executing.</p>

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
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.
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	<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>The <i>Valid Data Value</i> MUST be LOCAL or EXTERNAL.</p>
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.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
PROGRAM_NEST_LEVEL	<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 PROGRAM_NEST_LEVEL MUST be an integer.</p>
ROTARY_MODE	<p>The current operating mode for a Rotary type axis.</p> <p>The <i>Valid Data Value</i> MUST 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>
ROTATION	<p>A three space angular rotation relative to a coordinate system.</p> <p>When the DataItem has a coordinateSystemIdRef attribute and the CoordinateSystem does not specify a Rotation, the value of the <i>observation</i> is the rotation of the the referenced CoordinateSystem.</p> <p>The units MUST be DEGREE_3D</p>

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
SENSOR_ATTACHMENT	<p>A <code>SensorAttachment</code> is an Event defining an <i>Attachment</i> between a sensor and an entity.</p> <p>The <i>Valid Data Value</i> MUST be a string.</p> <p>The <code>EntryDefinition</code> key MUST be from the following:</p> <p style="padding-left: 40px;"><code>SENSOR_ID</code>: The identity of a sensor used to observe some measurement of an item.</p>
SERIAL_NUMBER	<p>The serial number associated with a Component, Asset, or Device. The <i>Valid Data Value</i> MUST be a text string.</p>
SPECIFICATION_LIMIT	<p>A set of limits defining a range of values designating acceptable performance for a variable.</p> <p>The <i>Valid Data Value</i> MUST be a float.</p> <p>The <code>representation</code> attribute MUST be <code>DATA_SET</code>.</p> <p>The <code>EntryDefinition</code> key MUST be from the following:</p> <p style="padding-left: 40px;"><code>UPPER_LIMIT</code>: The upper conformance boundary for a variable.</p> <p>Note: immediate concern or action may be required.</p> <p style="padding-left: 40px;"><code>NOMINAL</code>: The ideal or desired value for a variable.</p> <p style="padding-left: 40px;"><code>LOWER_LIMIT</code>: The lower conformance boundary for a variable.</p> <p>Note: immediate concern or action may be required.</p>

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
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</i> MUST be:</p> <p>ACTIVE if power has been removed and the spindle cannot be operated.</p> <p>INACTIVE if power to the spindle has not been deactivated.</p>
TOOL_ASSET_ID	The identifier of an individual tool asset. The <i>Valid Data Value</i> MUST be a text string.
TOOL_GROUP	An identifier for the tool group associated with a specific tool. Commonly used to designate spare tools.
TOOL_ID	DEPRECATED in Version 1.2.0. See TOOL_ASSET_ID. The identifier of the tool currently in use for a given Path.
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.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
TRANSFER_COUNT	<p>Accumulation of the number of times an operation has attempted to, or is planned to attempt to, transfer materials, parts, or other items from one location to another.</p> <p>Use the <code>discrete</code> attribute with value <code>true</code> to report non-aggregate count.</p> <p>See <i>Section 7.2.3.5 - ResetTrigger Element for DataItem</i> to reset the count.</p> <p>The <i>Valid Data Value</i> MUST be numeric.</p>
ALL	An accumulation representing all actions, items, or activities being counted independent of the outcome. ALL is the default subType.
BAD	An accumulation representing actions, items, or activities being counted that do not conform to specification or expectation.
GOOD	An accumulation representing actions, items, or activities being counted that conform to specification or expectation.
TARGET	The goal of the operation or process.
REMAINING	An accumulation representing actions, items, or activities yet to be counted.
COMPLETE	An accumulation representing actions, items, or activities that have been completed, independent of the outcome.
FAILED	An accumulation representing actions or activities that were attempted, but failed to complete or resulted in an unexpected or unacceptable outcome.
ABORTED	An accumulation representing actions or activities that were attempted, but terminated before they could be completed.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
TRANSLATION	<p>A three space linear translation relative to a coordinate system.</p> <p>When the DataItem has a coordinateSystemIdRef attribute and the CoordinateSystem does not specify a Translation, the value of the <i>observation</i> is the translation of the referenced CoordinateSystem.</p> <p>The units MUST be MILLIMETER_3D</p>
UNLOAD_COUNT	<p>Accumulation of the number of times an operation has attempted to, or is planned to attempt to, unload materials, parts, or other items.</p> <p>Use the discrete attribute with value true to report non-aggregate count.</p> <p>See Section 7.2.3.5 - <i>ResetTrigger Element for DataItem</i> to reset the count.</p> <p>The <i>Valid Data Value</i> MUST be numeric.</p>
ALL	An accumulation representing all actions, items, or activities being counted independent of the outcome. ALL is the default subType.
BAD	An accumulation representing actions, items, or activities being counted that do not conform to specification or expectation.
GOOD	An accumulation representing actions, items, or activities being counted that conform to specification or expectation.
TARGET	The goal of the operation or process.
REMAINING	An accumulation representing actions, items, or activities yet to be counted.
COMPLETE	An accumulation representing actions, items, or activities that have been completed, independent of the outcome.

Continuation of Table 44: DataItem type subType for category EVENT	
DataItem type subType	Description
FAILED	An accumulation representing actions or activities that were attempted, but failed to complete or resulted in an unexpected or unacceptable outcome.
ABORTED	An accumulation representing actions or activities that were attempted, but terminated before they could be completed.
USER	The identifier of the person currently responsible for operating the piece of equipment. A subType MUST always be specified.
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.
VALVE_STATE	The state of a valve is one of open, closed, or transitioning between the states. The <i>Valid Data Value</i> MUST be OPEN, OPENING, CLOSED, or CLOSING.
ACTUAL	The measured or reported value of an <i>observation</i> .
PROGRAMMED	An instructed target value without offsets and adjustments.
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 44: DataItem type subType for category EVENT	
DataItem type subType	Description
WAIT_STATE	<p>An indication of the reason that EXECUTION is reporting a value of WAIT.</p> <p>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.</p>
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 a text string.</p>
WORKHOLDING_ID	<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	<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> <p>The reported value returned for WORK_OFFSET identifies the location in a table or list where the actual work offset values are stored.</p>

1314 8.3 Data Items in category CONDITION

1315 CONDITION category data items report data representing a *Structural Element*'s status
 1316 regarding its ability to operate or it provides an indication whether the data reported for
 1317 the *Structural Element* is within an expected range.

1318 CONDITION is reported differently than SAMPLE or EVENT. CONDITION **MUST** be
 1319 reported as Normal, Warning, or Fault.

1320 All DataItem types in the SAMPLE category **MAY** have associated CONDITION states.
 1321 CONDITION states indicate whether the value for the data is within an expected range and
 1322 **MUST** be reported as Normal, or the value is unexpected or out of tolerance for the data
 1323 and a Warning or Fault **MUST** be provided.

1324 Some DataItem types in the EVENT category **MAY** have associated CONDITION states.

1325 Additional CONDITION types are provided to represent the health and fault status of
 1326 *Structural Elements*. Table 45 defines these additional DataItem types.

1327 CONDITION type data items are unlike other data item types since they **MAY** have mul-
 1328 tiple concurrently active values at any point in time.

Table 45: 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 45	
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.

1329 9 Configuration

1330 Configuration contains technical information about a component describing its phys-
 1331 ical layout, functional characteristics, and relationships with other components within a
 1332 piece of equipment.

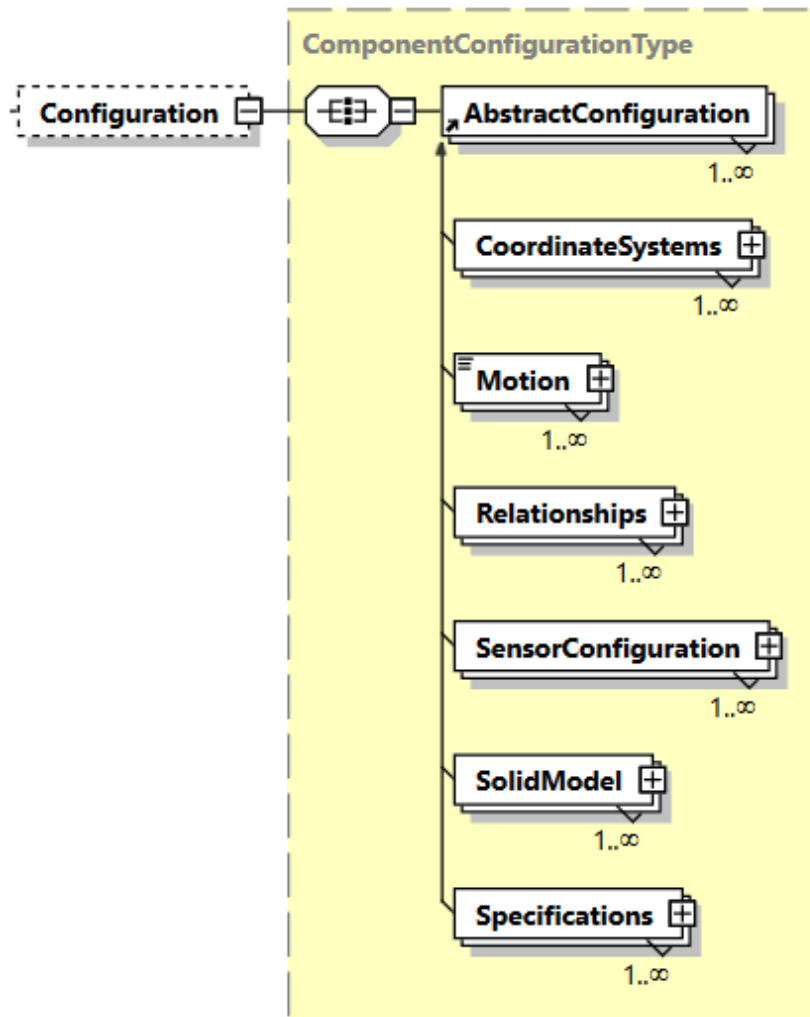


Figure 17: Configuration Element

1333 Table 46 lists the types of Configuration defined for a Component.

Table 46: Types of Configuration

type	Description
CoordinateSystems	CoordinateSystems <i>organizes</i> CoordinateSystem elements for a Component and its children.
Motion	Motion defines the movement of the Component relative to a coordinate system.
Relationships	Relationships <i>organizes</i> Relationship elements for a Component.
SensorConfiguration	SensorConfiguration contains configuration information about a Sensor.
SolidModel	SolidModel references a file with the three-dimensional geometry of the Component or Composition.
Specifications	Specifications <i>organizes</i> Specification elements for a Component.

1334 9.1 Sensor

1335 *Sensor* is a unique type of a piece of equipment. A *Sensor* is typically comprised of
 1336 two major components: a *sensor unit* that provides signal processing, conversion, and
 1337 communications and the *sensing elements* that provides a signal or measured value.

1338 The *sensor unit* is modeled as a *Lower Level* Component called *Sensor*. The *sensing*
 1339 *element* may be modeled as a *Composition* element of a *Sensor* element and the mea-
 1340 sured value would be modeled as a *DataItem* (See *Section 8 - Listing of Data Items* for
 1341 more information on *DataItem* elements). Each *sensor unit* may have multiple *sensing*
 1342 *elements*; each representing the data for a variety of measured values.

1343 Example: A pressure transducer could be modeled as a *Sensor* (Component) with a
 1344 name = *Pressure Transducer B* and its measured value could be modeled as a *PRESSURE*
 1345 type *DataItem*.

1346 While a *Sensor* may be modeled in the XML document in different ways, it will always be
 1347 modeled to associate the information measured by each *sensor element* with the *Structural*
 1348 *Element* to which the measured value is most closely associated.

1349 9.1.1 Sensor Data

1350 The most basic implementation of a sensor occurs when the *sensing element* itself is not
 1351 identified in the data model, but the data that is measured by the *sensing element* is pro-
 1352 vided as a data item associated with a *Component*. An example would be the measured
 1353 value of the temperature of a spindle motor. This would be represented as a *DataItem*
 1354 called *TEMPERATURE* that is associated with the *Rotary* type axis element called "C"
 1355 as shown in *Example 7*:

Example 7: Example of Sensing Element provided as data item associated with a *Com-
 ponent*

```

1356 1 <Components>
1357 2   <Axes
1358 3     <Components>
1359 4       <Rotary id="c" name="C">
1360 5         <DataItems>
1361 6           <DataItem type="TEMPERATURE"
1362 7             id="ctemp" category="SAMPLE"
1363 8             name="Stemp" units="DEGREE"/>
1364 9         </DataItems>
1365 10      </Rotary>
1366 11    </Components>
1367 12  </Axes>
1368 13 </Components>

```

1369 A sensor may measure values associated with any *Component* or *Device* element.
 1370 Some examples of how sensor data may be modeled are represented in *Figure 18* :

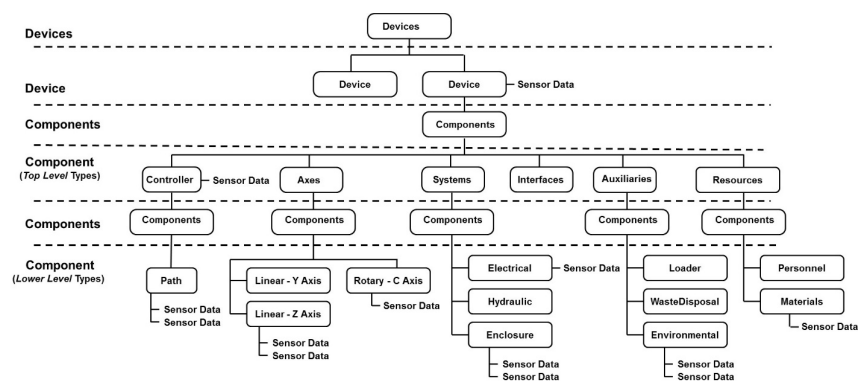


Figure 18: Sensor Data Associations

1371 9.1.2 Sensor Unit

1372 A *sensor unit* is an intelligent piece of equipment that manages the functions of one or
1373 more *sensing elements*.

1374 Typical functions of the *sensor unit* include:

- 1375 • convert low level signals from the *sensing elements* into data that can be used by
1376 other pieces of equipment. (Example: Convert a non-linear millivolt signal from a
1377 temperature sensor into a scaled temperature value that can be transmitted to another
1378 piece of equipment.)
- 1379 • process *sensing element* data into calculated values. (Example: temperature sensor
1380 data is converted into calculated values of average temperature, maximum tempera-
1381 ture, minimum temperature, etc.)
- 1382 • provide calibration and configuration information associated with each *sensing ele-*
1383 *ment*
- 1384 • monitor the health and integrity of the *sensing elements* and the *sensor unit*. (Exam-
1385 ple: The *sensor unit* may provide diagnostics on each *sensing element* (e.g., open
1386 wire detection) and itself (e.g., measure internal temperature of the *sensor unit*).

1387 Depending on how the *sensor unit* is used, it may be considered as either an independent
1388 piece of equipment and modeled in the XML document as a `Device`, or it may be mod-
1389 eled as a *Top Level Component* called `Sensor` if it is integral to a piece of equipment.

1390 A `Sensor` **MAY** have its own `uuid` so it can be tracked throughout its lifetime.

1391 The following examples demonstrate how a *Sensor* may be modeled in the XML document
1392 differently based on how the *Sensor* functions within the overall piece of equipment

1393 Example#1: If the `Sensor` provides vibration measurement data for the spindle on a
1394 piece of equipment, it could be modeled as a `Sensor` for rotary axis named C.

Example 8: Example of Sensor for rotary axis

```

1395 1 <Components>
1396 2   <Axes
1397 3     <Components>
1398 4       <Rotary id="c" name="C">
1399 5         <Components>
1400 6           <Sensor id="spdlm" name="Spindlemonitor">
1401 7             <DataItems>
1402 8               <DataItem type="DISPLACEMENT" id="cvib">
```

```

1403 9          category="SAMPLE" name="Svib"
1404 10          units="MILLIMETER"/>
1405 11          </DataItems>
1406 12          </Sensor >
1407 13          <Components>
1408 14          </Rotary>
1409 15          </Components>
1410 16          </Axes>
1411 17 </Components>

```

1412 **Example#2:** If a Sensor provides measurement data for multiple Component elements
 1413 within a piece of equipment and is not associated with any particular Component ele-
 1414 ment, it **MAY** be modeled in the XML document as an independent *Lower Level* Com-
 1415 ponent and the data associated with measurements are associated with their associated
 1416 Component elements.

1417 This example represents a *sensor unit* with two *sensing elements*, one measures spindle
 1418 vibration and the other measures the temperature for the X axis. The *sensor unit* also has
 1419 a *sensing element* measuring the internal temperature of the *sensor unit*.

Example 9: Example of Sensor Unit with Sensing Element

```

1420 1 <Device id="d1" uuid="HM1" name="HMC_3Axis">
1421 2   <Description>3 Axis Mill</Description>
1422 3   <Components>
1423 4     <Axes
1424 5       <Components>
1425 6         <Sensor id="sens1" name="Sensorunit">
1426 7           <DataItems>
1427 8             <DataItem type="TEMPERATURE" id="sentemp"
1428 9               category="SAMPLE" name="Sensortemp"
1429 10              units="DEGREE"/>
1430 11           </DataItems>
1431 12         </Sensor >
1432 13         <Rotary id="c" name="C">
1433 14           <DataItems>
1434 15             <DataItem type="DISPLACEMENT" id="cvib"
1435 16               %category="SAMPLE" name="Svib"
1436 17               units="MILLIMETER">
1437 18               <Source componentId="sens1"/>
1438 19             </DataItem/>
1439 20           </DataItems>
1440 21         </Rotary>
1441 22         <Linear id="x" name="X">
1442 23           <DataItems>
1443 24             <DataItem type="TEMPERATURE" id="xt"
1444 25               category="SAMPLE" name="Xtemp"
1445 26               units="DEGREE">
1446 27               <Source componentId="sens1"/>
1447 28             </DataItem/>

```

```

1448 29          </DataItems>
1449 30          </Linear>
1450 31      <Components>
1451 32          </Axes>
1452 33      </Components>
1453 34 </Device>

```

1454 9.1.3 Sensor Configuration

1455 When a `Sensor` unit is modeled in the XML document as a `Component` or as a separate
 1456 piece of equipment, it may provide additional configuration information for the *sensor*
 1457 *elements* and the *sensor unit* itself.

1458 Configuration data provides information required for maintenance and support of the
 1459 sensor.

1460 Configuration data is only available when the `Sensor` unit is modeled as a `Com-`
 1461 `ponent` or a separate piece of equipment. For details on the modeling of configuration
 1462 data in the XML document, see *Section 4.4.3.2 - Configuration for Component*.

1463 When `Sensor` represents the *sensor unit* for multiple *sensing element(s)*, each sensing
 1464 element is represented by a `Channel`. The *sensor unit* itself and each `Channel` repre-
 1465 senting one *sensing element* **MAY** have its own configuration data.

1466 `SensorConfiguration` can contain any descriptive content for a *sensor unit*. This
 1467 element is defined to contain mixed content and additional XML elements (indicated by
 1468 the any element in *Figure 19*) **MAY** be added to extend the schema for `SensorCon-`
 1469 `figuration`.

1470 *Figure 19* represents the structure of the `SensorConfiguration` XML element show-
 1471 ing the attributes defined for `SensorConfiguration`.

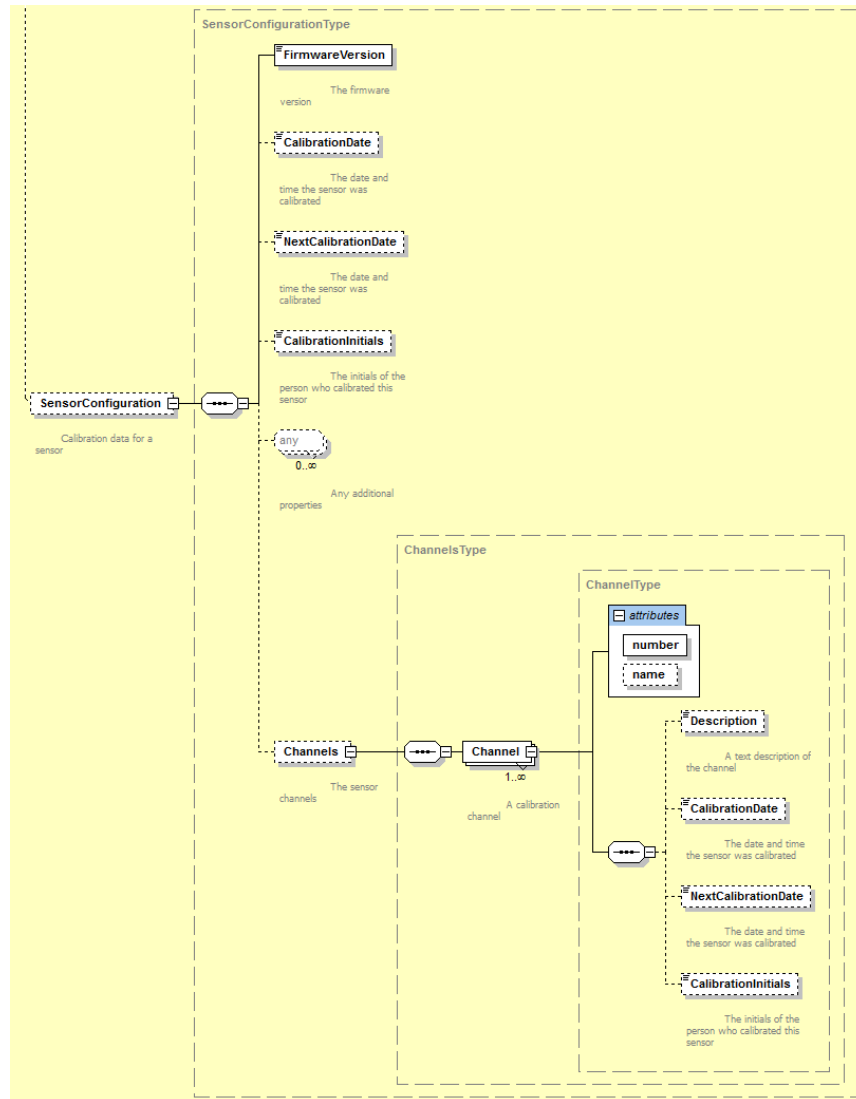


Figure 19: SensorConfiguration Diagram

Table 47: MTConnect SensorConfiguration Element

Element	Description	Occurrence
SensorConfiguration	<p>An element that can contain descriptive content defining the configuration information for <code>Sensor</code>.</p> <p>For <code>Sensor</code>, the valid configuration is <code>SensorConfiguration</code> 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

1472 **9.1.3.1 Elements for SensorConfiguration**

1473 *Table 48* defines the configuration elements available for `SensorConfiguration`:

Table 48: Elements for SensorConfiguration

Element	Description	Occurrence
FirmwareVersion	<p>Version number for the sensor unit as specified by the manufacturer.</p> <p><code>FirmwareVersion</code> is a required element if <code>SensorConfiguration</code> is used.</p> <p>The data value for <code>FirmwareVersion</code> is provided in the <code>CDATA</code> for this element and MAY be any numeric or text content.</p>	1

Continuation of Table 48		
Element	Description	Occurrence
CalibrationDate	<p>Date upon which the <i>sensor unit</i> was last calibrated.</p> <p>The data value for CalibrationDate is provided in the CDATA for this element and MUST be represented in the W3C ISO 8601 format.</p>	0..1
NextCalibrationDate	<p>Date upon which the <i>sensor unit</i> is next scheduled to be calibrated.</p> <p>The data value for NextCalibrationDate is provided in the CDATA for this element and MUST be represented in the W3C ISO 8601 format.</p>	0..1
CalibrationInitials	<p>The initials of the person verifying the validity of the calibration data.</p> <p>The data value for CalibrationInitials is provided in the CDATA for this element and MAY be any numeric or text content.</p>	0..1
Channels	<p>When Sensor represents multiple <i>sensing elements</i>, each <i>sensing element</i> is represented by a Channel for the Sensor.</p> <p>Channels is an XML container used to organize information for the <i>sensing elements</i>.</p>	0..1

1474 9.1.3.1.1 Attributes for Channel

1475 Channel represents each *sensing element* connected to a *sensor unit*. Table 49 defines
 1476 the attributes for Channel:

Table 49: 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

1477 9.1.3.1.2 Elements for Channel

1478 *Table 50* describes the elements provided for Channel.

Table 50: 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 50		
Element	Description	Occurrence
CalibrationDate	<p>Date upon which the <i>sensor unit</i> was last calibrated to the <i>sensor element</i>.</p> <p>The data value for CalibrationDate is provided in the CDATA for this element and MUST be represented in the W3C ISO 8601 format.</p>	0..1
NextCalibrationDate	<p>Date upon which the <i>sensor element</i> is next scheduled to be calibrated with the <i>sensor unit</i>.</p> <p>The data value for NextCalibrationDate is provided in the CDATA for this element and MUST be represented in the W3C ISO 8601 format.</p>	0..1
CalibrationInitials	<p>The initials of the person verifying the validity of the calibration data.</p> <p>The data value for CalibrationInitials is provided in the CDATA for this element and MAY be any numeric or text content.</p>	0..1

1479 *Example 10* is an example of the configuration data for Sensor that is modeled as a Com-
1480 ponent. It has Configuration data for the *sensor unit*, one Channel named A/D:1,
1481 and two DataItems – Voltage (as a SAMPLE) and Voltage (as a CONDITION or
1482 alarm).

Example 10: Example of configuration data for Sensor

```

1483 1 <Sensor id="sensor" name="sensor">
1484 2   <Configuration>
1485 3     <SensorConfiguration>
1486 4       <FirmwareVersion>2.02</FirmwareVersion>
1487 5       <CalibrationDate>2010-05-16</CalibrationDate>
1488 6       <NextCalibrationDate>2010-05-16</NextCalibrationDate>
1489 7       <CalibrationInitials>WS</CalibrationInitials>
1490 8     <Channels>
1491 9       <Channel number="1" name="A/D:1">
1492 10        <Description>A/D With Thermister</Description>
1493 11      </Channel>

```

```

1494 12      </Channels>
1495 13      </SensorConfiguration>
1496 14  </Configuration>
1497 15  <DataItems>
1498 16      <DataItem category="CONDITION" id="senvc"
1499 17          type="VOLTAGE" />
1500 18      <DataItem category="SAMPLE" id="senv"
1501 19          type="VOLTAGE" units="VOLT" subType="DIRECT" />
1502 20  </DataItems>
1503 21 </Sensor>

```

1504 9.2 Relationships

1505 Relationships is an XML container that organizes information defining the associ-
 1506 ation between pieces of equipment that function independently but together perform a
 1507 manufacturing operation. Relationships may also define the association between
 1508 components within a piece of equipment.

1509 Relationships may be modeled as part of a Device or a Component *Structural*
 1510 *Element*.

1511 Relationships contains one or more Relationship XML elements.

Table 51: 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

1512 9.2.1 Relationship

1513 Relationship is an XML element that describes the association between two pieces
 1514 of equipment that function independently but together perform a manufacturing operation.
 1515 Relationship may also be used to define the association between two components
 1516 within a piece of equipment.

1517 Relationship is an abstract type XML element, Relationship will be replaced in

1518 the XML document by specific Relationship types. XML elements representing Re-
1519 lationship are described in *Section 9.2.1.1 - DeviceRelationship* and *Section 9.2.1.2 -*
1520 *ComponentRelationship*.

1521 A separate Relationship type element **MAY** be defined to describe each pair of as-
1522 sociations with a piece of equipment or between Component elements within a piece of
1523 equipment.

1524 Pieces of equipment may only be associated with other pieces of equipment and Compo-
1525 nent elements may only be associated with other Component elements within a specific
1526 piece of equipment.

1527 The XML schema diagram in *Figure 20* represents the structure of the Relationship
1528 XML element.

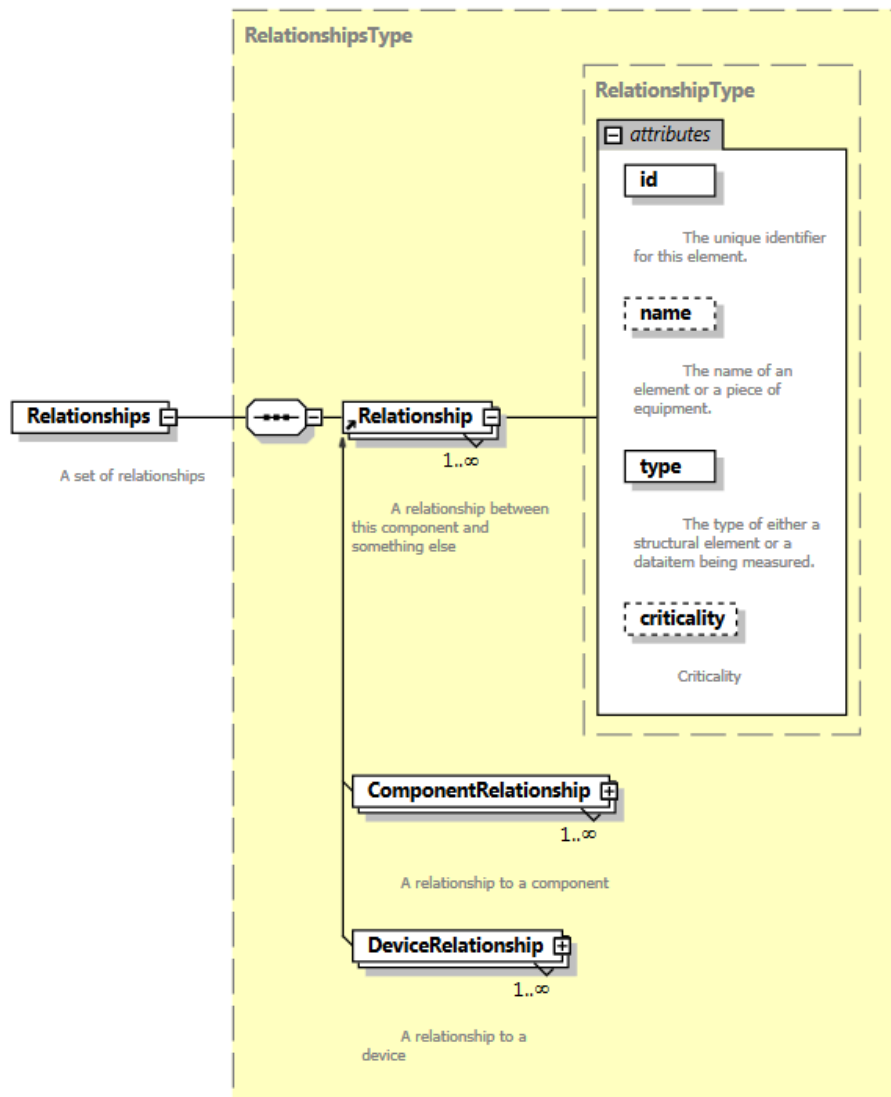


Figure 20: Relationship Diagram

1529 **9.2.1.1 DeviceRelationship**

1530 DeviceRelationship describes the association between two pieces of equipment that
1531 function independently but together perform a manufacturing operation.

1532 The XML schema diagram in *Figure 21* represents the structure of a DeviceRela-
1533 tionship XML element showing the attributes defined for DeviceRelationship.

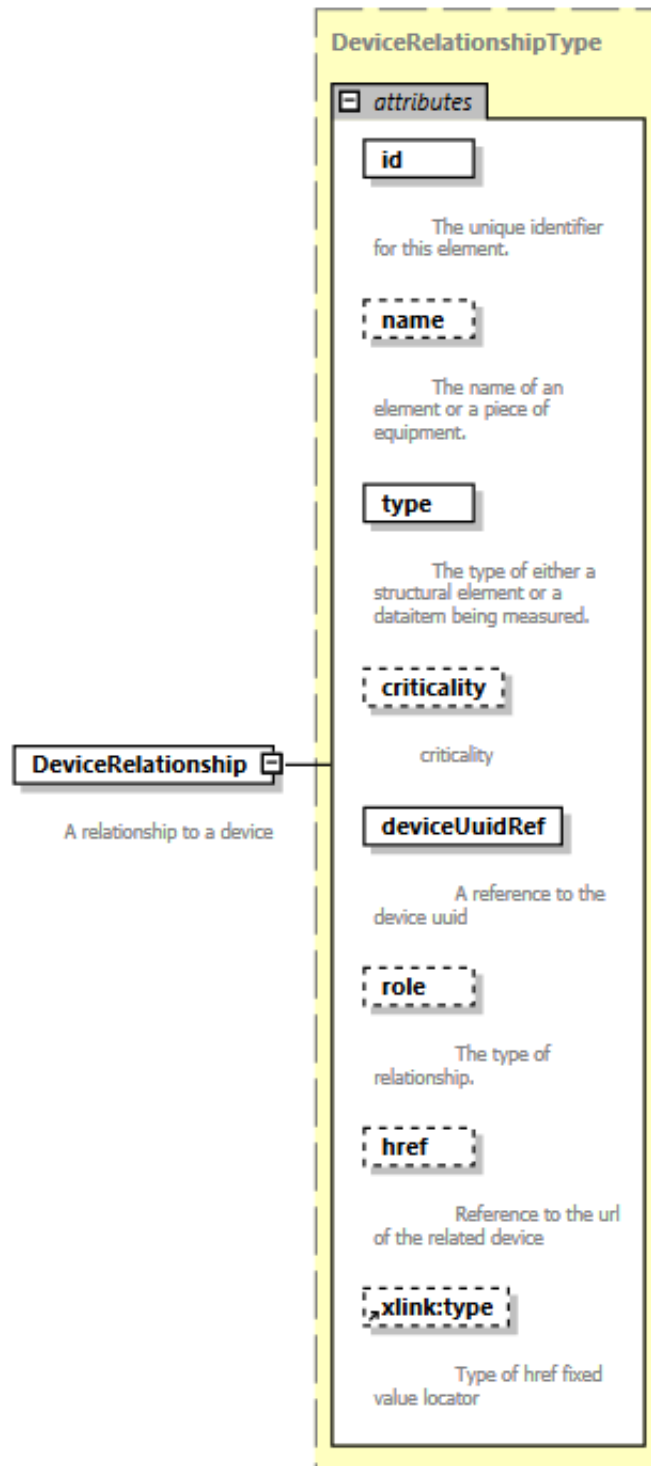


Figure 21: DeviceRelationship Diagram

1534 The *Table 52* lists the attributes defined for the `DeviceRelationship` element.

Table 52: Attributes for `DeviceRelationship`

Attribute	Description	Occurrence
<code>id</code>	<p>The unique identifier for this <code>DeviceRelationship</code>.</p> <p><code>id</code> is a required attribute.</p> <p>The <code>id</code> attribute MUST be unique within the <code>MTConnectDevices</code> document.</p> <p>An XML ID-type.</p>	1
<code>name</code>	<p>The name associated with this <code>DeviceRelationship</code>.</p> <p><code>name</code> is provided as an additional human readable identifier for this <code>DeviceRelationship</code>.</p> <p><code>name</code> is an optional attribute.</p> <p>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.</p> <p><code>type</code> is a required attribute.</p> <p>The value provided for <code>type</code> MUST be one of the following values:</p> <p>PARENT: This piece of equipment functions as a parent in the relationship with the associated piece of equipment.</p> <p>CHILD: This piece of equipment functions as a child in the relationship with the associated piece of equipment.</p> <p>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 52		
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> <p>CRITICAL: The services or functions provided by the associated piece of equipment is required for the operation of this piece of equipment.</p> <p>NONCRITICAL: The services or functions provided by the associated piece of equipment is not required for the operation of this piece of equipment.</p>	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 52		
Attribute	Description	Occurrence
<code>role</code>	<p>Defines the services or capabilities that the referenced piece of equipment provides relative to this piece of equipment.</p> <p><code>role</code> is an optional attribute.</p> <p>The value provided for <code>role</code> MUST be one of the following values:</p> <p>SYSTEM: The associated piece of equipment performs the functions of a <code>System</code> for this piece of equipment. In <code>MTConnect</code>, <code>System</code> 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.</p> <p>AUXILIARY: The associated piece of equipment performs the functions as an <code>Auxiliary</code> for this piece of equipment. In <code>MTConnect</code>, <code>Auxiliary</code> extends the capabilities of a piece of equipment, but is not required for the equipment to function.</p>	0..1
<code>href</code>	<p>A URI identifying the <i>Agent</i> that is publishing information for the associated piece of equipment. <code>href</code> MUST also include the UUID for that specific piece of equipment.</p> <p><code>href</code> is of type <code>xlink:href</code> from the W3C XLink specification: (https://www.w3.org/TR/xlink11/).</p> <p><code>href</code> is an optional attribute.</p>	0..1
<code>xlink:type</code>	<p>The XLink <code>type</code> attribute MUST have a fixed value of <code>locator</code> as defined in W3C XLink 1.1 https://www.w3.org/TR/xlink11/ <i>section 5.4 Locator Attribute</i> (<code>href</code>).</p> <p>If the <code>href</code> 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

1535 9.2.1.2 ComponentRelationship

1536 ComponentRelationship describes the association between two components within
 1537 a piece of equipment that function independently but together perform a capability or
 1538 service within a piece of equipment.

1539 The XML schema in *Figure 22* represents the structure of a ComponentRelation-
 1540 ship XML element showing the attributes defined for ComponentRelationship.

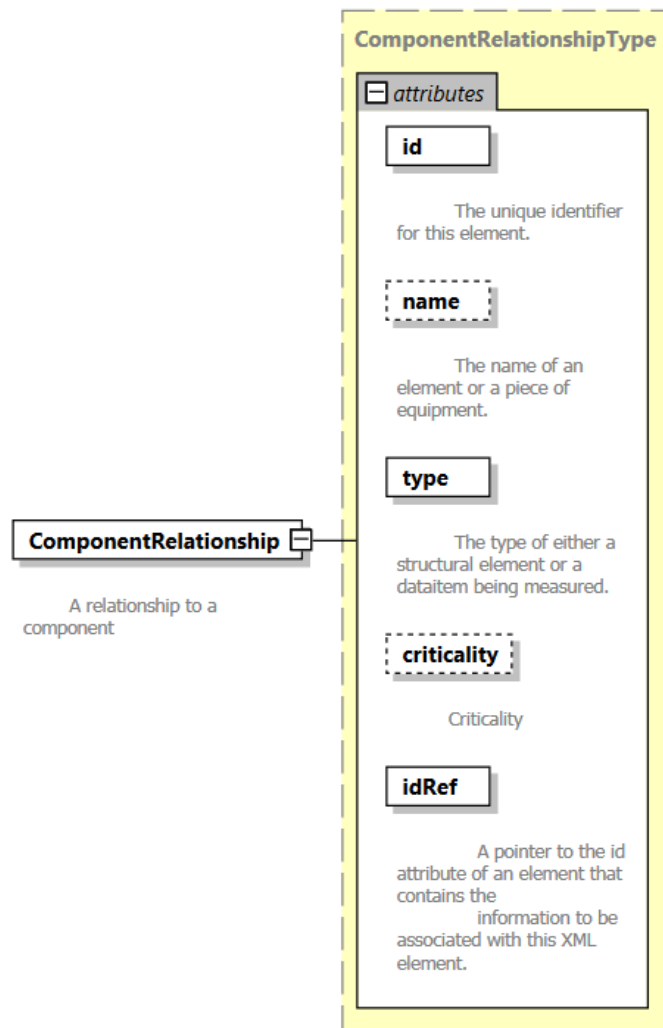


Figure 22: ComponentRelationship Diagram

1541 The *Table 53* lists the attributes defined for the ComponentRelationship element.

Table 53: 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> <p>PARENT: This component functions as a parent in the relationship with the associated component element.</p> <p>CHILD: This component functions as a child in the relationship with the associated component element.</p> <p>PEER: This component functions as a peer which provides equal functionality and capabilities in the relationship with the associated component element.</p>	1

Continuation of Table 53		
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> <p>CRITICAL: The services or functions provided by the associated component element is required for the operation of this piece of equipment.</p> <p>NONCRITICAL: The services or functions provided by the associated component element is not required for the operation of this piece of equipment.</p>	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

1542 9.3 Specifications

1543 Specifications is an XML container in the Configuration of a Component
 1544 that contains one or more Specification elements describing the design characteris-
 1545 tics for a piece of equipment.

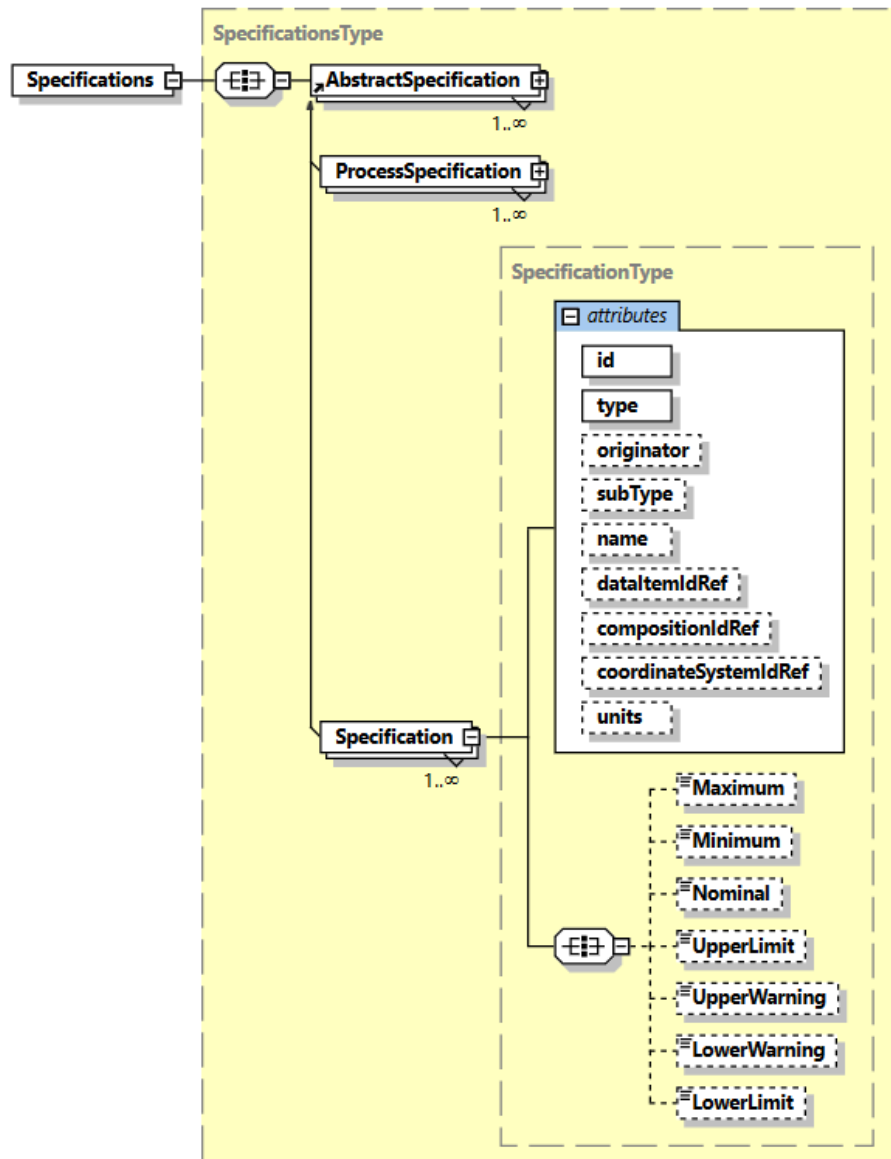


Figure 23: Specifications Diagram

1546 9.3.1 Specification

1547 Specification elements define information describing the design characteristics for
1548 a piece of equipment.

1549 9.3.1.1 Attributes for Specification

1550 *Table 54* lists the attributes defined to provide information for a Specification ele-
1551 ment.

Table 54: Attributes for Specification

Attribute	Description	Occurrence
type	Same as <code>DataItem</code> type. See <i>Section 8 - Listing of Data Items</i> .	1
subType	Same as <code>DataItem</code> subtypes. See <i>Section 8 - Listing of Data Items</i> .	0..1
dataItemIdRef	A reference to the <code>id</code> attribute of the <code>DataItem</code> associated with this element.	0..1
units	Same as <code>DataItem</code> units. See <i>Section 7.2.2.5 - units Attribute for DataItem</i> .	0..1
compositionIdRef	A reference to the <code>id</code> attribute of the <code>Composition</code> associated with this element.	0..1
name	The name provides additional meaning and differentiates between Specifications. A name MUST exist when two Specifications have the same type and subType within a Component.	0..1
coordinateSystemIdRef	References the <code>CoordinateSystem</code> for geometric Specification elements.	0..1

Continuation of Table 54		
Attribute	Description	Occurrence
id	<p>The unique identifier for this Specification. The id attribute MUST be unique within the MTConnectDevices document.</p> <p>An XML ID-type.</p>	0..1
originator	<p>A reference to the creator of the Specification.</p> <p>The values reported for originator are:</p> <p>MANUFACTURER: The manufacturer of a piece of equipment or Component.</p> <p>USER: The owner or implementer of a piece of equipment or Component.</p> <p>Note: The default value for originator is MANUFACTURER.</p>	0..1

1552 9.3.1.2 Elements for Specification

1553 *Table 55* lists the elements defined to provide information for a *Specification* ele-
 1554 ment.

Table 55: Elements for Specification

Element	Description	Occurrence
Maximum	A numeric upper constraint.	0..1
UpperLimit	The upper conformance boundary for a variable. Note: immediate concern or action may be required.	0..1
UpperWarning	The upper boundary indicating increased concern and supervision may be required.	0..1
Nominal	The ideal or desired value for a variable.	0..1
LowerWarning	The lower boundary indicating increased concern and supervision may be required.	0..1
LowerLimit	The lower conformance boundary for a variable. Note: immediate concern or action may be required.	0..1
Minimum	A numeric lower constraint.	0..1

1555 9.3.2 ProcessSpecification

1556 *ProcessSpecification* provides information used to assess the conformance of a
 1557 variable to process requirements.

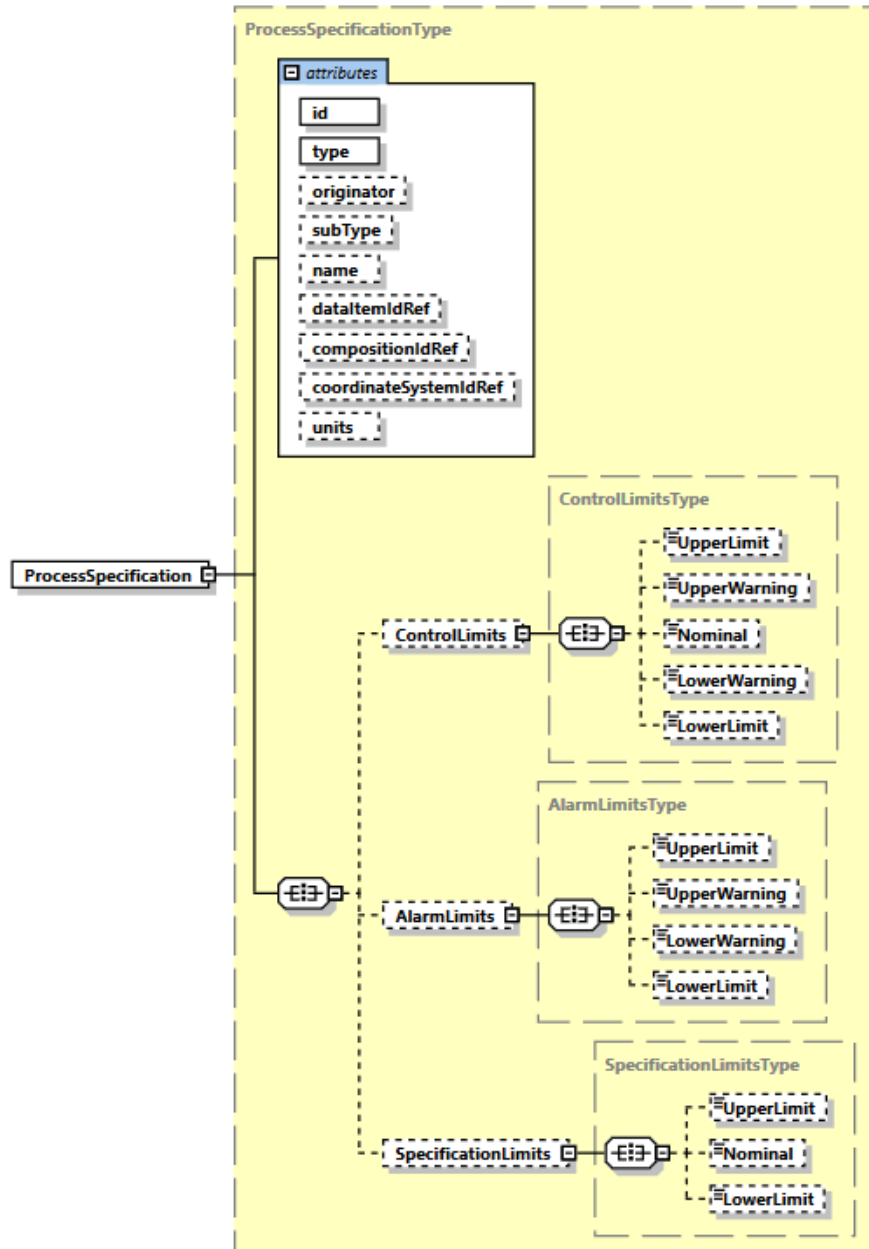


Figure 24: ProcessSpecification Diagram

1558 See *Section 9.3.1.1 - Attributes for Specification* for definitions on attributes of `ProcessSpecification`.
 1559

1560 **9.3.2.1 Elements for ProcessSpecification**

1561 *Table 56* lists the elements defined to provide information for a `ProcessSpecification` element.
 1562

Table 56: Elements for ProcessSpecification

Element	Description	Occurrence
<code>ControlLimits</code>	A set of limits used to indicate whether a process variable is stable and in control.	0..1
<code>SpecificationLimits</code>	A set of limits defining a range of values designating acceptable performance for a variable.	0..1
<code>AlarmLimits</code>	A set of limits used to trigger warning or alarm indicators.	0..1

1563 **9.3.2.2 ControlLimits**

1564 A set of limits used to indicate whether a process variable is stable and in control.

1565 **9.3.2.2.1 Elements for ControlLimits**

Table 57: Elements for ControlLimits

Element	Description	Occurrence
<code>UpperLimit</code>	The upper conformance boundary for a variable. Note: immediate concern or action may be required.	0..1
<code>UpperWarning</code>	The upper boundary indicating increased concern and supervision may be required.	0..1
<code>Nominal</code>	The ideal or desired value for a variable.	0..1
<code>LowerWarning</code>	The lower boundary indicating increased concern and supervision may be required.	0..1

Continuation of Table 57		
Element	Description	Occurrence
LowerLimit	The lower conformance boundary for a variable. Note: immediate concern or action may be required.	0..1

1566 9.3.2.3 SpecificationLimits

1567 A set of limits defining a range of values designating acceptable performance for a vari-
1568 able.

1569 9.3.2.3.1 Elements for SpecificationLimits

Table 58: Elements for SpecificationLimits

Element	Description	Occurrence
UpperLimit	The upper conformance boundary for a variable. Note: immediate concern or action may be required.	0..1
Nominal	The ideal or desired value for a variable.	0..1
LowerLimit	The lower conformance boundary for a variable. Note: immediate concern or action may be required.	0..1

1570 9.3.2.4 AlarmLimits

1571 A set of limits used to trigger warning or alarm indicators.

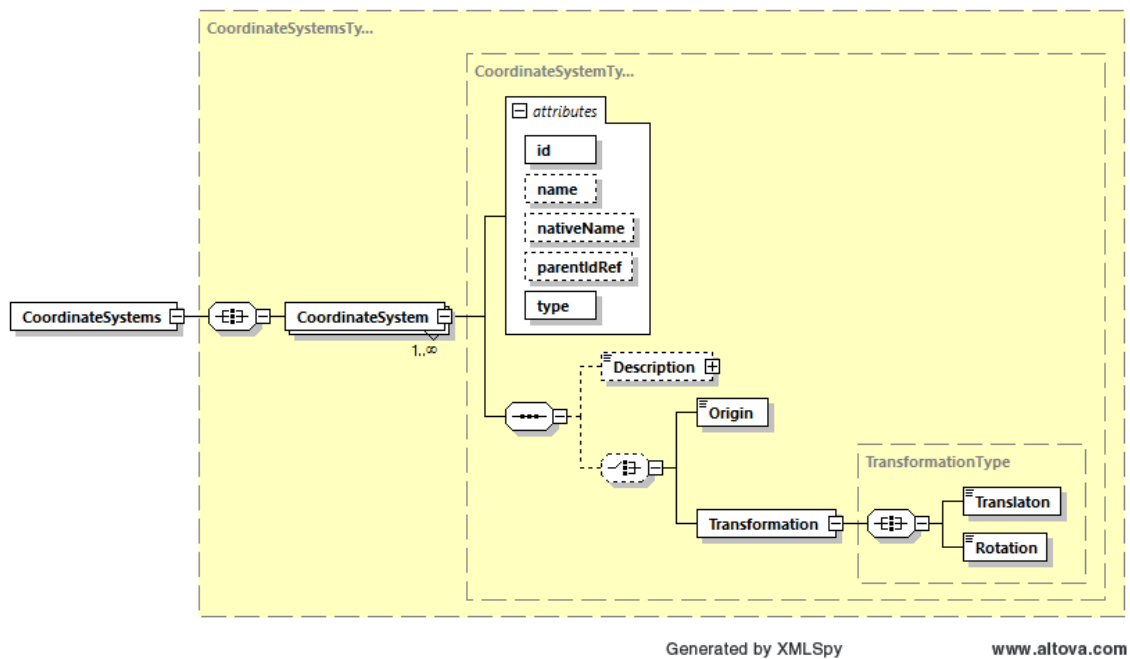
1572 9.3.2.4.1 Elements for AlarmLimits

Table 59: Elements for AlarmLimits

Element	Description	Occurrence
UpperLimit	The upper conformance boundary for a variable. Note: immediate concern or action may be required.	0..1
UpperWarning	The upper boundary indicating increased concern and supervision may be required.	0..1
LowerWarning	The lower boundary indicating increased concern and supervision may be required.	0..1
LowerLimit	The lower conformance boundary for a variable. Note: immediate concern or action may be required.	0..1

1573 9.4 CoordinateSystems

1574 CoordinateSystems aggregates CoordinateSystem configurations for a Com-
1575 ponent.

**Figure 25:** CoordinateSystems Diagram

1576 9.4.1 CoordinateSystem

1577 A `CoordinateSystem` is a reference system that associates a unique set of n parameters with each point in an n -dimensional space. *Ref: ISO 10303-218:2004*

1579 9.4.1.1 Attributes for CoordinateSystem

1580 *Table 60* lists the attributes defined to provide information for a `CoordinateSystem` element.

Table 60: Attributes for `CoordinateSystem`

Attribute	Description	Occurrence
<code>id</code>	The unique identifier for this element.	1
<code>name</code>	The name of the coordinate system. If more than one <code>CoordinateSystem</code> elements have the same <code>type</code> for the same <code>Component</code> , then the name attribute MUST be provided. Otherwise, the name attribute is optional. <code>name</code> provides as an additional human-readable identifier in addition to the <code>id</code> .	0..1
<code>nativeName</code>	The manufacturer's name or users name for the coordinate system.	0..1
<code>parentIdRef</code>	A pointer to the <code>id</code> attribute of the parent <code>CoordinateSystem</code> .	0..1
<code>type</code>	The type of coordinate system.	1

1582 9.4.1.1.1 CoordinateSystem types

1583 *Table 61* defines the various types of coordinate systems.

Table 61: CoordinateSystem types

type	Description
WORLD	stationary coordinate system referenced to earth, which is independent of the robot motion. <i>Ref:ISO 9787:2013</i> For non-robotic devices, stationary coordinate system referenced to earth, which is independent of the motion of a piece of equipment.
BASE	coordinate system referenced to the base mounting surface. <i>Ref:ISO 9787:2013</i> A base mounting surface is a connection surface between the arm and its supporting structure. <i>Ref:ISO 9787:2013</i> For non-robotic devices, it is the connection surface between the device and its supporting structure.
OBJECT	coordinate system referenced to the object. <i>Ref:ISO 9787:2013</i>
TASK	coordinate system referenced to the site of the task. <i>Ref:ISO 9787:2013</i>
MECHANICAL_INTERFACE	coordinate system referenced to the mechanical interface. <i>Ref:ISO 9787:2013</i>
TOOL	coordinate system referenced to the tool or to the end effector attached to the mechanical interface. <i>Ref:ISO 9787:2013</i>
MOBILE_PLATFORM	coordinate system referenced to one of the components of a mobile platform. <i>Ref:ISO 8373:2012</i>
MACHINE	coordinate system referenced to the home position and orientation of the primary axes of a piece of equipment.
CAMERA	coordinate system referenced to the sensor which monitors the site of the task. <i>Ref:ISO 9787:2013</i>

1584 9.4.1.2 Elements for CoordinateSystem

1585 *Table 62* lists the elements defined to provide information for a `CoordinateSystem`
 1586 element.

Table 62: Elements for `CoordinateSystem`

Element	Description	Occurrence
Origin	The coordinates of the origin position of a coordinate system. The coordinate MUST be in <code>MILLIMETER_3D</code> .	0..1
Transformation	The process of transforming to the origin position of the coordinate system from a parent coordinate system using <code>Translation</code> and <code>Rotation</code> .	0..1
Description	The natural language description of the <code>CoordinateSystem</code> .	0..1

1587 Notes: Only one of `Origin` or `Transformation` can be defined for a `Coordi-`
 1588 `nateSystem`.

1589 9.4.1.2.1 Elements for Transformation

1590 *Table 63* lists the elements defined to provide information for a `Transformation` ele-
 1591 ment.

Table 63: Elements for `Transformation`

Element	Description	Occurrence
Translation	Translations along X, Y, and Z axes are expressed as x,y, and z respectively within a 3-dimensional vector. The values MUST be given in <code>MILLIMETER_3D</code> .	0..1

Continuation of Table 63		
Element	Description	Occurrence
Rotation	<p>Rotations about X, Y, and Z axes are expressed in A, B, and C respectively within a 3-dimensional vector.</p> <p>The values MUST be given in DEGREE_3D.</p> <p>Positive A, B, and C are in the directions to advance right-hand screws in the positive X, Y, and Z directions, respectively. <i>Ref:ISO 9787:2013</i></p>	0..1

1592 9.5 Motion

1593 Motion defines the movement of the Component relative to a coordinate system. Mo-
 1594 tion specifies the kinematic chain of the Components.

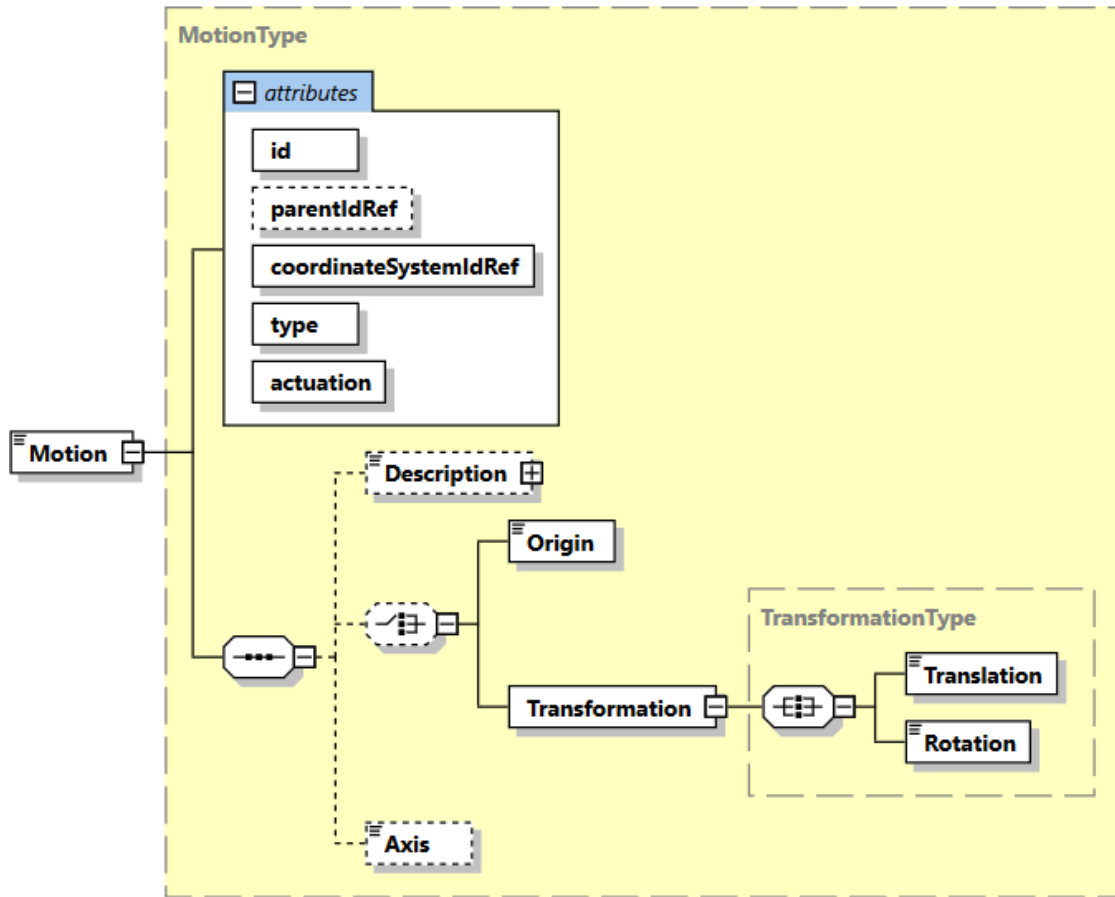


Figure 26: Motion Diagram

1595 9.5.1 Attributes for Motion

1596 *Table 64* lists the attributes defined to provide information for a `Motion` element.

Table 64: Attributes for Motion

Attribute	Description	Occurrence
id	The unique identifier for this element.	1

Continuation of Table 64		
Attribute	Description	Occurrence
parentIdRef	A pointer to the <code>id</code> attribute of the parent <code>Motion</code> . The kinematic chain connects all components using the parent relations. All motion is connected to the motion of the parent. The first node in the chain will not have a parent.	0..1
coordinateSystemIdRef	The coordinate system within which the kinematic motion occurs.	1
type	Describes the type of motion.	1
actuation	Describes if this <code>Component</code> is actuated directly or indirectly as a result of other motion.	1

1597 9.5.1.1 Motion types

1598 *Table 65* defines the types of `Motion`.

Table 65: Motion types

type	Description
REVOLUTE	Rotates around an axis with a fixed range of motion.
CONTINUOUS	Revolves around an axis with a continuous range of motion.
PRISMATIC	Sliding linear motion along an axis with a fixed range of motion.
FIXED	The axis does not move.

1599 9.5.1.2 Motion actuation types

1600 *Table 66* defines the types of actuation of `Motion`.

Table 66: Motion actuation types

type	Description
DIRECT	The movement is initiated by the Component.
VIRTUAL	The motion is computed and is used for expressing an imaginary movement.
NONE	There is no actuation of this Axis. Note: Actuation of NONE can be either a derived REVOLUTE or PRISMATIC motion or static FIXED relationship.

1601 9.5.2 Elements for Motion

1602 Table 67 lists the elements defined to provide information for a Motion element.

Table 67: Elements for Motion

Element	Description	Occurrence
Description	An element that can contain any descriptive content.	0..1
Axis	Axis defines the axis along or around which the Component moves relative to a coordinate system. The value of Axis MUST be in UNIT_VECTOR_3D.	1
Origin	A fixed point from which measurement or motion commences. The value MUST be in MILLIMETER_3D.	0..1
Transformation	The Transformation of the parent Origin or Transformation using Translation and Rotation. At a minimum, a Translation or Rotation MUST be given. See Section 9.4.1.2.1 - Elements for Transformation for definitions of Translation and Rotation.	0..1

1603 Notes: Only one of `Origin` or `Transformation` can be defined for a `Motion`.

1604 9.6 SolidModel

1605 A `SolidModel` is a `Configuration` that references a file with the three-dimensional
 1606 geometry of the `Component` or `Composition`. The geometry **MAY** have a transfor-
 1607 mation and a scale to position the `Component` with respect to the other `Components`.
 1608 A geometry file can contain a set of assembled items, in this case, the `SolidModel`
 1609 reference the `id` of the assembly model file and the specific item within that file.

1610 The `SolidModel` **MAY** provide a translation, rotation, and scale to correctly place it
 1611 relative to the other geometries in the machine. If the `Component` can move and has
 1612 a `Motion Configuration`, the `SolidModel` will move when the `Component` or
 1613 `Composition` moves.

1614 Either an `href` or a `solidModelIdRef` and an `itemRef` **MUST** be specified.

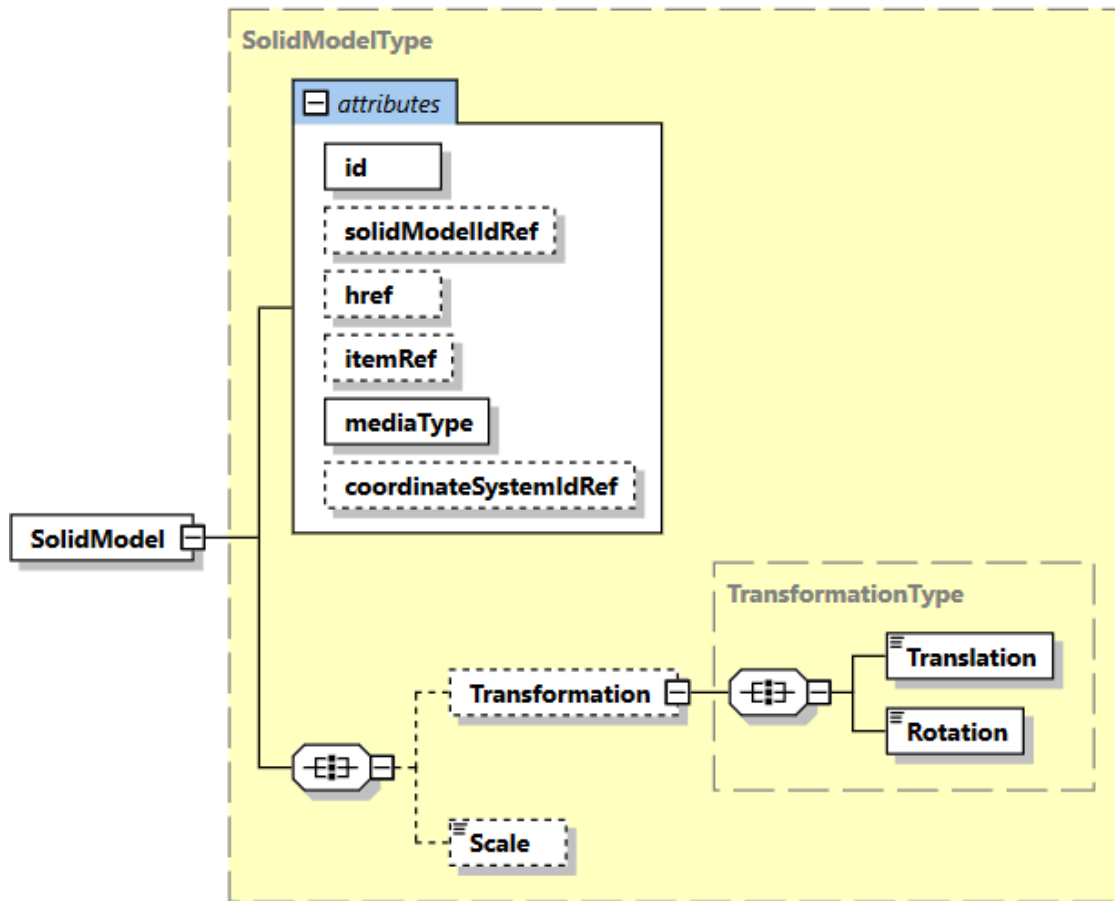


Figure 27: SolidModel Diagram

1615 9.6.1 Attributes for SolidModel

1616 Table 68 lists the attributes defined to provide information for a `SolidModel` element.

Table 68: Attributes for SolidModel

Attribute	Description	Occurrence
<code>id</code>	The unique identifier for this entity within the <code>MTConnectDevices</code> document.	1
<code>solidModelIdRef</code>	The associated model file if an item reference is used.	0..1

Continuation of Table 68		
Attribute	Description	Occurrence
href	The URL giving the location of the Solid Model. If not present, the model referenced in the <code>solidModelIdRef</code> is used. href is of type <code>xlink:href</code> from the W3C XLink specification.	0..1
itemRef	The reference to the item within the model within the related geometry. A <code>solidModelIdRef</code> MUST be given. Note: Item defined in ASME Y14.100 - A nonspecific term used to denote any unit or product, including materials, parts, assemblies, equipment, accessories, and computer software.	0..1
mediaType	The format of the referenced document.	1
coordinateSystemIdRef	A reference to the coordinate system for this <code>SolidModel</code> .	0..1

1617 9.6.1.1 SolidModel mediaType

1618 Table 69 defines the type of `mediaType` for `SolidModel`.

Table 69: SolidModel mediaType

type	Description
STEP	ISO 10303 STEP AP203 or AP242 format.
STL	Stereolithography file format.
GDML	Geometry Description Markup Language.
OBJ	Wavefront OBJ file format.
COLLADA	ISO 17506.
IGES	Initial Graphics Exchange Specification.

Continuation of Table 69	
type	Description
3DS	Autodesk file format.
ACIS	Dassault file format.
X_T	Parasolid XT Siemens data interchange format.

1619 9.6.2 Elements for SolidModel

1620 *Table 70* lists the elements defined to provide information for a `SolidModel` element.

Table 70: Elements for SolidModel

Element	Description	Occurrence
Transformation	<p>The translation of the origin to the position and orientation.</p> <p>At a minimum, a <code>Translation</code> or <code>Rotation</code> MUST be given.</p> <p>See <i>Section 9.4.1.2.1 - Elements for Transformation</i> for definitions of <code>Translation</code> and <code>Rotation</code>.</p>	0..1
Scale	<p>The <code>SolidModel</code> <code>Scale</code> is either a single multiplier applied to all three dimensions or a three space multiplier given in the X, Y, and Z dimensions in the coordinate system used for the <code>SolidModel</code>.</p>	0..1

1621 Appendices

1622 A Bibliography

- 1623 Engineering Industries Association. *EIA Standard - EIA-274-D*, Interchangeable Variable,
 1624 Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically
 1625 Controlled Machines. Washington, D.C. 1979.
- 1626 ISO TC 184/SC4/WG3 N1089. *ISO/DIS 10303-238*: Industrial automation systems and
 1627 integration Product data representation and exchange Part 238: Application Protocols: Ap-
 1628 plication interpreted model for computerized numerical controllers. Geneva, Switzerland,
 1629 2004.
- 1630 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
 1631 tems and integration – Physical device control – Data model for computerized numerical
 1632 controllers – Part 10: General process data. Geneva, Switzerland, 2004.
- 1633 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
 1634 tems and integration – Physical device control – Data model for computerized numerical
 1635 controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.
- 1636 International Organization for Standardization. *ISO 6983/1* – Numerical Control of ma-
 1637 chines – Program format and definition of address words – Part 1: Data format for posi-
 1638 tioning, line and contouring control systems. Geneva, Switzerland, 1982.
- 1639 Electronic Industries Association. *ANSI/EIA-494-B-1992*, 32 Bit Binary CL (BCL) and
 1640 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines.
 1641 Washington, D.C. 1992.
- 1642 National Aerospace Standard. *Uniform Cutting Tests - NAS Series: Metal Cutting Equip-*
 1643 *ment Specifications*. Washington, D.C. 1969.
- 1644 International Organization for Standardization. *ISO 10303-11*: 1994, Industrial automa-
 1645 tion systems and integration Product data representation and exchange Part 11: Descrip-
 1646 tion methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.
- 1647 International Organization for Standardization. *ISO 10303-21*: 1996, Industrial automa-
 1648 tion systems and integration – Product data representation and exchange – Part 21: Imple-
 1649 mentation methods: Clear text encoding of the exchange structure. Geneva, Switzerland,
 1650 1996.
- 1651 H.L. Horton, F.D. Jones, and E. Oberg. *Machinery's Handbook*. Industrial Press, Inc.

- 1652 New York, 1984.
- 1653 International Organization for Standardization. *ISO 841-2001: Industrial automation sys-*
1654 *tems and integration - Numerical control of machines - Coordinate systems and motion*
1655 *nomenclature*. Geneva, Switzerland, 2001.
- 1656 ASME B5.57: *Methods for Performance Evaluation of Computer Numerically Controlled*
1657 *Lathes and Turning Centers*, 1998.
- 1658 ASME/ANSI B5.54: *Methods for Performance Evaluation of Computer Numerically Con-*
1659 *trolled Machining Centers*. 2005.
- 1660 OPC Foundation. *OPC Unified Architecture Specification, Part 1: Concepts Version 1.00*.
1661 July 28, 2006.
- 1662 IEEE STD 1451.0-2007, *Standard for a Smart Transducer Interface for Sensors and Ac-*
1663 *tuators – Common Functions, Communication Protocols, and Transducer Electronic Data*
1664 *Sheet (TEDS) Formats*, IEEE Instrumentation and Measurement Society, TC-9, The In-
1665 *stitute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH99684,*
1666 *October 5, 2007.*
- 1667 IEEE STD 1451.4-1994, *Standard for a Smart Transducer Interface for Sensors and Ac-*
1668 *tuators – Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet*
1669 *(TEDS) Formats*, IEEE Instrumentation and Measurement Society, TC-9, The Institute of
1670 *Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH95225, December*
1671 *15, 2004.*



MTConnect® Standard

Part 3.0 – Streams Information Model

Version 1.8.0

Prepared for: MTConnect Institute
Prepared on: September 6, 2021

MTConnect® is a registered trademark of AMT - The Association for Manufacturing Technology. Use of *MTConnect* is limited to use as specified on <http://www.mtconnect.org/>.

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	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	Valid Data Values for Sample	38
5.3.4	Unavailability of Valid Data Values for Sample	40
5.4	Events Container	40
5.5	Event Data Entities	41
5.5.1	XML Schema Structure for Event	42
5.5.2	Attributes for Event	43
5.5.3	Valid Data Values for Event	44
5.5.4	Unavailability of Valid Data Value for Event	46
5.6	Representations	46

5.6.1	Observations for DataItem with representation of TIME_SERIES	47
5.6.1.1	XML Schema for Time Series Observation	47
5.6.1.2	Attributes for Time Series Observation	49
5.6.2	Observations for DataItem with representation of DISCRETE (DEP- RECATED)	49
5.6.3	Observations for DataItem with representation of DATA_SET . .	49
5.6.3.1	XML Schema for Data Set Observation	50
5.6.3.2	Entry Element for Data Set Observation	51
5.6.3.3	Attributes for Entry Element for Data Set Observation .	52
5.6.3.4	Constraints for Entry Values	52
5.6.4	Management of Data Set Observations	53
5.6.5	Observations for DataItem with representation of TABLE	53
5.6.5.1	Structure of Table Observations	54
5.6.5.2	Attributes of Table Observations	56
5.6.5.3	Elements of Table Observations	56
5.6.5.3.1	Structure for Table Entry for an Observation .	56
5.6.5.3.2	Attributes for Table Entry for an Observation .	56
5.6.5.3.3	Elements for Table Cell for an Observation . .	57
5.6.5.3.4	Structure for Table Cell for an Entry	57
5.6.5.3.5	Attributes for Table Cell for an Observation . .	57
5.6.5.3.6	Constraints for Cell Values	57
5.6.5.3.7	Example Table Observation	58
5.7	Condition Container	58
5.8	Condition Data Entity	59
5.8.1	Element Names for Condition	60
5.8.2	XML Schema Structure for Condition	61
5.8.3	Attributes for Condition	62
5.8.3.1	qualifier Attribute for Condition	65
5.8.4	Valid Data Value for Condition	66
5.9	Unavailability of Fault State for Condition	66
6	Listing of Data Entities	68
6.1	Sample Element Names	68
6.2	Event Element Names	93
6.3	Types of Condition Elements	149
	Appendices	151
A	Bibliography	151

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: Event Schema Diagram	42
Figure 8: AbsTimeSeries Schema Diagram	48
Figure 9: Sample Data Set Schema Diagram	50
Figure 10:Entry Element Schema Diagram	51
Figure 11:Table Schema Diagram	55
Figure 12:Condition Schema Diagram	61

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 Event Element	41
Table 12: MTConnect Event Element	42
Table 13: Attributes for Event	43
Table 14: Attributes for Time Series Observation	49
Table 15: Attributes for Data Set Observation	50
Table 16: Elements for Data Set Observation	51
Table 17: Attributes for Entry	52
Table 18: Attributes for Table	56
Table 19: Elements for Table	56
Table 20: Elements for Table Cell	57
Table 21: Attributes for Table Cell	57
Table 22: MTConnect Condition Element Container	59
Table 23: MTConnect Condition Element	60
Table 24: Attributes for Condition	62
Table 25: Element Names for Sample	68
Table 26: Element Names for Event	94
Table 27: Element Names for Condition	150

1 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 NMTOKEN

34 The data type for XML identifiers.

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

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

39 URI

40 Stands for Universal Resource Identifier.

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

42 UUID

43 General meaning:

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

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

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

49 Used as an attribute for an XML element:

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

52 Appears in the documents in the following form: *uuid*.

53 **XML**

54 Stands for eXtensible Markup Language.

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

57 XML is the language used for all code examples in the MTConnect Standard.

58 Refer to <http://www.w3.org/XML> for more information about XML.

59 ***Adapter***

60 An optional piece of hardware or software that transforms information provided by
61 a piece of equipment into a form that can be received by an *Agent*.

62 Appears in the documents in the following form: *adapter*.

63 ***Agent***

64 Refers to an MTConnect Agent.

65 Software that collects data published from one or more piece(s) of equipment, orga-
66 nizes that data in a structured manner, and responds to requests for data from client
67 software systems by providing a structured response in the form of a *Response Doc-*
68 *ument* that is constructed using the *semantic data models* defined in the Standard.

69 Appears in the documents in the following form: *Agent*.

70 ***Attachment***

71 The connection by which one thing is associated with another.

72 ***Child Element***

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

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

76 ***Component***

77 General meaning:

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

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

Used in *Information Models*:

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

- When used as an XML container to organize *Lower Level Component* elements.

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

- When used as an abstract XML element. *Component* is replaced in a data model by a type of *Component* element. *Component* is also an XML container used to organize *Lower Level Component* elements, *Data Entities*, or both.

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

Condition

An indicator of the ability of a piece of equipment or *Component* to function to specification.

Controlled Vocabulary

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

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

Current Request

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

Data Entity

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

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

Data Set

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

Devices Information Model

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

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

114 ***Element Name***

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

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

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

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

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

122 ***Equipment Metadata***

123 See *Metadata*

124 ***Fault State***

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

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

128 ***Force***

129 A push or pull on a mass which results in an acceleration.

130 ***Information Model***

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

133 For example, an information model is used to define the structure for each *MTCon-*
134 *nect Response Document*; the definition of each piece of information within those
135 documents and the relationship between pieces of information.

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

137 ***Interaction Model***

138 Defines how information is exchanged across an *Interface* between independent sys-
139 tems.

140 ***Interface***

141 The means by which communication is achieved between independent systems.

142 ***key***

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

144 ***key-value pair***

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

148 ***Lower Level***

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

150 ***Metadata***

151 Data that provides information about other data.

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

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

157 ***MTConnect Agent***

158 See definition for *Agent*.

159 ***MTConnectDevices Response Document***

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

162 ***MTConnectStreams Response Document***

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

165 ***observation***

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

167 ***Observations Information Model***

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

170 ***Parent Element***

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

173 Appears in the documents in the following form: *Parent Element*.

174 ***Probe Request***

175 A *Probe Request* is a *Request* to an *Agent* to produce an *MTConnectDevices Re-*
176 *sponse Document* containing the *Devices Information Model*.

177 ***Request***

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

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

181 ***reset***

182 A reset is associated with an occurrence of a *Data Entity* indicated by the `reset-`
183 `Triggered` attribute. When a reset occurs, the accumulated value or statistic are
184 reverted back to their initial value. A *Data Entity* with a *Data Set* representation
185 removes all *key-value pairs*, setting the *Data Set* to an empty set.

186 ***Response Document***

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

189 ***Sample Request***

190 A *Sample Request* is a *Request* to an *Agent* to produce an *MTConnectStreams Re-*
191 *sponse Document* containing the *Observations Information Model* for a set of time-
192 stamped *observations* made by *Components*.

193 ***semantic data model***

194 A methodology for defining the structure and meaning for data in a specific logical
195 way.

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

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

199 ***sequence number***

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

202 *sequence number* is a monotonically increasing number within an instance of an
203 *Agent*.

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

205 ***Streaming Data***

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

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

209 ***Streams Information Model***

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

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

214 ***Structural Element***

215 General meaning:

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

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

219 Used to indicate hierarchy of Components:

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

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

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

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

226 ***Table***

227 A two dimensional set of values given by a set of *key-value pairs Table Entries*.

228 Each *Table Entry* contains a set of *key-value pairs* of *Table Cells*. The `Entry` and
229 `Cell` elements comprise a tabular representation of the information.

230 ***Table Cell***

231 A subdivision of a *Table Entry* representing a singular value.

232 ***Table Entry***

233 A subdivision of a *Table* containing a set of *key-value pairs* representing *Table Cells*.

234 ***Top Level***

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

237 ***Valid Data Value***

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

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

241 ***XML Schema***

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

244 **2.2 Acronyms**

245 ***AMT***

246 The Association for Manufacturing Technology

247 **2.3 MTConnect References**

248 [MTConnect Part 1.0] *MTConnect Standard Part 1.0 - Overview and Fundamentals*. Ver-
249 sion 1.8.0.

250 [MTConnect Part 2.0] *MTConnect Standard: Part 2.0 - Devices Information Model*. Ver-
251 sion 1.8.0.

252 [MTConnect Part 3.0] *MTConnect Standard: Part 3.0 - Streams Information Model*. Ver-
253 sion 1.8.0.

254 [MTConnect Part 5.0] *MTConnect Standard: Part 5.0 - Interfaces*. Version 1.8.0.

255 3 Streams Information Model

256 The *Streams Information Model* provides a representation of the data reported by a piece
 257 of equipment used for a manufacturing process, or used for any other purpose. Additional
 258 descriptive information associated with the reported data is defined in the *MTConnect-*
 259 *Devices* document, which is described in *MTConnect Standard: Part 2.0 - Devices*
 260 *Information Model*.

261 Information defined in the *Streams Information Model* allows a software application to (1)
 262 determine the value for *Data Entities* returned from a piece of equipment and (2) interpret
 263 the data associated with those *Data Entities* with the same meaning, value, and context
 264 that it had at its original source. To do this, the software application issues one of two
 265 HTTP requests to an *Agent* associated with a piece of equipment. They are:

- 266 • *sample*: Returns a designated number of time stamped *Data Entities* from an *Agent*
 267 associated with a piece of equipment; subject to any HTTP filtering associated with
 268 the request. See *Section 8.3.3 of MTConnect Standard Part 1.0 - Overview and Fun-*
 269 *damentals* of the *MTConnect Standard* for details on the *sample* HTTP request.
- 270 • *current*: Returns a snapshot of either the most recent values or the values at a
 271 given sequence number for all *Data Entities* associated with a piece of equipment
 272 from an *Agent*; subject to any HTTP filtering associated with the request. See *Sec-*
 273 *tion 8.3.2 of MTConnect Standard Part 1.0 - Overview and Fundamentals* of the
 274 *MTConnect Standard* for details on the *current* HTTP request.

275 An *Agent* responds to either the *sample* or *current* HTTP request with an
 276 *MTConnectStreams* XML document. This document contains information describing
 277 *Data Entities* reported by an *Agent* associated with a piece of equipment. A client software
 278 application may correlate the information provided in the *MTConnectStreams* XML
 279 document with the physical and logical structure for that piece of equipment defined in the
 280 *MTConnectDevices* document to form a clear and unambiguous understanding of the
 281 information provided. (See details on the structure for a piece of equipment described in
 282 *MTConnect Standard: Part 2.0 - Devices Information Model*).

283 The *MTConnectStreams* XML document is comprised of two sections: *Header* and
 284 *Streams*.

285 The *Header* section contains protocol related information as defined in *Section 6.5 of*
 286 *MTConnect Standard Part 1.0 - Overview and Fundamentals* of the *MTConnect Standard*.

287 The *Streams* section of the *MTConnectStreams* document contains a
 288 *DeviceStream* XML container for each piece of equipment represented in the docu-

ment. Each `DeviceStream` container is comprised of two primary types of XML elements – *Structural Elements* and *Data Entities*. The contents of the `DeviceStream` container are described in detail in this document, *MTConnect Standard: Part 3.0 - Streams Information Model* of the MTConnect Standard.

Structural Elements are defined for both the `MTConnectDevices` and the `MTConnectStreams` XML documents. These *Structural Elements* are used to provide a logical organization of the information provided in each document. While used for a similar purpose, the *Structural Elements* in the `MTConnectStreams` document are specifically designed to be distinctly different from those in the `MTConnectDevices` document:

- `MTConnectDevices` document: *Structural Elements* organize information that represents the physical and logical parts and sub-parts of a piece of equipment. (See *MTConnect Standard: Part 2.0 - Devices Information Model*, Section 4 of the MTConnect Standard for more details on *Structural Elements* used in the `MTConnectDevices` document).
 - `MTConnectStreams` document: *Structural Elements* provide the structure to organize the data returned from a piece of equipment and establishes the proper context for that data. The *Structural Elements* specifically defined for use in the `MTConnectStreams` document are `DeviceStream` (see *Section 4.2 - DeviceStream*) and `ComponentStream` (see *Section 4.3 - ComponentStream*).
- `DeviceStream` and `ComponentStream` elements have a direct correlation to each of the *Structural Elements* defined in the `MTConnectDevices` document.

Data Entities that describe data reported by a piece of equipment are also defined for both the `MTConnectDevices` and the `MTConnectStreams` XML documents. The *Data Entities* provided in both documents directly relate to each other. However, *Data Entities* are used for different purposes in each document:

- `MTConnectDevices` document: *Data Entity* elements define the data that may be returned from a piece of equipment. *MTConnect Standard: Part 2.0 - Devices Information Model*, Sections 7 and 8 lists the possible *Data Entity* XML elements that can be returned in a `MTConnectDevices` document.
- `MTConnectStreams` document: *Data Entity* elements provide the data reported by a piece of equipment. This data is organized in separate `ComponentStream` XML containers for each of the *Structural Elements* defined in the `MTConnectDevices` document associated with the data that is reported by a piece of equipment.

322 Within each `ComponentStream` XML container in the `MTConnectStreams` docu-
323 ment, *Data Entities* are organized into three types of XML container elements - `Samples`,
324 `Events`, and `Conditions`. (See *Section 5 - Data Entities* and *Section 6 - Listing of*
325 *Data Entities* for more information on these elements.)

326 4 Structural Elements for MTConnectStreams

327 *Structural Elements* are XML elements that form the logical structure for the MTCon-
328 nectStreams XML document. These elements are used to organize the information
329 and data that is reported by an *Agent* for a piece of equipment. See *Figure 1* for an
330 overview of the *Structural Elements* used in an MTConnectStreams document.

331 The first, or highest level, *Structural Element* in an MTConnectStreams XML docu-
332 ment is *Streams*. *Streams* is a container type XML element used to group the data
333 reported from one or more pieces of equipment into a single XML document. *Streams*
334 **MUST** always appear in the MTConnectStreams document.

335 *DeviceStream* is the next *Structural Element* in the MTConnectStreams document.
336 *DeviceStream* is also a XML container type element. A separate *DeviceStream*
337 container is used to organize the information and data reported by each piece of equip-
338 ment represented in the MTConnectStreams document. There **MUST** be at least one
339 *DeviceStream* element in the *Streams* container.

340 A *DeviceStream* element provides the data reported by a piece of equipment. Each
341 *DeviceStream* element **MUST** contain the attributes *name* and *uuid* to correlate the
342 *DeviceStream* with a specific *Device* defined in the MTConnectDevices docu-
343 ment. Once the *DeviceStream* element is associated with a specific piece of equipment
344 based on this identity, all data reported by that piece of equipment is directly associated
345 with that unique identity and that association does not need to be repeated for every piece
346 of data reported. A client software application may then directly relate the information
347 provided in the MTConnectDevices document with the data provided in the MTCon-
348 nectStreams document based on this identity.

349 *ComponentStream* is the next level XML element in the MTConnectStreams docu-
350 ment. *ComponentStream* is also a container type XML element. There **MUST** be
351 a separate *ComponentStream* XML element for each of the *Structural Elements* (*De-*
352 *vice elements, Top Level Component elements, or Lower Level Component elements*)
353 defined for that piece of equipment in the associated MTConnectDevices XML docu-
354 ment. A *ComponentStream* representing a *Structural Element* will only appear if there
355 is data reported for that *Structural Element*. (Note: See *MTConnect Standard: Part 2.0 -*
356 *Devices Information Model* of the MTConnect Standard for a description of the *Structural*
357 *Elements* for a piece of equipment).

358 There are three (3) *Structural Elements* – *Samples*, *Events*, and *Condition* at the
359 next level of the MTConnectStreams document. Each one of these *Structural Elements*
360 is a container type XML element. These *Structural Elements* group the data reported for
361 each component of a piece of equipment according to the *Data Entity* categories defined

362 in *MTConnect Standard: Part 2.0 - Devices Information Model*, Sections 7 and 8.

- 363 • **Samples** contains **SAMPLE** category *Data Entities* defined in the *MTConnect-*
364 *Devices XML* document (See *MTConnect Standard: Part 2.0 - Devices Informa-*
365 *tion Model*, Section 8.1)
- 366 • **Events** contains **EVENT** category *Data Entities* defined in the *MTConnectDe-*
367 *VICES XML* document (See *MTConnect Standard: Part 2.0 - Devices Informa-*
368 *tion Model*, Section 8.2)
- 369 • **Condition** contains **CONDITION** category *Data Entities* defined in the *MTCon-*
370 *nectDevices XML* document (See *MTConnect Standard: Part 2.0 - Devices*
371 *Information Model*, Section 8.3)

372 There **MUST** be at least one of **Samples**, **Events**, or **Condition** elements in each
373 **ComponentStream** container.

374 *Figure 1* XML tree structure illustrates the various *Structural Elements* used to organize
375 the data reported by a piece of equipment and the relationship between these elements.

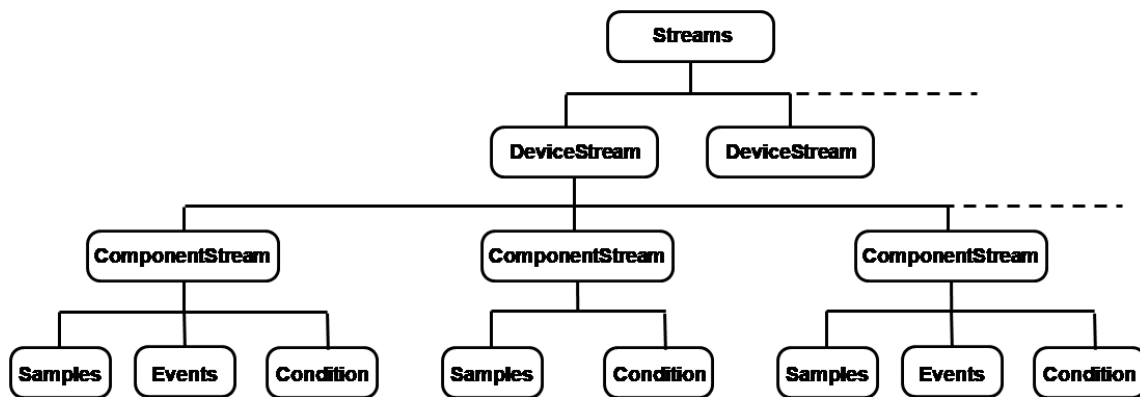


Figure 1: Streams Data Structure

376 *Example 1* is a sample from an *MTConnectStreams XML* document that contains the
377 response from an *Agent* representing two pieces of equipment, *mill-1* and *mill-2*. The data
378 from each piece of equipment is reported in a separate *DeviceStream* container.

Example 1: Example of DeviceStream

```

379 1 <MTConnectStreams ...>
380 2   <Header ... />
381 3   <Streams>
382 4     <DeviceStream name="mill-1" uuid="1">
383 5       <ComponentStream component="Device" name="mill-1">

```

```

384 6         componentId="d1">
385 7     <Events>
386 8         <Availability dataItemId="avail1" name="avail"
387 9             sequence="5"
388 10             timestamp="2010-04-06T06:19:35.153141">
389 11             AVAILABLE</Availability>
390 12     </Events>
391 13 </ComponentStream>
392 14 </DeviceStream>
393 15 <DeviceStream name="mill-2" uuid="2">
394 16     <ComponentStream component="Device" name="mill-2"
395 17         componentId="d2">
396 18         <Events>
397 19             <Availability dataItemId="avail2" name="avail"
398 20                 sequence="15"
399 21                 timestamp="2010-04-06T06:19:35.153141">
400 22                 AVAILABLE</Availability>
401 23             </Events>
402 24         </ComponentStream>
403 25     </DeviceStream>
404 26 </Streams>
405 27 </MTConnectStreams>

```

406 In *Example 1*, it should be noted that the *sequence numbers* are unique across the two
 407 pieces of equipment. Client software applications **MUST NOT** assume that the `Events`
 408 and `Samples` sequence numbers are strictly in sequence. All sequence numbers **MAY**
 409 **NOT** be included. For instance, such a case would occur when HTTP filtering is applied to
 410 the request and the `SAMPLE`, `EVENT`, and `CONDITION` data types for other components
 411 are not returned. Another case would occur when an *Agent* is supporting more than one
 412 piece of equipment and data from only one piece of equipment is requested. Refer to MT-
 413 Connect Standard *MTConnect Standard Part 1.0 - Overview and Fundamentals, Section 5*
 414 for more information on *sequence numbers*.

415 4.1 Streams

416 `Streams` is a container type XML element that **MUST** contain only `DeviceStream`
 417 elements. `Streams` **MAY** contain any number of `DeviceStream` elements. If there is
 418 no data to be reported for a request for data, an `MTConnectStreams` document **MUST**
 419 be returned with an empty `Streams` container. *Data Entities* **MAY NOT** be directly
 420 associated with the `Streams` container.

421 The XML schema in *Figure 2* represents the structure of the `Streams` XML element.

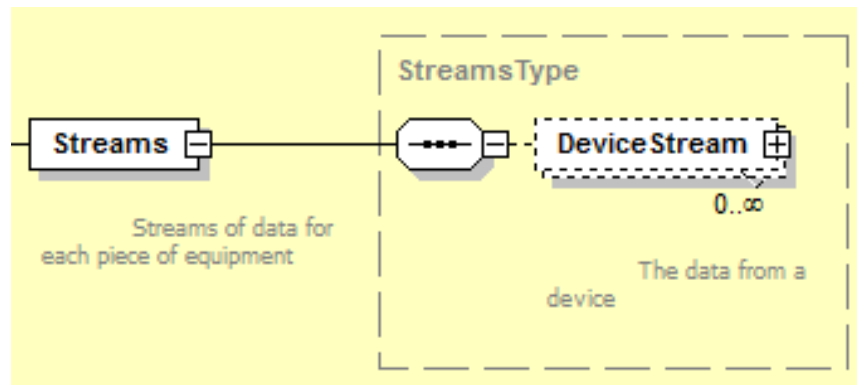


Figure 2: Streams Schema Diagram

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

422 4.2 DeviceStream

423 DeviceStream is a XML container that organizes data reported from a single piece of
 424 equipment. A DeviceStream element **MUST** be provided for each piece of equipment
 425 reporting data in an MTConnectStreams document.

426 A DeviceStream **MAY** contain any number of ComponentStream elements; lim-
 427 ited to one for each component element represented in the MTConnectDevices doc-
 428 ument. If the response to the request for data from an *Agent* does not contain any data
 429 for a specific piece of equipment, an empty DeviceStream element **MAY** be created to
 430 indicate that the piece of equipment exists, but there was no data available. In this case,
 431 there will be no ComponentStream elements provided.

Table 2: MTConnect DeviceStream Element

Element	Description	Occurrence
DeviceStream	<p>An XML container element provided in the Streams container in the MTConnectStreams document.</p> <p>There MAY be one or more DeviceStream elements in a Streams container; one for each piece of equipment represented in the MTConnectStreams document.</p>	0..*

432 4.2.1 XML Schema for DeviceStream

433 The XML schema in *Figure 3* represents the structure of the DeviceStream XML
 434 element showing the attributes defined for DeviceStream and the elements that **MAY**
 435 be associated with DeviceStream.

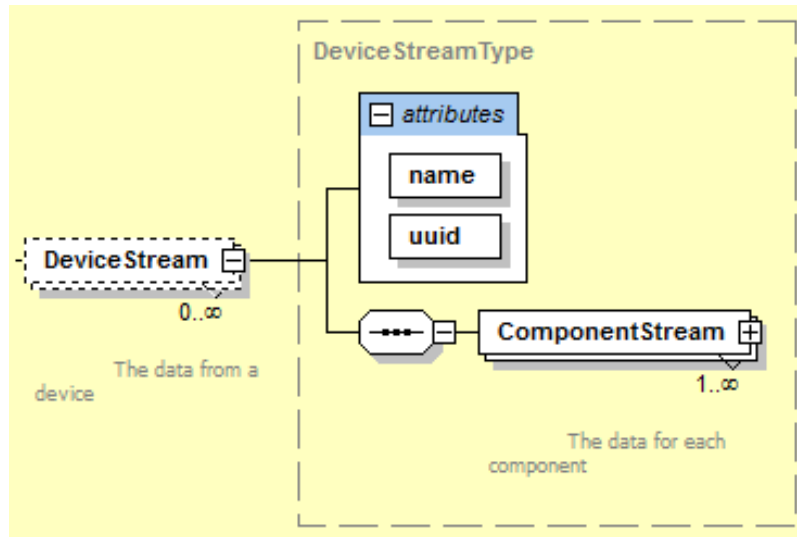


Figure 3: DeviceStream Schema Diagram

4.2.2 Attributes for DeviceStream

Table 3 defines the attributes that **MUST** be provided to uniquely identify each specific 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 <code>DeviceStream</code> 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 <code>MTConnectDevices</code> 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 uuid associated with the piece of equipment reporting the data contained in this DeviceStream container.</p> <p>uuid is a required attribute.</p> <p>The value reported for uuid MUST be the same as the value defined for the uuid attribute of the same piece of equipment in the MTConnectDevices document.</p>	1

439 4.2.3 Elements for DeviceStream

440 Table 4 lists the XML element(s) that **MAY** be provided in the DeviceStream XML
 441 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 ComponentStream elements MAY be provided in a DeviceStream container.</p> <p>There MUST be a separate ComponentStream XML element for each of the <i>Structural Elements</i> (Device elements, <i>Top Level</i> Component elements, or <i>Lower Level</i> Component elements) defined for that piece of equipment in the associated MTConnectDevices XML document. A ComponentStream representing a <i>Structural Element</i> will only appear if there is data reported for that <i>Structural Element</i>.</p>	1..*

442 4.3 ComponentStream

443 ComponentStream is a XML container that organizes the data associated with each
444 *Structural Element* (Device element, *Top Level* Component, or *Lower Level* Com-
445 ponent element) defined for that piece of equipment in the associated MTConnectDe-
446 vices XML document. The data reported in each ComponentStream element **MUST**
447 be grouped into individual XML containers based on the value of the category attribute
448 (SAMPLE, EVENT, or CONDITION) defined for each *Data Entity* in the MTConnect-
449 Devices XML document. These containers are Samples, Events, and Condition.

450 4.3.1 XML Schema for ComponentStream

451 The XML schema in *Figure 4* represents the structure of a ComponentStream XML
452 element showing the attributes defined for ComponentStream and the elements that
453 **MAY** be associated with ComponentStream.

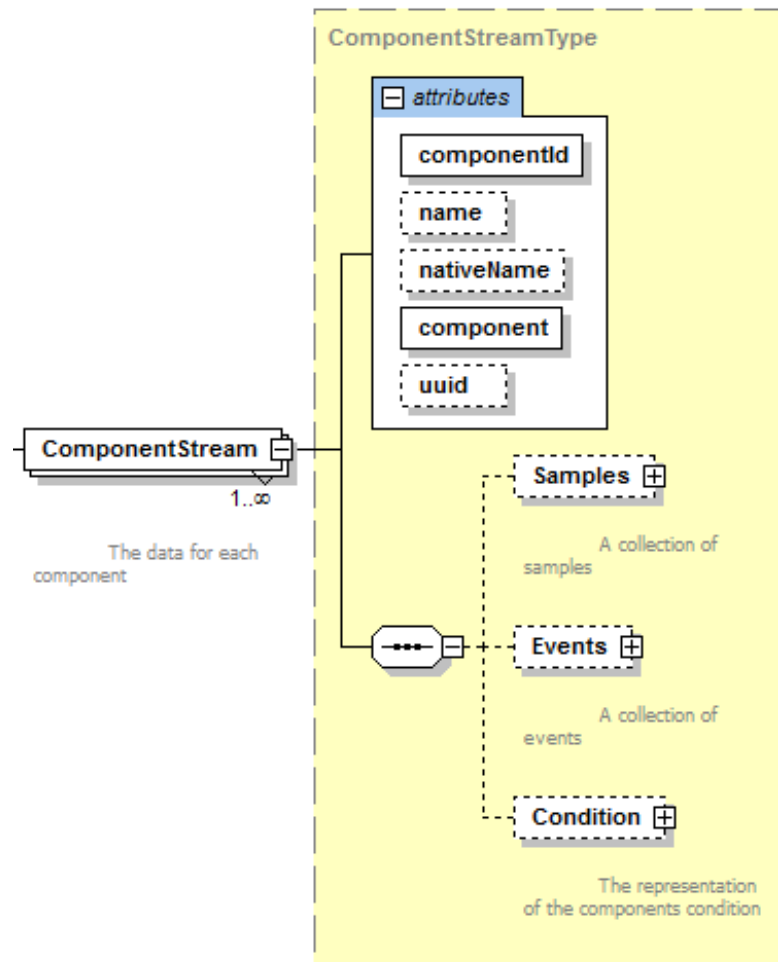


Figure 4: ComponentStream Schema Diagram

454 ComponentStream is similar to DeviceStream in that the attributes uniquely identify the *Structural Element* with which the data reported is directly associated. This information does not have to be repeated for each *Data Entity*. In the case of the DeviceStream, the attributes uniquely identify the piece of equipment associated with the data. In the case of the ComponentStream, the attributes identify the specific *Structural Element* within a piece of equipment associated with each *Data Entity*.

460 4.3.2 Attributes for ComponentStream

461 The Table 5 defines the attributes used to uniquely identify the specific *Structural Element(s)* of a piece of equipment associated with the data reported in the MTConnect-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</i> Component element, or <i>Lower Level</i> Component element) as defined by the <code>id</code> 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</i> Component element, or <i>Lower Level</i> Component 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> (Device, <i>Top Level</i> Component, or <i>Lower Level</i> Component) associated with the ComponentStream 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> (Device element, <i>Top Level</i> Component element, or <i>Lower Level</i> Component element) defined in the MTConnectDevices XML document.</p> <p>Examples of Component are Device, Axes, Controller, Linear, Electric and Loader.</p>	1
uuid	<p>uuid of the ComponentStream element.</p> <p>uuid is an optional attribute.</p> <p>If uuid 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 uuid 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, but it is not required.</p> <p>If provided, the value reported for uuid MUST be the same as the value defined for the uuid 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

4.3.3 Elements for ComponentStream

In the `ComponentStream` container, an *Agent* **MUST** organize the data reported in each `ComponentStream` into individual `Samples`, `Events`, or `Condition` XML containers based on the value of the `category` attribute (i.e., `SAMPLE`, `EVENT`, or `CONDITION`) defined for each *Data Entity* defined in the `MTConnectDevices` XML document.

Each `ComponentStream` element **MUST** include at least one `Events`, `Samples`, or `Condition` XML container element. *Data Entities* returned in each of the `ComponentStream` container elements are defined in the *Table 6*.

Table 6: Elements for ComponentStream

Element	Description	Occurrence
<code>Samples</code>	An XML container type element. <code>Samples</code> organizes the <code>SAMPLE</code> type <i>Data Entities</i> defined in the <code>MTConnectDevices</code> document that are reported in each <code>ComponentStream</code> XML element.	0..1 [†]
<code>Events</code>	An XML container type element. <code>Events</code> organizes the <code>EVENT</code> type <i>Data Entities</i> defined in the <code>MTConnectDevices</code> document that are reported in each <code>ComponentStream</code> XML element.	0..1 [†]
<code>Condition</code>	An XML container type element. <code>Condition</code> organizes the <code>CONDITION</code> type <i>Data Entities</i> defined in the <code>MTConnectDevices</code> document that are reported in each <code>ComponentStream</code> XML element.	0..1 [†]

Note: [†]The `ComponentStream` element **MUST** contain at least one of these element types.

475 5 Data Entities

476 When a piece of equipment reports values associated with `DataItem` elements defined
 477 in the `MTConnectDevices` document, that information is organized as *Data Entities*
 478 in the `MTConnectStreams` document. These *Data Entities* are organized in containers
 479 within each `ComponentStream` element based on the category attribute defined for
 480 the corresponding `DataItem` in the `MTConnectDevices` document:

481 `DataItem` elements defined with a category attribute of `SAMPLE` in the `MTCon-`
 482 `nectDevices` document are mapped to the `Samples` XML container in the associated
 483 `ComponentStream` element.

484 `DataItem` elements defined with a category attribute of `EVENT` in the `MTCon-`
 485 `nectDevices` document are mapped to the `Events` XML container in the associated
 486 `ComponentStream` element.

487 `DataItem` elements defined with a category attribute of `CONDITION` in the `MT-`
 488 `ConnectDevices` document are mapped to the `Condition` XML container in the
 489 associated `ComponentStream` element.

490 The XML tree in *Figure 5* demonstrates how *Data Entities* are organized in these contain-
 491 ers.

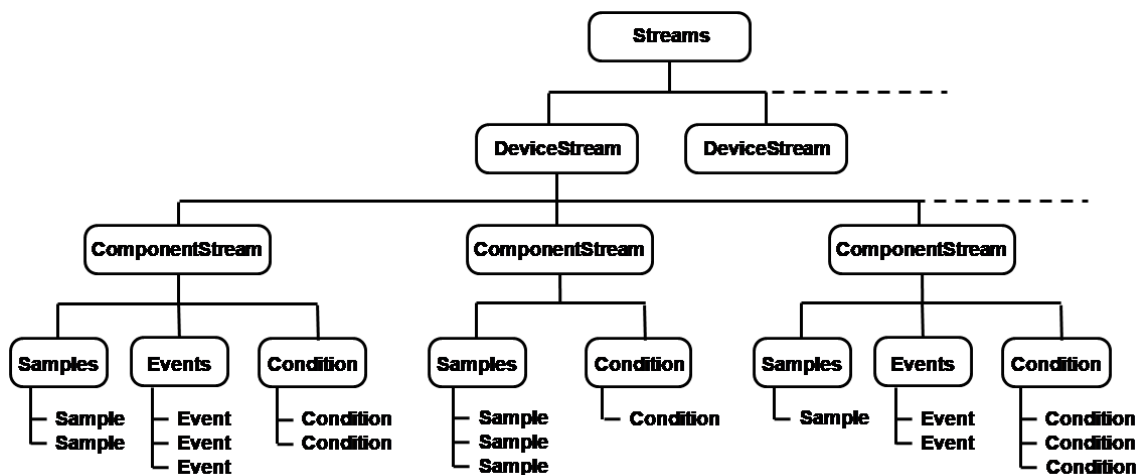


Figure 5: ComponentStream XML Tree Diagram

492 *Example 2* is an illustration of the structure of an XML document demonstrating how *Data*
 493 *Entities* are reported in a `MTConnectStreams` document:

Example 2: Example of MTConnectStreams

```

494 1 <MTConnectStreams>
495 2   <Header/>
496 3   <Streams>
497 4     <DeviceStream>
498 5       <ComponentStream>
499 6         <Samples>
500 7           <Sample/>
501 8           <Sample/>
502 9         </Samples>
503 10        <Events>
504 11          <Event/>
505 12          <Event/>
506 13        </Events>
507 14        <Condition>
508 15          <Condition/>
509 16          <Condition/>
510 17        </Condition>
511 18      </ComponentStream>
512 19      <ComponentStream>
513 20        <Samples>
514 21          <Sample/>
515 22          <Sample/>
516 23        </Samples>
517 24        <Events>
518 25          <Event/>
519 26          <Event/>
520 27        </Events>
521 28        <Condition>
522 29          <Condition/>
523 30          <Condition/>
524 31        </Condition>
525 32      </ComponentStream>
526 33    </DeviceStream>
527 34  </Streams>
528 35 </MTConnectStreams>

```

529 **Note:** There are no specific requirements defining the sequence in which the Com-
530 ponentStream XML elements are organized in the MTConnectStreams
531 document. They **MAY** be organized in any sequence based on the implemen-
532 tation of an *Agent*. The sequence in which the ComponentStream XML
533 elements appear does not impact the ability for a client software application to
534 interpret the information that it receives in the document.

535 When an *Agent* responds to a current HTTP request, the information returned in the
536 MTConnectStreams document **MUST** include the most current value for every *Data*
537 *Entity* defined in the MTConnectDevices document subject to any filtering included
538 within the request.

When an *Agent* responds to a sample HTTP request, the information returned in the MTConnectStreams document **MUST** include the occurrences for each *Data Entity* that are available to an *Agent* subject to filtering and the count parameter included within the request (see *MTConnect Standard Part 1.0 - Overview and Fundamentals* for a full definition of the protocol).

5.1 Element Names for Data Entities

In the MTConnectDevices document, *Data Entities* are grouped as *DataItem* XML elements within each *Device*, *Top Level Component*, and *Lower Level Component Structural Element*. The *Data Entities* reported in the MTConnectStreams document associated with each of these *Structural Elements* are represented with an *Element Name* based on the category and type defined for each of the *DataItem* elements in the MTConnectDevices document.

5.1.1 Element Names when MTConnectDevices category is SAMPLE or EVENT

The *Data Entities* reported in the MTConnectStreams document associated with each *DataItem* element defined in the MTConnectDevices document with a category attribute of SAMPLE or EVENT **MUST** be identified in the MTConnectStreams document with an *Element Name* derived from the type attribute defined for that *DataItem* element in the MTConnectDevices document.

The element name **MUST** derive from the *DataItem* type converted to *Pascal-Case* by removing underscores (`_`) and capitalizing each word. The conversion **MUST NOT** apply to the following abbreviated words: PH, AC, DC and URI. MTCONNECT **MUST** be converted to MTConnect.

Example 3 describes the most common method used to derive the *Element Name* for a *Data Entity* reported in the MTConnectStreams document from the information describing that *DataItem* element in the MTConnectDevices document:

DataItem Represented in the MTConnectDevices Document

Example 3: DataItem Represented in MTConnectDevices Document

```
1 <DataItem type="AXIS_FEEDRATE" id="xf" name="Xfrt"
2   category="SAMPLE" units="MILLIMETER/SECOND"
3   nativeUnits="MILLIMETER/SECOND"/>
```

- `DataItem`: The XML *Element Name* for this *Data Entity*.

Note: *Element Name* must not be confused with the name attribute for the data item element.

- `type`, `category`, `units`, and `nativeUnits`: Attributes that provide additional information regarding each data item in the `MTConnectDevices` document.

Response Format reported in the `MTConnectStreams` Document

Example 4: Response Format reported in the `MTConnectStreams` Document

```
1 <AxisFeedrate name="Xfirt" sequence="61315517"
2   timestamp="2016-07-28T02:06:01.364428Z"
3   dataItemId="xf">10.83333</AxisFeedrate>
```

- `AXIS_FEEDRATE`: The *Element Name* provided in the `MTConnectStreams` response format for the data item. The *Element Name* for a data item is defined by the `type` attribute of `AXIS_FEEDRATE` in the `MTConnectDevices` document. The *Element Name* **MUST** be provided in Pascal case format (first letter of each word is capitalized).

5.1.2 Changes to Element Names when representation attribute is used

The *Element Name* for a *Data Entity* reported in the `MTConnectStreams` document is extended when the `representation` attribute is used to further describe that `DataItem` element in the `MTConnectDevices` document.

5.1.3 Element Names when `MTConnectDevices` category is CONDITION

Data Entities defined in the `MTConnectDevices` document with a `category` attribute of `CONDITION` are reported with an *Element Name* that is defined differently from other *Data Entity* types. The *Element Name* for these *Data Entities* are defined based on the *Fault State* (Normal, Warning, or Fault) associated with each *Data Entity* at the time that a value for that *Data Entity* is reported. See *Section 5.8.1 - Element Names for Condition* and *Section 5.9 - Unavailability of Fault State for Condition* for details on how these *Data Entities* are reported in the `MTConnectStreams` document.

598 5.2 Samples Container

599 `Samples` is a XML container type element. `Samples` organizes the *Data Entities* re-
 600 turned in the `MTConnectStreams` XML document for those `DataItem` elements de-
 601 fined with a `category` attribute of `SAMPLE` in the `MTConnectDevices` document.

602 A separate `Samples` container will be provided for the data returned for the `DataItem`
 603 elements associated with each *Structural Element* of a piece of equipment defined in the
 604 `MTConnectDevices` document.

Table 7: MTConnect Samples Element

Element	Description	Occurrence
<code>Samples</code>	<p>An XML container type element that organizes the data reported in the <code>MTConnectStreams</code> document for <code>DataItem</code> elements defined in the <code>MTConnectDevices</code> document with a <code>category</code> attribute of <code>SAMPLE</code>.</p> <p>A separate <code>Samples</code> container MUST be provided for each <code>ComponentStream</code> element for which data is returned for a <code>DataItem</code> element defined in the <code>MTConnectDevices</code> document with a <code>category</code> attribute of <code>SAMPLE</code>.</p> <p>If provided in the document, a <code>Samples</code> XML container MUST contain at least one <code>Sample</code> element.</p>	0..1

605 5.3 Sample Data Entities

606 A `Sample` XML element provides the information and data reported from a piece of
 607 equipment for those `DataItem` elements defined with a `category` attribute of `SAMPLE`
 608 in the `MTConnectDevices` document.

609 `Sample` is an abstract type XML element and will never appear directly in the `MTCon-`
 610 `nectStreams` XML document. As an abstract type XML element, `Sample` will be
 611 replaced in the XML document by a specific type of `Sample` specified by the *Element*
 612 *Name* for that *Data Entity*. The different types of `Sample` elements are defined in
 613 *Section 6.1 - Sample Element Names*. Examples of XML elements representing `Sample`
 614 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 <code>DataItem</code> elements defined with a <code>category</code> attribute of <code>SAMPLE</code> in the <code>MTConnectDevices</code> document.</p> <p><code>Sample</code> is an abstract type XML element. It is replaced in the <code>MTConnectStreams</code> document by a specific type of <code>Sample</code> element.</p> <p>There MAY be multiple types of <code>Sample</code> elements in a <code>Samples</code> container.</p>	1..*

615 5.3.1 XML Schema Structure for Sample

616 The XML schema in *Figure 6* represents the structure of a `Sample` XML element show-
617 ing the attributes defined for `Sample` elements.

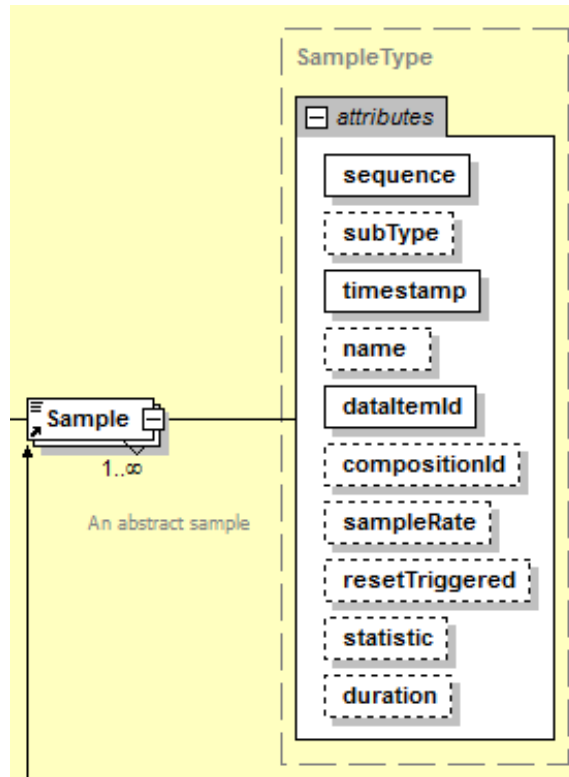


Figure 6: Sample Schema Diagram

5.3.2 Attributes for Sample

The *Table 9* defines the attributes used to provide additional information for a `Sample` XML element.

Table 9: Attributes for Sample

Attribute	Description	Occurrence
sequence	<p>A number representing the sequential position of an occurrence of the <code>Sample</code> 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

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 NMTOKEN 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. 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 <code>theStatistic</code> attribute of the <code>DataItem</code> element is defined in the <code>MTConnectDevices</code> document that this <code>Sample</code> element represents.</p>	0..1
resetTriggered	<p>For those <code>DataItem</code> elements that report data that may be periodically reset to an initial value, <code>resetTriggered</code> identifies when a reported value has been reset and what has caused that reset to occur.</p> <p><code>resetTriggered</code> is an optional attribute.</p> <p><code>resetTriggered</code> MUST only be provided for the specific occurrence of a <i>Data Entity</i> reported in the <code>MTConnectStreams</code> document when the reset occurred and MUST NOT be provided for any other occurrence of the <i>Data Entity</i> reported in a <code>MTConnectStreams</code> document.</p>	0..1
compositionId	<p>The identifier of the <code>Composition</code> element defined in the <code>MTConnectDevices</code> document associated with the data reported for the <code>Sample</code> element.</p> <p><code>compositionId</code> is an optional attribute.</p>	0..1

5.3.2.1 duration Attribute for Sample

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 the Sample **MUST** reference the time the data collection was completed. timestamp **MUST NOT** represent any other time associated with the data collection or the calculation of the statistic. The actual time the interval began can be computed by subtracting the duration from the timestamp.

Two Sample elements **MAY** have overlapping time periods when statistics are computed at different frequencies. For example, there may be two *Data Entities* reporting a statistic representing the average value for the readings of the same measured signal calculated over one and five minute intervals. These *Data Entities* can both have the same start time for their calculations (e.g., 05:10:00), but the timestamp and duration will be 05:11:00 and 60 seconds, respectively, for the *Data Entity* reporting the one-minute average and 05:15:00 and 300 seconds, respectively, for the *Data Entity* reporting the five-minute average. This allows for varying statistical methods to be applied with different interval lengths each having different values for the timestamp and duration attributes.

5.3.2.2 resetTriggered Attribute for Sample

Some *Data Entities* **MAY** have their reported value reset to an initial value. These reset actions may be based upon a specific elapsed time or may be triggered by a physical or logical reset action that causes the reset to occur. Examples of *Data Entities* that **MAY** have their reported value reset to an initial value are *Data Entities* representing a counter, a timer, or a statistic.

resetTriggered defines the type of reset action that caused the value of the reported data to be reset. The value reported for resetTriggered **MAY** be defined by the ResetTrigger element for the *Data Entity* in the MTConnectDevices document that this Sample element represents. If the ResetTrigger element is not defined in the MTConnectDevices document, a resetTriggered attribute **SHOULD** be reported in the MTConnectStreams document if the type of reset action can be determined and reported by the piece of equipment.

resetTriggered **MUST** only be reported for the first occurrence of a *Data Entity* after a reset action has occurred and **MUST NOT** be provided for any other occurrence of the *Data Entity* reported in a MTConnectStreams document. When a reset occurs, the piece of equipment **MUST** report an occurrence of the *Data Entity* that was reset even if that occurrence of the *Data Entity* would normally be suppressed based on the filtering criteria established in the MTConnectDevices document that this Sample element represents.

657 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.

658 5.3.3 Valid Data Values for Sample

659 All `Sample` elements reported in an `MTConnectStreams` XML document **MUST** pro-
 660 vide a value in the CDATA of the *Data Entity*.

661 The value returned in the CDATA **MUST** be reported as either a *Valid Data Value* rep-
 662 resenting the information reported from a piece of equipment or `UNAVAILABLE` when a
 663 *Valid Data Value* cannot be determined.

664 The *Valid Data Value* reported for a `Sample` represents the reading of the value of a
 665 continuously variable or analog data source.

666 The `representation` attribute for a `SAMPLE` category `DataItem` element defined
 667 in the `MTConnectDevices` document specifies how an *Agent* **MUST** record instances
 668 of the data associated with that data item and how often that data **MUST** be reported as a
 669 `Sample` element in the `MTConnectStreams` document.

670 The data reported for a `Sample` element associated with a `SAMPLE` category `DataItem`
 671 element with a `representation` of `VALUE` can be measured at any point-in-time and
 672 **MUST** always produce a result with a single data value.

673 Note: If a `representation` attribute is not specified in the `MTConnectDe-`
 674 `vices` document for a `DataItem` element, it **MUST** be assumed that the
 675 data reported in the `MTConnectStreams` document for the *Data Entity* has
 676 a `representation` type of `VALUE`.

677 In the case of a `Sample` element associated with a `SAMPLE` category `DataItem` element
 678 with a `representation` attribute of `TIME_SERIES`, the data provided **MUST** be a
 679 series of data values representing multiple sequential samples of the measured value that
 680 will be provided only at the end of the completion of a sampling period. (See Section
 681 *Section 5.6.1 - Observations for DataItem with representation of TIME_SERIES* for more
 682 information on `TIME_SERIES` type data).

683 In the case of a `Sample` element associated with a `SAMPLE` category `DataItem` element
 684 with a `representation` attribute of `DATA_SET`, the data reported for each *key-value*
 685 *pair* **MUST** be provided in the same *Valid Data Values* and units as specified by the `type`
 686 attribute for the `DataItem` element.

687 When an *Agent* responds to a *Current Request*, the information returned in the `MTCon-`
 688 `nectStreams` document for a *Data Entity* defined to represent a *Data Set* **MUST** in-
 689 clude the full set of *key-value pairs* that are valid for that *Data Entity*. If the *Current*
 690 *Request* includes an `at` query parameter, the *Agent* **MUST** provide the set of *key-value*
 691 *pairs* that are valid at the specified *sequence number*.

692 When an *Agent* responds to a *Sample Request*, the information returned in the `MTCon-`
 693 `nectStreams` document for a *Data Entity* defined to represent a *Data Set* **MUST** in-
 694 clude only those *key-value pairs* that are valid for the *Data Entity* at each *sequence number*.

695 Data values provided for a `Sample` **MUST** always be a floating-point number. In the
 696 `MTConnect` Standard, floating-point numbers are defined as XML `xs:float` type numbers
 697 as defined by W3C. Any of the following number formats are valid XML floating type
 698 numbers: 1267.43233E12, -1E4, 12.78e-2, 12, 137.2847, 0, and INF.

699 Note: For some `Sample` elements, the *Valid Data Value* **MAY** be restricted to spe-
 700 cific formats. See Section 6.1 of this document for a description of any restric-
 701 tions of the acceptable format for *Valid Data Value*.

For *Sample* elements, a client software application can determine the appropriate accuracy of the value reported for the *Data Entity* by applying the `significantDigits` attribute defined for the corresponding *DataItem* element defined in the `MTConnectDevices` document.

The *Valid Data Value* reported as CDATA for a *Sample* element **MUST** be formatted as 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 5*.

Example 5: Example showing CDATA of a *DataItem* Element

```
<Position sequence="112" name="Xabs"
timestamp="2016-07-28T02:06:01.364428Z"
dataItemId="10">123.3333</Position>
```

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.

5.3.4 Unavailability of Valid Data Values for Sample

If an *Agent* cannot determine a *Valid Data Value* for a *Sample* element, the value returned for the CDATA for the *Data Entity* **MUST** be reported as UNAVAILABLE.

Example 6 demonstrates how an *Agent* reports the value for a *Sample* in the CDATA when it is unable to determine a *Valid Data Value*:

Example 6: Example of CDATA when Data Entity is UNAVAILABLE

```
<Samples>
  <PathPosition dataItemId="p2"
    timestamp="2009-03-04T19:45:50.458305"
    subType="ACTUAL" name="Zact"
    sequence="15065113">UNAVAILABLE</PathPosition>
  <Temperature dataItemId="t6"
    timestamp="2009-03-04T19:45:50.458305" name="temp"
    sequence="150651134">UNAVAILABLE</Temperature>
</Samples>
```

5.4 Events Container

Events is a XML container type element. *Events* organizes the *Data Entities* returned in the `MTConnectStreams` XML document for those *DataItem* elements defined with a `category` attribute of `EVENT` in the `MTConnectDevices` document.

733 A separate `Events` container will be provided for the data returned for the `DataItem`
 734 elements associated with each *Structural Element* of a piece of equipment defined in the
 735 `MTConnectDevices` document.

Table 11: MTConnect Event Element

Element	Description	Occurrence
Events	<p>An XML container type element that organizes the data reported in the <code>MTConnectStreams</code> document for <code>DataItem</code> elements defined in the <code>MTConnectDevices</code> document with a <code>category</code> attribute of <code>EVENT</code>.</p> <p>A separate <code>Events</code> container MUST be provided for each <code>ComponentStream</code> element for which data is returned for a <code>DataItem</code> element defined in the <code>MTConnectDevices</code> document with a <code>category</code> attribute of <code>EVENT</code>.</p> <p>If provided in the document, an <code>Events</code> XML container MUST contain at least one <code>Event</code> element.</p>	0..1

736 5.5 Event Data Entities

737 An `Event` XML element provides the information and data provided from a piece of
 738 equipment for those `DataItem` elements defined with a `category` attribute of `EVENT`
 739 in the `MTConnectDevices` document.

740 `Event` is an abstract type XML element and will never appear directly in the `MTCon-`
 741 `nectStreams` XML document. As an abstract type XML element, `Event` will be
 742 replaced in the XML document by a specific type of `Event` specified by the *Element*
 743 *Name* for that *Data Entity*. The different types of `Event` elements are defined in *Sec-*
 744 *tion 6.2 - Event Element Names*. Examples of XML elements representing `Event` include
 745 `Block` and `Execution`.

746 `Event` is similar to `Sample`, but its value can change with unpredictable frequency.
 747 `Events` do not report intermediate values. As an example, when `Availability` tran-
 748 sitions from `UNAVAILABLE` to `AVAILABLE`, there is no intermediate state that can be
 749 inferred.

750 `Event` elements **MAY** report data values defined by a controlled vocabulary as speci-

751 fied in *Section 6.2 - Event Element Names*, by numeric values, or by a character string
 752 representing text or a message provided by the piece of equipment.

Table 12: MTConnect Event Element

Element	Description	Occurrence
Event	<p>An XML element which provides the information and data reported from a piece of equipment for those <code>DataItem</code> elements defined with a <code>category</code> attribute of <code>EVENT</code> in the <code>MTConnectDevices</code> document.</p> <p>Event is an abstract type XML element. It is replaced in the <code>MTConnectStreams</code> document by a specific type of Event element.</p> <p>There MAY be multiple types of Event elements in a <code>Events</code> container.</p>	1..*

753 5.5.1 XML Schema Structure for Event

754 The XML schema in *Figure 7* represents the structure of an Event XML element show-
 755 ing the attributes defined for Event elements.

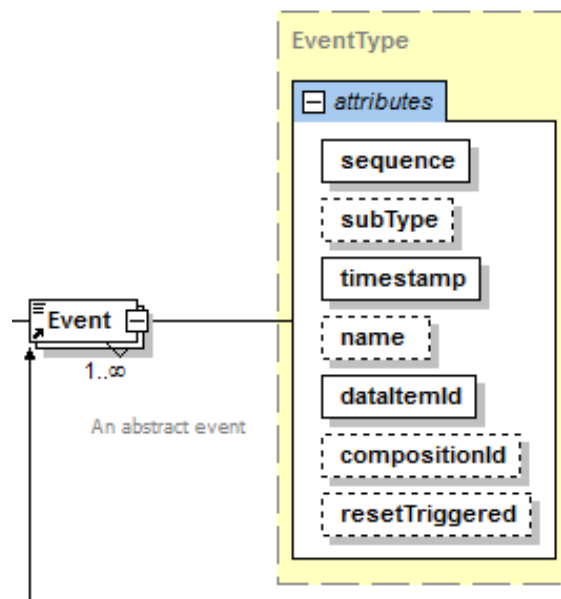


Figure 7: Event Schema Diagram

756 5.5.2 Attributes for Event

757 *Table 13* defines the attributes that **MAY** be used to provide additional information for an
 758 Event XML element.

Table 13: Attributes for Event

Attribute	Description	Occurrence
sequence	<p>A number representing the sequential position of an occurrence of the <code>Event</code> 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 <code>DataItem</code> element as defined in the <code>MTConnectDevices</code> document that the <code>Event</code> 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 <code>Event</code> was measured.</p> <p>timestamp is a required attribute.</p>	1
name	<p>The name of the <code>Event</code> element.</p> <p>name is an optional attribute.</p> <p>name MUST match the name attribute of the <code>DataItem</code> element defined in the <code>MTConnectDevices</code> document that the <code>Event</code> element represents.</p> <p>An NMTOKEN XML type.</p>	0..1

Continuation of Table 13		
Attribute	Description	Occurrence
dataItemId	<p>The unique identifier for the <code>Event</code> element.</p> <p><code>dataItemId</code> is a required attribute.</p> <p><code>dataItemId</code> MUST match the <code>id</code> attribute of the <code>DataItem</code> element defined in the <code>MTConnectDevices</code> document that the <code>Event</code> element represents.</p>	1
resetTriggered	<p>For those <code>DataItem</code> elements that report data that may be periodically reset to an initial value, <code>resetTriggered</code> identifies when a reported value has been reset and what has caused that reset to occur.</p> <p><code>resetTriggered</code> is an optional attribute.</p> <p><code>resetTriggered</code> MUST only be provided for the specific occurrence of a <i>Data Entity</i> reported in the <code>MTConnectStreams</code> document when the reset occurred and MUST NOT be provided for any other occurrence of the <i>Data Entity</i> reported in a <code>MTConnectStreams</code> document.</p>	0..1
compositionId	<p>The identifier of the <code>Composition</code> element defined in the <code>MTConnectDevices</code> document associated with the data reported for the <code>Event</code> element.</p> <p><code>compositionId</code> is an optional attribute.</p>	0..1

759 5.5.3 Valid Data Values for Event

760 `Event` elements reported in an `MTConnectStreams` XML document **MUST** provide
761 a value in the CDATA of the *Data Entity*.

762 The value reported in the CDATA **MUST** be reported as either a *Valid Data Value* rep-
763 resenting the information reported from a piece of equipment or UNAVAILABLE when a
764 *Valid Data Value* cannot be determined.

765 The *Valid Data Value* reported for an `Event` represents a distinct piece of information
 766 provided from a piece of equipment. Unlike `Sample`, `Event` does not report intermediate
 767 values that vary over time. `Event` reports information that, when provided at any specific
 768 point in time, represents the current state of the piece of equipment.

769 The `representation` attribute for an `EVENT` category data item defined in the `MT-`
 770 `ConnectDevices` document specifies how an *Agent* **MUST** record instances of data
 771 associated with that data item and how that data **MUST** be reported as an `Event` element
 772 in the `MTConnectStreams` document.

773 The data reported for an `Event` element associated with an `EVENT` category data item
 774 with a `representation` attribute of `VALUE` **MUST** be either an integer, a floating-
 775 point number, a descriptive value (text string) representing one of two or more state values
 776 defined for that data item, or a text string representing a message.

777 If a `representation` attribute is not specified for a data item in an `MTConnectDe-`
 778 `vices` document, the designation for the `representation` attribute **MUST** be inter-
 779 preted as `VALUE`.

780 In the case of an `Event` element associated with a `EVENT` category `DataItem` element
 781 with a `representation` attribute of `DATA_SET`, the data reported for each *key-value*
 782 *pair* **MUST** be provided in the same *Valid Data Values* and units as specified by the `type`
 783 attribute for the `DataItem` element.

784 When an *Agent* responds to a *Current Request*, the information returned in the `MTCon-`
 785 `nectStreams` document for a *Data Entity* defined to represent a *Data Set* **MUST** in-
 786 clude the full set of *key-value pairs* that are valid for that *Data Entity*. If the *Current*
 787 *Request* includes an `at` query parameter, the *Agent* **MUST** provide the set of *key-value*
 788 *pairs* that are valid at the specified *sequence number*.

789 When an *Agent* responds to a *Sample Request*, the information returned in the `MTCon-`
 790 `nectStreams` document for a *Data Entity* defined to represent a *Data Set* **MUST** in-
 791 clude only those *key-value pairs* that are valid for the *Data Entity* at each *sequence number*
 792 The *Valid Data Value* reported as `CDATA` for an `Event` element **MUST** be formatted as
 793 part of the content between the element tags in the XML element representing that *Data*
 794 *Entity*. As an example, `Event` elements are formatted as shown in *Example 7*:

Example 7: Example of Event Element

```

795 1 <PartCount dataItemId="pc4"
796 2     timestamp="2009-02-26T02:02:36.48303"
797 3     name="pcount" sequence="185">238</PartCount>
798 4 <ControllerMode dataItemId="p3"
799 5     timestamp="2009-02-26T02:02:35.716224"
800 6     name="mode" sequence="192">AUTOMATIC</ControllerMode>
801 7 <Block dataItemId="cn2" name="block" sequence="206">
```

802 8 timestamp="2009-02-26T02:02:37.394055">G0Z1</Block>

803 In these examples, 238 is the CDATA for PartCount and is a numeric value; AUTO-
 804 MATIC is the CDATA for the ControllerMode and is a descriptive value representing
 805 a state for the *Data Entity*; and G0Z1 is a text string representing a message describing the
 806 program code associated with the Block *Data Entity*.

807 5.5.4 Unavailability of Valid Data Value for Event

808 If an *Agent* cannot determine a *Valid Data Value* for an Event element, the value returned
 809 for the CDATA for the *Data Entity* **MUST** be reported as UNAVAILABLE.

810 The example in *Example 8* demonstrates how an *Agent* reports the value for an Event in
 811 the CDATA when it is unable to determine a *Valid Data Value*:

Example 8: Example of Event Element when data value is UNAVAILABLE

```
812 1 <Events>
813 2   <ControllerMode dataItemId="p3"
814 3       timestamp="2009-02-26T02:02:35.716224" name="mode"
815 4       sequence="182">UNAVAILABLE</ControllerMode>
816 5 </Events>
```

817 5.6 Representations

818 A representation specifies the format and structure of the information for an *obser-*
 819 *vation*. The default representation is VALUE indicating the format as specified in
 820 *MTConnect Standard: Part 3.0 - Streams Information Model*.

821 A representation, other than VALUE, will modify the *Element Name* of the *obser-*
 822 *vation* by appending the pascal case of the representation as follows:

- 823 • A DataItem with type TEMPERATURE and representation of TIME_
 824 SERIES becomes TemperatureTimeSeries
- 825 • **DEPRECATED** A DataItem with type PART_COUNT and representa-
 826 tion of DISCRETE (**DEPRECATED** in *Version 1.5*) becomes PartCount-
 827 Discrete
- 828 • A DataItem with type VARIABLE and representation of DATA_SET be-
 829 comes VariableDataSet

- 830 • A `DataItem` with type `WORK_OFFSET` and representation of `TABLE` be-
831 comes `WorkOffsetTable`

832 The following constraints apply to each representation:

- 833 • A `DataItem` with representation `TIME_SERIES` **MUST** have a cate-
834 gory `SAMPLE`
- 835 • **DEPRECATED** A `DataItem` with representation `DISCRETE` (**DEPRECATED**
836 in *Version 1.5*) **MUST** have a category `EVENT`
- 837 • A `DataItem` with representation `DATA_SET` **MUST** have a category
838 `EVENT` or `SAMPLE`
- 839 • A `DataItem` with representation `TABLE` **MUST** have a category `EVENT`
840 or `SAMPLE`

841 5.6.1 Observations for `DataItem` with representation of `TIME_SE-` 842 `RIES`

843 A `DataItem` with `TIME_SERIES` representation **MUST** have a category of
844 `SAMPLE`.

845 A *Time Series observation* **MUST** have a `sampleCount` attribute.

846 *Time Series observation* **MUST** report multiple values at fixed intervals in a single *obser-*
847 *vation*. At minimum, one of `DataItem` or *observation* **MUST** specify the `sampleR-`
848 `ate` in *hertz* (values/second); fractional rates are permitted. When the *observation* and
849 the `DataItem` specify the `sampleRate`, the *observation* `sampleRate` supersedes
850 the `DataItem`.

851 The *observation* **MUST** set the `timestamp` to the time the last value was observed. The
852 `duration` **MAY** indicate the time interval from the first to the last value in the series.

853 In XML, the format of the *Time Series observation* **MUST** be space-separated floating-
854 point numbers.

855 5.6.1.1 XML Schema for Time Series Observation

856 *Figure 8* shows the attributes that can be applied to all `TIME_SERIES` *observations*.

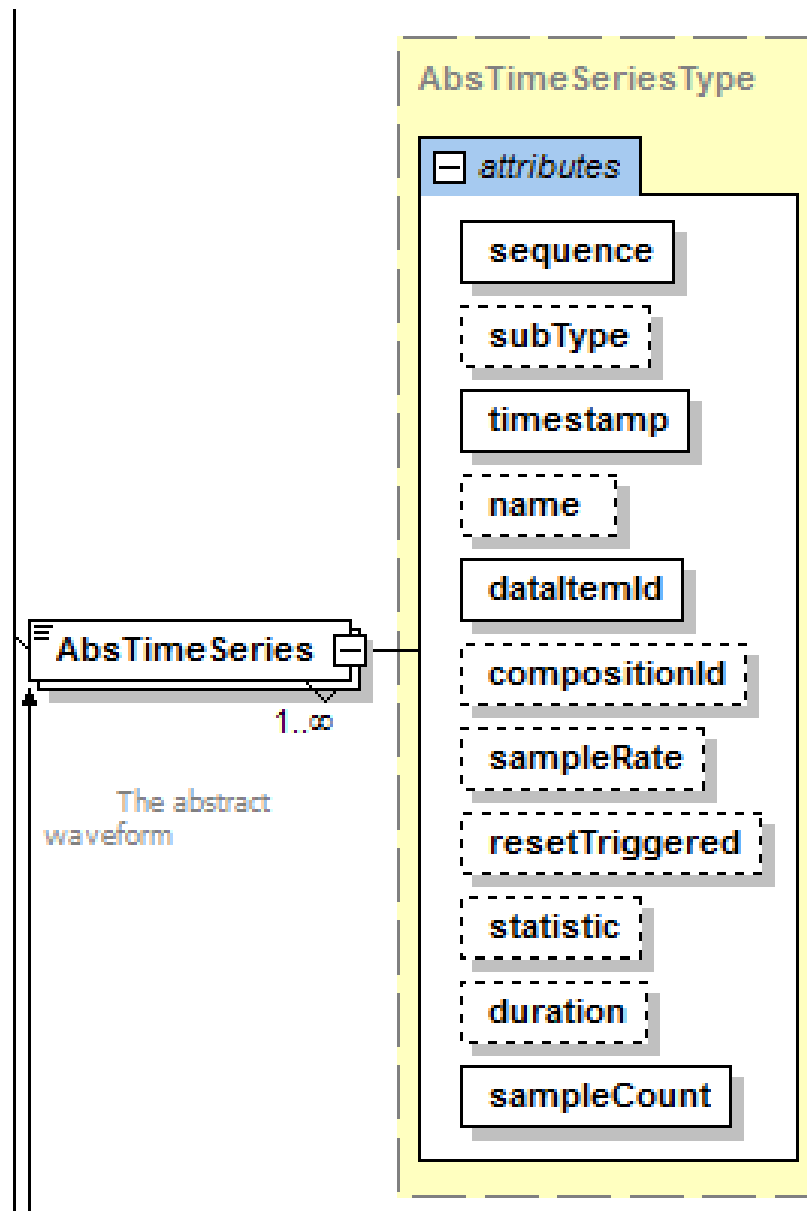


Figure 8: AbsTimeSeries Schema Diagram

857 5.6.1.2 Attributes for Time Series Observation

858 *Table 14* defines the additional attribute provided for a `DataItem` of category SAM-
859 PLE with a representation attribute of `TIME_SERIES`.

Table 14: Attributes for Time Series Observation

Attribute	Description	Occurrence
<code>sampleCount</code>	The number of values given for the <i>observation</i>	1

860 5.6.2 Observations for `DataItem` with representation of **DISCRETE** 861 **(DEPRECATED)**

862 *MTConnect* Version 1.5 replaced representation `DISCRETE` **(DEPRECATED in**
863 *Version 1.5)* with a discrete *attribute* for `DataItem`.

864 `DISCRETE` **(DEPRECATED in Version 1.5)** **MUST** only be used with a `DataItem`
865 with a category of `EVENT`.

866 Each occurrence of the *observation* **MAY** have the same value as the previous occurrence,
867 and **MUST NOT** suppress duplicates.

868 Examples of `DISCRETE` **(DEPRECATED in Version 1.5)** information as follows: A
869 `PartCount` reporting the completion of each part using a 1 to indicate completion of a
870 single part, a `Message` that occurs each time a door opens.

871 5.6.3 Observations for `DataItem` with representation of **DATA_SET**

872 A `DataItem` with `DATA_SET` representation **MUST** have a category of SAM-
873 PLE or `EVENT`.

874 A *Data Set observation* **MUST** have a `count` attribute.

875 *Data Set observation* reports multiple values as a set of *key-value pairs* where each *key*
876 **MUST** be unique. The representation of the *key-value pair* in XML is an `Entry`. The
877 value of each `Entry` **MUST** have the same constraints and format as the *observation*
878 defined for the `VALUE` representation for the `DataItem` type.

879 The meaning of each `Entry` **MAY** be provided as the `DataItemEntryDefinition`.

880 5.6.3.1 XML Schema for Data Set Observation

881 *Figure 9* represents the *XML Schema* of a *DataItem* with a representation at-
 882 tribute of *DATA_SET*.

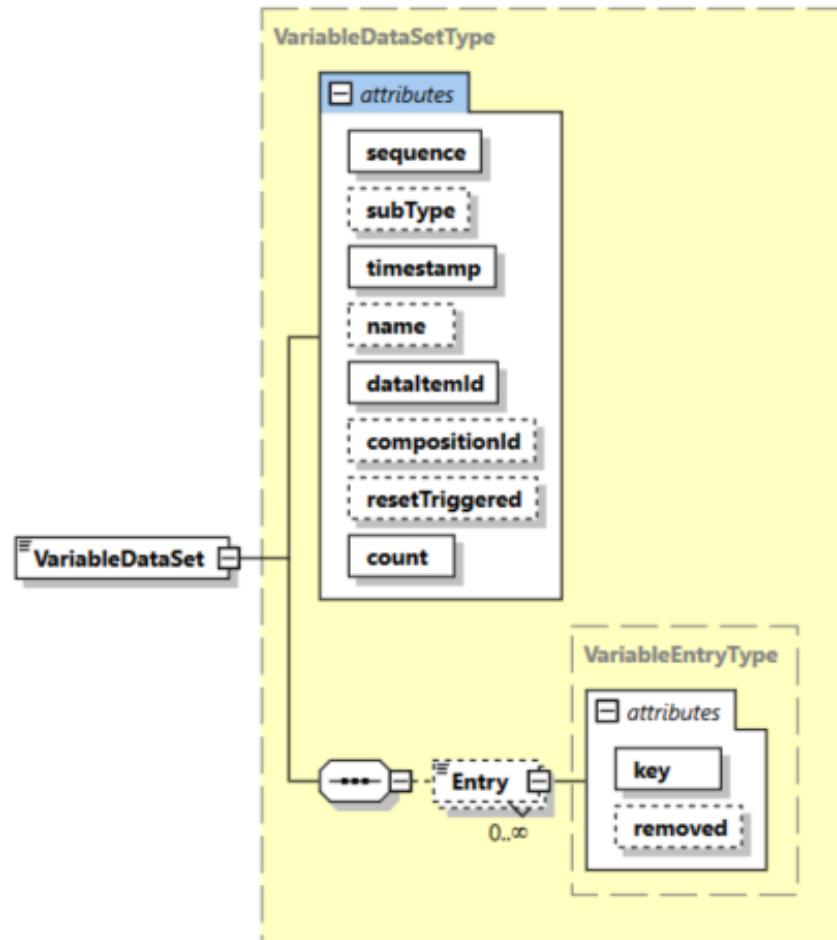


Figure 9: Sample Data Set Schema Diagram

883 *Table 15* defines the additional attribute provided for a *DataItem* with a represen-
 884 tation attribute of *DATA_SET*.

Table 15: Attributes for Data Set Observation

Attribute	Description	Occurrence
count	The number of <code>Entry</code> elements for the <i>observation</i> .	1

885 Table 16 defines the elements provided for a DataItem with a representation at-
886 tribute of DATA_SET.

Table 16: Elements for Data Set Observation

Element	Description	Occurrence
Entry	A key-value pair published as part of a Data Set observation.	0..*

887 5.6.3.2 Entry Element for Data Set Observation

888 Figure 10 represents the XML Schema structure for a Entry XML element that represents
889 the information published for a key-value pair. Any number of Entry elements MAY be
890 provided for a Data Entity defined with a representation attribute of DATA_SET.

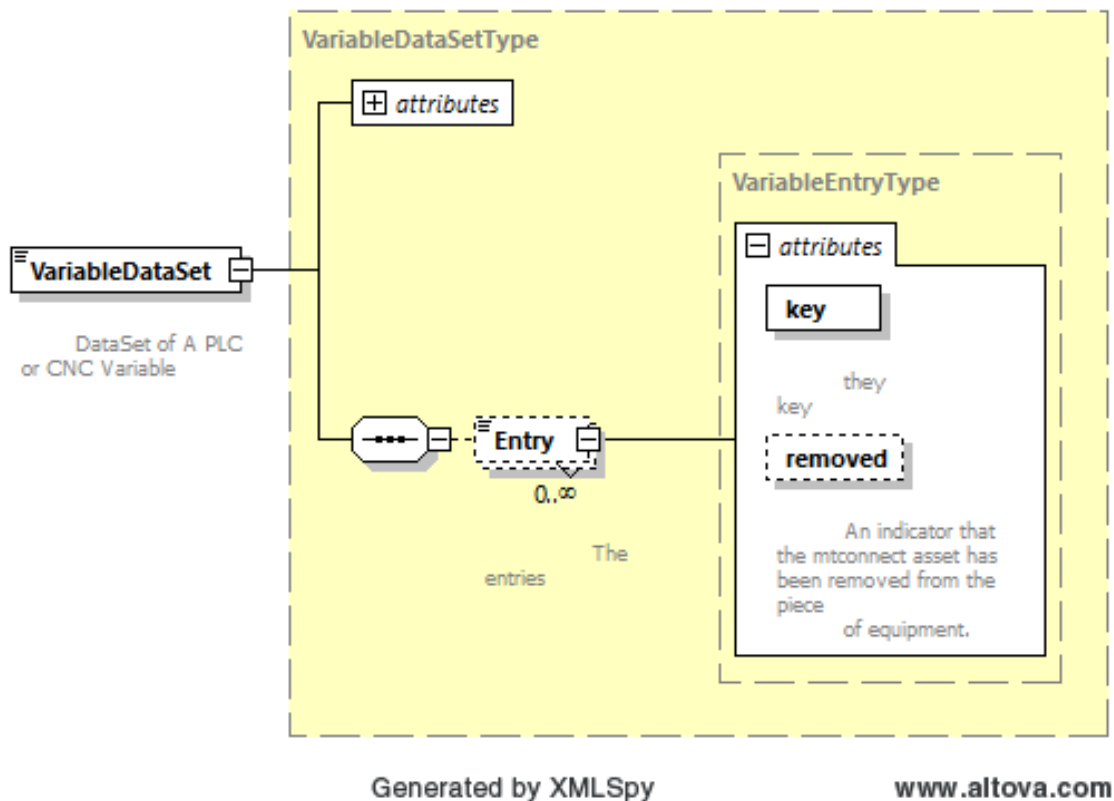


Figure 10: Entry Element Schema Diagram

891 Notes: The VariableDataSet is an example of a DataItem with type VARI-
892 ABLE and representation DATA_SET.

893 The following is an example in XML of Entry elements for a DataItem with type
 894 VARIABLE:

Example 9: Example of multiple key-value pairs Reported for a Data Entity

```

895 1 <VariableDataSet timestamp="..." sequence="..." count="2">
896 2   <Entry key="a101">100.21</Entry>
897 3   <Entry key="a102">609</Entry>
898 4   <Entry key="a103" removed="true" />
899 5 </VariableDataSet>

```

900 **5.6.3.3 Attributes for Entry Element for Data Set Observation**

901 *Table 17* defines the attributes provided for a Entry XML element.

Table 17: 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 a set of Entry elements.</p> <p>The value provided for key MUST be an XML NMTOKEN type.</p>	1
removed	<p>Boolean removal indicator of a <i>key-value pair</i> that MUST be true or false.</p> <p>true indicates the Entry is removed.</p> <p>false (default) indicates the Entry is present.</p>	0..1

902 **5.6.3.4 Constraints for Entry Values**

903 The value of each Entry **MUST** have the same restrictions as the value of an *observation*
 904 with representaton of VALUE.

905 An Entry **MAY** be further constrained by the DataItem definition (see *MTConnect*
 906 *Standard: Part 2.0 - Devices Information Model*), for example a VariableDataSet
 907 having a string value **MAY** have a floating-point Temperature value. A restriction
 908 **MUST NOT** be broadened or removed, for example, the value "READY" **MUST NOT**
 909 occur with a TemperatureDataSet constrained to floating-point numbers.

910 The *MTConnect Standard: Part 2.0 - Devices Information Model* `DataItem` Defini-
 911 tion **MAY** provide the type and units of an `Entry` for a key.

912 5.6.4 Management of Data Set Observations

913 An *Agent* **MUST** maintain the current state of the *Data Set* as described in *MTConnect*
 914 *Standard Part 1.0 - Overview and Fundamentals Section Part 1: Management of Stream-*
 915 *ing Data Storage*.

916 One or more *key-value pairs* **MAY** be added, removed, or changed in an *observation*. An
 917 *Agent* **MUST** publish the changes to one or more *key-value pairs* as a single *observation*.
 918 An *Agent* **MUST** indicate the removal of a *key-value pair* from a *Data Set* using the
 919 `removed` attribute equal `true`.

920 When the `DataItem` `discrete` attribute is `false` or is not present, an *Agent* in re-
 921 sponse to a *sample request* **MUST** only publish the changed *key-value pair* since the pre-
 922 vious state of the *Data Set*.

923 When the `DataItem` `discrete` attribute is `true`, an *Agent*, in response to a *sample*
 924 *request*, **MUST** report all *key-value pairs* ignoring the state of the *Data Set*.

925 When an *Agent* responds to a *Current Request*, the *response document* **MUST** include the
 926 full set of *key-value pairs*. If the *Current Request* includes an `at query parameter`, the
 927 *Agent* **MUST** provide the set of *key-value pairs* at the *sequence number*.

928 When an *observation reset* occurs, the *Data Set* **MUST** remove all *key-value pairs* making
 929 the set empty. The *observation* **MAY** simultaneously populate the *Data Set* with new
 930 *key-value pairs*. The previous entries **MUST NOT** be included and **MUST NOT** have
 931 `removed` attribute equal `true`.

932 When the *observation* is `UNAVAILABLE` the *Data Set* **MUST** remove all *key-value pairs*
 933 making the set empty.

934 5.6.5 Observations for `DataItem` with representation of `TABLE`

935 A *Table* represents two-dimensional sets of *key-value pairs* where the `Entry` represents
 936 rows containing sets of *key-value pairs* given by `Cell` elements. The *Table* has the same
 937 behavior as the *Data Set* for change tracking, clearing, and history. When an `Entry`
 938 changes. All `Cell` elements update at the same time; they are not tracked separately like
 939 `Entry`.

940 The meaning of each Entry and Cell **MAY** be provided as the DataItem Entry-
941 Definition and CellDefinition.

942 The Entry key attribute **MUST** be the unique identity of the Entry within an *obser-*
943 *vation*. The Cell key attribute **MUST** be the unique identity of the Cell within an
944 Entry.

945 **5.6.5.1 Structure of Table Observations**

946 *Figure 11* represents the XML schema representing DataItem defined in the *MTConnect*
947 *Standard: Part 2.0 - Devices Information Model* with a representation attribute of
948 TABLE.

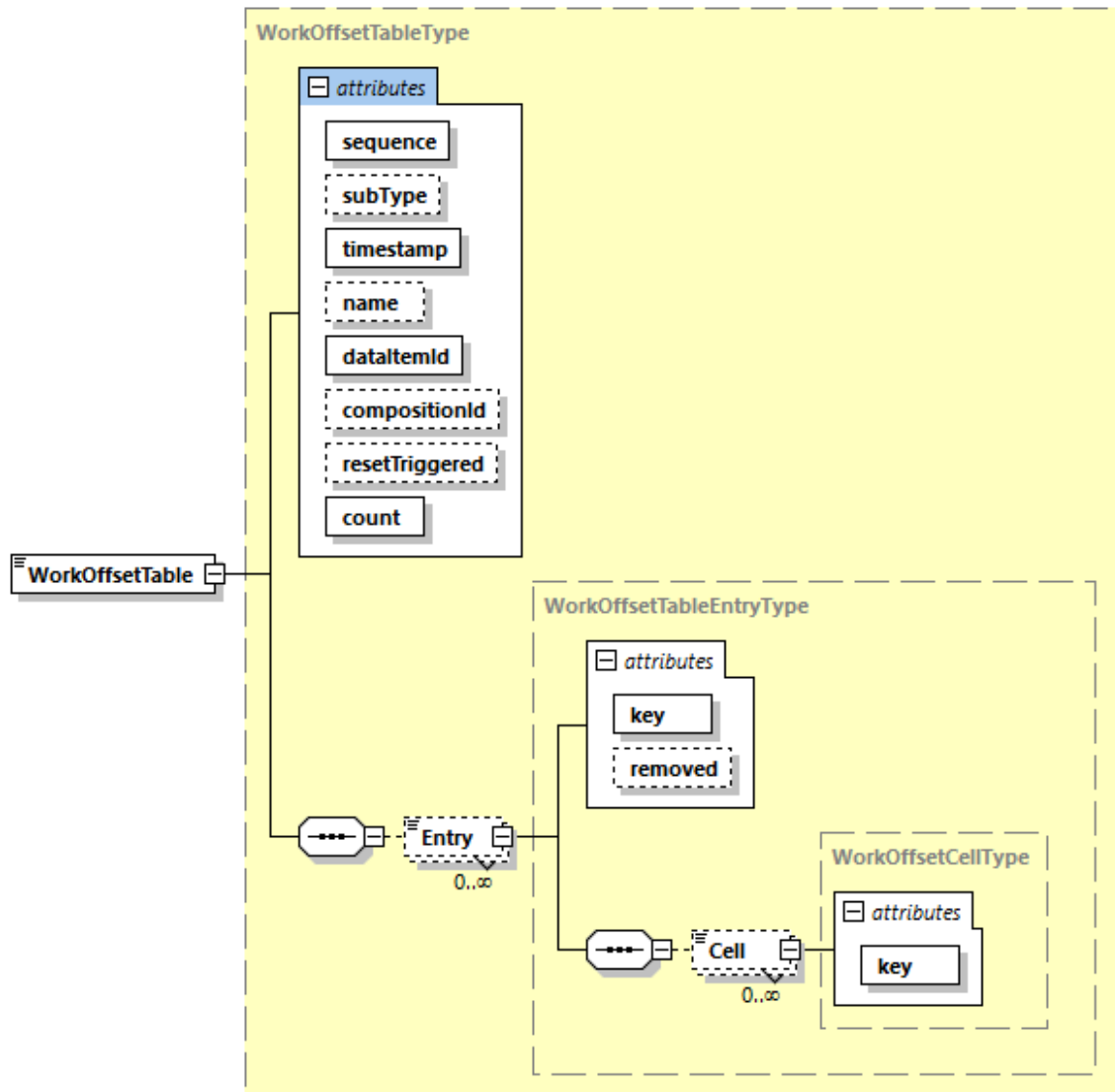


Figure 11: Table Schema Diagram

949 5.6.5.2 Attributes of Table Observations

Table 18: Attributes for Table

Attribute	Description	Occurrence
count	Represents the number of <i>key-value pairs</i> represented as <code>Entry</code> elements. count MUST be provided when the <code>DataItem</code> representation is <code>TABLE</code> .	1

950 5.6.5.3 Elements of Table Observations

951 *Table 19* An `Entry` is the only child element that **MAY** be associated with a *Table obser-*
952 *vation*.

Table 19: Elements for Table

Element	Description	Occurrence
Entry	A <i>key-value-pair</i> containing a set of <i>key-value pairs</i> .	0..*

953 5.6.5.3.1 Structure for Table Entry for an Observation

954 An `Entry` represents a *Row* subdivided into `Cell` elements when representing tabular
955 data. The meaning of an `Entry` **MAY** be given in the `DataItemEntryDefinition`
956 associated with its unique key.

957 5.6.5.3.2 Attributes for Table Entry for an Observation

958 See *Section 5.6.3.3 - Attributes for Entry Element for Data Set Observation*.

959 5.6.5.3.3 Elements for Table Cell for an Observation

Table 20: Elements for Table Cell

Element	Description	Occurrence
Cell	An element representing a <i>key-value pair</i> published as part of an Entry.	0..*

960 5.6.5.3.4 Structure for Table Cell for an Entry

961 A Cell represents a *Column* within a *Row* of a tabular data. The `DataItem CellDef-`
 962 `inition` **MAY** give the meaning of the Cell associated with its unique key.

963 Any number of Cell elements **MAY** be provided for an Entry for a *Table observation*.

964 The type of the `DataItem` constrains the *CDATA* of the Cell as specified in *MTConnect Standard: Part 2.0 - Devices Information Model*.
 965

966 5.6.5.3.5 Attributes for Table Cell for an Observation

967 Table 21 defines the attributes provided for a Cell XML element for an Entry.

Table 21: Attributes for Table Cell

Attribute	Description	Occurrence
key	A unique identifier for each <i>key-value pair</i> . The value provided for key MUST be unique in a set of Cell elements. The value provided for key MUST be an XML NMTOKEN type.	1

968 5.6.5.3.6 Constraints for Cell Values

969 The value of each Cell **MUST** have the same restrictions as the value of an *observation*
 970 with representaton of VALUE.

971 An Cell **MAY** be further constrained by the `DataItem` definition (see *MTConnect Stan-*

972 *dard: Part 2.0 - Devices Information Model*), for example a `VariableDataSet` having
 973 a string value **MAY** have a floating-point Temperature value. A restriction **MUST**
 974 **NOT** be broadened or removed, for example, the value "READY" **MUST NOT** occur
 975 with a `TemperatureDataSet` constrained limited to floating-point numbers.

976 The *MTConnect Standard: Part 2.0 - Devices Information Model* `DataItem` Defini-
 977 tion **MAY** provide the type and units of a `Cell` for a key.

978 5.6.5.3.7 Example Table Observation

Example 10: Example of `WorkpieceOffset` observation for a TABLE representation

```

979 1 <WorkpieceOffsetTable dataItemId="wp1" timestamp="TIME" name="wpo"
980 2   sequence="15" count="3">
981 3   <Entry key="G53.1"><Cell key="X">1</Cell><Cell key="Y">2</Cell>
982 4     <Cell key="Z">3</Cell></Entry>
983 5   <Entry key="G53.2"><Cell key="X">4</Cell><Cell key="Y">5</Cell>
984 6     <Cell key="Z">6</Cell></Entry>
985 7   <Entry key="G53.3"><Cell key="U">10</Cell><Cell key="X">7</Cell>
986 8     <Cell key="Y">8</Cell><Cell key="Z">9</Cell></Entry>
987 9 </WorkpieceOffsetTable>
  
```

988 5.7 Condition Container

989 `Condition` is a XML container type element. `Condition` organizes the *Data Entities*
 990 returned in the `MTConnectStreams` XML document for those `DataItem` elements
 991 defined with a category attribute of `CONDITION` in the `MTConnectDevices` docu-
 992 ment.

993 A separate `Condition` container will be provided for the data returned for the `DataItem`
 994 elements associated with each *Structural Element* of a piece of equipment defined in the
 995 `MTConnectDevices` document.

Table 22: 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

996 5.8 Condition Data Entity

997 A Condition XML element provides the information and data provided from a piece of
 998 equipment for those DataItem elements defined with a category attribute of CON-
 999 DITION in the MTConnectDevices document.

1000 Condition provides information reported by a piece of equipment describing its health
 1001 and ability to function.

1002 Condition is an abstract type XML element and will never appear directly in the MT-
 1003 ConnectStreams XML document. As an abstract type XML element, Condition
 1004 will be replaced in the XML document by a *Data Entity* representing the CONDITION
 1005 category DataItem element defined in the MTConnectDevices document that this
 1006 Condition element represents.

1007 The *Data Entities* represented by Condition are structured differently than the *Data*
 1008 *Entities* representing Sample and Event. The *Element Name* for each Condition
 1009 element reported in the MTConnectStreams document defines the *Fault State* of the
 1010 *Data Entity*. A Condition element is identified by the *Structural Element* to which it is
 1011 associated, along with the type and dataItemId defined for the element. *Section 6.3*
 1012 *- Types of Condition Elements* provides details on the different types of Condition
 1013 elements.

Table 23: 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 <code>DataItem</code> elements defined with a <code>category</code> attribute of <code>CONDITION</code> in the <code>MTConnectDevices</code> document.</p> <p><code>Condition</code> is an abstract type XML element. It is replaced in the <code>MTConnectStreams</code> document by a specific type of <code>Condition</code> element.</p> <p>There MAY be multiple types of <code>Condition</code> elements in a <code>Conditions</code> container.</p>	1..*

1014 `CONDITION` type `DataItem` elements defined in the `MTConnectDevices` document
1015 **MAY** report multiple simultaneous *Fault States* in the `MTConnectStreams` document.
1016 This is unlike a `SAMPLE` or `EVENT` `DataItem` element that can only report a single
1017 occurrence of a `Sample` or `Event` element in the `MTConnectStreams` document at
1018 any one point in time.

1019 For example, a controller on a piece of equipment may detect and report multiple for-
1020 mat errors in a motion program. Each error represents a separate *Fault State* from the
1021 controller. Each *Fault State* is represented as a separate `Condition` element in the `MT-`
1022 `ConnectStreams` document since each *Fault State* **MUST** be identified and tracked
1023 individually in the document.

1024 5.8.1 Element Names for Condition

1025 `Condition` elements are reported differently from other *Data Entity* types. The *El-*
1026 *ement Name* reported for a `Condition` element represents the *Fault State* (Normal,
1027 Warning, or Fault) associated with each `Condition`.

1028 Examples of XML elements representing `Condition` elements for each of the possible
1029 *Fault States* are shown in *Example 11*:

Example 11: Example of Condition Element Fault States

```

1030 1 <Normal type="MOTION_PROGRAM" dataItemId="cc2" sequence="25"
1031 2     timestamp="2010-04-06T06:19:35.153141"></Normal>
1032 3 <Fault type="COMMUNICATIONS" dataItemId="cc1" sequence="26"
```

```

1033 4      nativeCode="IO1231" timestamp="2010-04-
1034 5      06T06:19:35.153141">Communications error</Fault>
1035 6 <Warning type="LOGIC_PROGRAM" dataItemId="pm6" sequence="32"
1036 7      timestamp="2010-04-06T06:19:35.153141">Warning/>

```

1037 5.8.2 XML Schema Structure for Condition

1038 The XML schema in *Figure 12* represents the structure of a Condition XML element
 1039 showing the attributes defined for Condition elements.

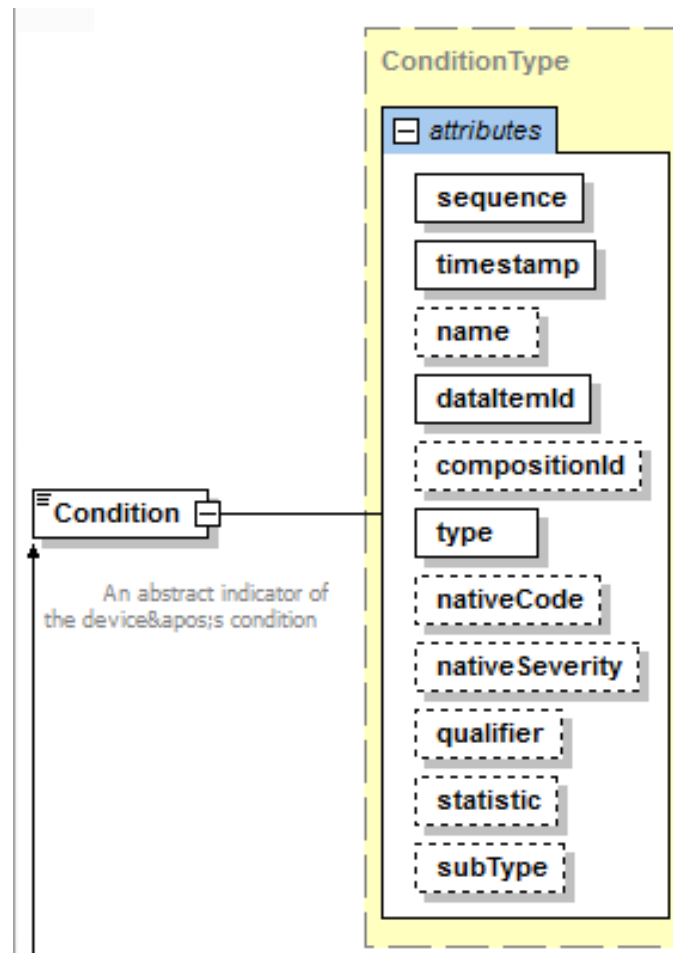


Figure 12: Condition Schema Diagram

1040 5.8.3 Attributes for Condition

1041 *Table 24* defines the attributes used to provide additional information for a `Condition`
 1042 XML element.

Table 24: Attributes for Condition

Attribute	Description	Occurrence
<code>sequence</code>	<p>A number representing the sequential position of an occurrence of the <code>Condition</code> in the data buffer of an MTConnect Agent.</p> <p><code>sequence</code> is a required attribute.</p> <p><code>sequence</code> MUST have a value represented as an unsigned 64-bit value from 1 to $2^{64} - 1$.</p>	1
<code>timestamp</code>	<p>The most accurate time available to a piece of equipment that represents the point in time that the data reported for the <code>Condition</code> was measured.</p> <p><code>timestamp</code> is a required attribute.</p>	1
<code>name</code>	<p>The name of the <code>Condition</code> element.</p> <p><code>name</code> is an optional attribute.</p> <p><code>name</code> MUST match the <code>name</code> attribute of the <code>DataItem</code> element defined in the MTConnectDevices document that the <code>Condition</code> element represents.</p> <p>An NMTOKEN XML type.</p>	0..1
<code>dataItemId</code>	<p>The unique identifier for the <code>Condition</code> element.</p> <p><code>dataItemId</code> is a required attribute.</p> <p><code>dataItemId</code> MUST match the <code>id</code> attribute of the <code>DataItem</code> element defined in the MTConnectDevices document that the <code>Condition</code> element represents.</p>	1

Continuation of Table 24		
Attribute	Description	Occurrence
type	<p>An identifier of the <code>type</code> of fault represented by the <code>Condition</code> element.</p> <p><code>type</code> is a required attribute.</p> <p><code>type</code> MUST match the <code>type</code> attribute of the <code>DataItem</code> element defined in the <code>MTConnectDevices</code> document that this <code>Condition</code> 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 <code>Condition</code>.</p> <p><code>nativeCode</code> 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, <code>nativeSeverity</code> reports that severity information to a client software application.</p> <p><code>nativeSeverity</code> is an optional attribute.</p>	0..1

Continuation of Table 24		
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 <i>Condition</i> 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 <i>Condition</i> element.</p> <p>statistic is an optional attribute.</p> <p>statistic MUST match the statistic attribute of the <i>DataItem</i> element defined in the <i>MTConnectDevices</i> document that this <i>Condition</i> element represents.</p>	0..1
subType	<p>subType provides additional information describing the meaning of the <i>Condition</i> element.</p> <p>subType is an optional attribute.</p> <p>subType MUST match the subType attribute of the <i>DataItem</i> element defined in the <i>MTConnectDevices</i> document that this <i>Condition</i> element represents.</p>	0..1

Continuation of Table 24		
Attribute	Description	Occurrence
compositionId	The identifier of the <code>Composition</code> element defined in the <code>MTConnectDevices</code> document associated with the data reported for the <code>Condition</code> element. compositionId is an optional attribute.	0..1
xs:lang	An optional attribute that specifies the language of the CDATA returned for the <code>Condition</code> . 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

1043 5.8.3.1 qualifier Attribute for Condition

1044 Many `Condition` elements report the *Fault State* associated with the measured value of
1045 a process variable.

1046 `qualifier` provides an indication whether the measured value is above or below an
1047 expected value of a process variable.

1048 As an example, a `Condition` element with a `type` attribute of `AMPERAGE` may differ-
1049 entiate between a higher than expected amperage and a lower than expected amperage by
1050 using the `qualifier` attribute.

1051 When a `qualifier` of either `HIGH` or `LOW` is used with `Fault` and `Warning`, the
1052 *Fault States* can be differentiated as follows:

1053 `Fault,LOW`

1054 `Warning,LOW`

1055 `Normal`

1056 `Warning,HIGH`

1057 Fault,HIGH

1058 *Example 12* is an example of an XML element representing Condition using quali-
 1059 fier:

Example 12: Example of a Condition Element using qualifier

```
1060     1 <Warning type="FILL_LEVEL" dataItemId="pm6"
1061       2       qualifier="HIGH" sequence="32"
1062       3       timestamp="2009-11-13T08:32:18">...</Warning>
```

1063 5.8.4 Valid Data Value for Condition

1064 Condition elements reported in an MTConnectStreams XML document **MAY** pro-
 1065 vide a value in the CDATA of the *Data Entity* when additional information regarding the
 1066 *Fault State* is available.

1067 A *Valid Data Value* for the CDATA included in a Condition element **MAY** be any text
 1068 string. A *Valid Data Value* is not required to be reported for a Condition category *Data*
 1069 *Entity*. The *Fault State* and the attributes provided in a Condition element **MAY** be
 1070 sufficient to fully describe the *Data Entity*.

1071 The *Valid Data Value* reported as CDATA for a Condition element **MUST** be formatted
 1072 as part of the content between the element tags in the XML element representing that *Data*
 1073 *Entity*. As an example, Condition elements are formatted as shown in *Example 13*:

Example 13: Example of CDATA for Condition

```
1074     1 <Warning type="FILL_LEVEL" dataItemId="pm6"
1075       2       qualifier="HIGH" sequence="32" timestamp=
1076       3       "2009-11-13T08:32:18">Fill Level on Tank
1077       4       #12 is reaching a high level</Warning>
```

1078 In this example, the “Fill Level on Tank #12 is reaching a high level” is the CDATA for
 1079 the *Data Entity*.

1080 5.9 Unavailability of Fault State for Condition

1081 When an *Agent* cannot determine a valid *Fault State* for a Condition element, it **MUST**
 1082 report the *Element Name* for the *Data Entity* as Unavailable.

1083 *Example 14* demonstrates how an *Agent* reports a Condition category *Data Entity* when
 1084 it is unable to determine a valid *Fault State*:

Example 14: Example of Condition when Fault State is UNAVAILABLE

```
1085 1 <Unavailable type="MOTION_PROGRAM" dataItemId="cc2"  
1086 2     sequence="25" timestamp=  
1087 3     "2009-11-13T08:32:18">...</Unavailable>  
1088 4 <Unavailable type="COMMUNICATIONS" dataItemId="cc1"  
1089 5     sequence="26" timestamp=  
1090 6     "2009-11-13T08:32:18">...</Unavailable>  
1091 7 <Unavailable type="LOGIC_PROGRAM" dataItemId="cc3"  
1092 8     sequence="28" timestamp=  
1093 9     "2009-11-13T08:32:18">...</Unavailable>  
1094 10 <Unavailable type="LOGIC_PROGRAM" dataItemId="pm6"  
1095 11     sequence="32" timestamp=  
1096 12     "2009-11-13T08:32:18">...</Unavailable>
```

1097 6 Listing of Data Entities

1098 *Data Entities* that report data in MTConnectStreams documents are represented by
 1099 Sample, Event, or Condition elements based upon the category and type at-
 1100 tributes defined for the corresponding DataItem XML element in the MTConnectDe-
 1101 vices document.

1102 Each *Data Entity* in the MTConnectStreams document has an *Element Name*, as de-
 1103 fined in the following sections, based upon the corresponding category attribute defined
 1104 for that DataItem element in the MTConnectDevices document.

1105 6.1 Sample Element Names

1106 *Table 25* lists the XML elements that can be placed in the Samples container of the
 1107 ComponentStream element.

1108 The *Table 25* shows both the type attribute for each SAMPLE category DataItem ele-
 1109 ment as defined in the MTConnectDevices document and the corresponding *Element*
 1110 *Name* for the *Data Entity* that **MUST** be reported as a Sample element in the MTCon-
 1111 nectStreams document.

Table 25: Element Names for Sample

DataItem Type	Element Name	Description
ACCELERATION	Acceleration	<p>The positive rate of change of velocity.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subType of ACTUAL.</p> <p>Subtypes of Acceleration are ACTUAL, COMMANDED and PROGRAMMED.</p> <p>Acceleration MUST be reported in units of MILLIMETER/SECOND².</p>

Continuation of Table 25: 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 SECOND.</p> <p>DEPRECATION WARNING : May be deprecated in the future. Recommend using ProcessTimer and EquipmentTimer.</p>
AMPERAGE	Amperage	DEPRECATED in Version 1.6. Replaced by AMPERAGE_AC and AMPERAGE_DC.
AMPERAGE_AC	AmperageAC	<p>The measurement of an electrical current that reverses direction at regular short intervals.</p> <p>Subtypes of AMPERAGE_AC are ACTUAL, COMMANDED and PROGRAMMED.</p> <p>AmperageAC is reported in units of AMPERE.</p>
AMPERAGE_DC	AmperageDC	<p>The measurement of an electric current flowing in one direction only.</p> <p>Subtypes of AMPERAGE_DC are ACTUAL, COMMANDED and PROGRAMMED.</p> <p>AmperageDC is reported in units of AMPERE.</p>

Continuation of Table 25: 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 positive rate of change of angular velocity.</p> <p>If a subType is not specified, the reported value for the data MUST default to the subType of ACTUAL.</p> <p>Subtypes of AngularAcceleration are ACTUAL, COMMANDED and PROGRAMMED.</p> <p>AngularAcceleration MUST be reported in units of DEGREE/SECOND².</p>
ANGULAR_- DECELERATION	AngularDeceleration	<p>Negative rate of change of angular velocity.</p> <p>Subtypes of AngularDeceleration are ACTUAL, COMMANDED and PROGRAMMED.</p> <p>AngularDeceleration MUST be reported in units of DEGREE/SECOND².</p>

Continuation of Table 25: Element Names for Sample		
DataItem Type	Element Name	Description
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>
ASSET_UPDATE_RATE	AssetUpdateRate	<p>The average rate of change of values for assets in the MTConnect streams. The average is computed over a rolling window defined by the implementation.</p> <p>AssetUpdateRate MUST be reported in units of COUNT/SECOND.</p>
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>

Continuation of Table 25: Element Names for Sample		
DataItem Type	Element Name	Description
CAPACITY_SPATIAL	CapacitySpatial	<p>The geometric capacity of an object or container.</p> <p>CapacitySpatial MUST be reported in units of CUBIC_MILLIMETER.</p>
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>

Continuation of Table 25: Element Names for Sample		
DataItem Type	Element Name	Description
DECELERATION	Deceleration	<p>Negative rate of change of velocity.</p> <p>Subtypes of Deceleration are ACTUAL, COMMANDED and PROGRAMMED.</p> <p>Deceleration MUST be reported in units of MILLIMETER/SECOND².</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>
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>

Continuation of Table 25: Element Names for Sample		
DataItem Type	Element Name	Description
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>
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>

Continuation of Table 25: Element Names for Sample		
DataItem Type	Element Name	Description
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>
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>

Continuation of Table 25: Element Names for Sample		
DataItem Type	Element Name	Description
DIAMETER	Diameter	<p>The measured dimension of a diameter.</p> <p>Diameter MUST be reported in units of 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 value of Wattage used or generated by a component over an interval of time.</p> <p>ElectricalEnergy MUST be reported in units of WATT_SECOND.</p>
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>

Continuation of Table 25: Element Names for Sample		
DataItem Type	Element Name	Description
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>
GLOBAL_POSITION	GlobalPosition	DEPRECATED in Version 1.1
HUMIDITY_-ABSOLUTE	HumidityAbsolute	<p>The amount of water vapor expressed in grams per cubic meter.</p> <p>Subtypes of HumidityAbsolute 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>HumidityAbsolute MUST be reported in units of GRAM/CUBIC_METER.</p>

Continuation of Table 25: Element Names for Sample		
DataItem Type	Element Name	Description
HUMIDITY_- RELATIVE	HumidityRelative	<p>The amount of water vapor present expressed as a percent to reach saturation at the same temperature.</p> <p>Subtypes of HumidityRelative 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>HumidityRelative MUST be reported in units of PERCENT.</p>
HUMIDITY_- SPECIFIC	HumiditySpecific	<p>The ratio of the water vapor present over the total weight of the water vapor and air present expressed as a percent.</p> <p>Subtypes of HumiditySpecific 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>HumiditySpecific MUST be reported in units of PERCENT.</p>

Continuation of Table 25: Element Names for Sample		
DataItem Type	Element Name	Description
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>A <i>Force</i> applied to a mass in one direction only.</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 25: Element Names for Sample		
DataItem Type	Element Name	Description
OBSERVATION_ UPDATE_RATE	ObservationUpdateRate	<p>The average rate of change of values for data items in the MTConnect streams. The average is computed over a rolling window defined by the implementation.</p> <p>ObservationUpdateRate MUST be reported in units of COUNT/SECOND.</p>
ORIENTATION	Orientation	<p>A measured or calculated orientation of a plane or vector relative to a cartesian coordinate system</p> <p>The value of Orientation MUST be three space-delimited floating-point numbers and MUST be in units of DEGREE_3D. The values represent the degrees of rotation around the X, Y, and Z axes respectively as the ordered values A, B, and C.</p> <p>If any of the rotations is not known, it MUST be zero (0).</p>

Continuation of Table 25: 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 25: 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 25: Element Names for Sample		
DataItem Type	Element Name	Description
PATH_POSITION (Continued)	PathPosition	An example of the value reported for PathPosition would be: <PathPosition ...>10.123 55.232 100.981 </PathPosition> 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 25: 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 25: 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 force per unit area measured relative to atmospheric pressure.</p> <p>Commonly referred to as gauge pressure.</p> <p>Pressure MUST be reported in units of PASCAL.</p>
PRESSURE_-ABSOLUTE	PressureAbsolute	<p>The force per unit area measured relative to a vacuum.</p> <p>PressureAbsolute MUST be reported in units of PASCAL.</p>

Continuation of Table 25: Element Names for Sample		
DataItem Type	Element Name	Description
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>
PRESSURIZATION_RATE	PressurizationRate	<p>The change of pressure per unit time.</p> <p>Subtypes of PressurizationRate are ACTUAL, COMMANDED and PROGRAMMED.</p> <p>PressurizationRate MUST be reported in units of PASCAL/SECOND.</p>
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>

Continuation of Table 25: Element Names for Sample		
DataItem Type	Element Name	Description
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>
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	DEPRECATED in Version 1.2. Replaced by ROTARY_VELOCITY

Continuation of Table 25: Element Names for Sample		
DataItem Type	Element Name	Description
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>Subtypes of Temperature 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>Temperature MUST be reported in units of CELSIUS.</p>
TENSION	Tension	<p>The measurement of a force that stretches or elongates an object.</p> <p>Tension MUST be reported in units of NEWTON.</p>
TILT	Tilt	<p>The measurement of angular displacement.</p> <p>Tilt MUST be reported in units of MICRO_RADIAN.</p>
TORQUE	Torque	<p>The measurement of the turning force exerted on an object or by an object.</p> <p>Torque MUST be reported in units of NEWTON_METER.</p>

Continuation of Table 25: 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>
VOLTAGE	Voltage	<p>DEPRECATED in <i>Version 1.6</i>. Replaced by VOLTAGE_AC and VOLTAGE_DC.</p>

Continuation of Table 25: Element Names for Sample		
DataItem Type	Element Name	Description
VOLTAGE_AC	VoltageAC	<p>The measurement of the electrical potential between two points in an electrical circuit in which the current periodically reverses direction.</p> <p>Subtypes of VOLTAGE_AC are ACTUAL, PROGRAMMED, and COMMANDED.</p> <p>VoltageAC MUST be in units of VOLT.</p>
VOLTAGE_DC	VoltageDC	<p>The measurement of the electrical potential between two points in an electrical circuit in which the current is unidirectional.</p> <p>Subtypes of VOLTAGE_DC are ACTUAL, PROGRAMMED, and COMMANDED.</p> <p>VoltageDC MUST be 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>

Continuation of Table 25: Element Names for Sample		
DataItem Type	Element Name	Description
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>
VOLUME_FLUID	VolumeFluid	<p>The fluid volume of an object or container.</p> <p>Subtypes of VolumeFluid are ACTUAL, START, ENDED, CONSUMED, WASTE, and PART.</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>

Continuation of Table 25: Element Names for Sample		
DataItem Type	Element Name	Description
VOLUME_SPATIAL	VolumeSpatial	<p>The geometric volume of an object or container.</p> <p>Subtypes of VolumeSpatial are ACTUAL, START, ENDED, CONSUMED, WASTE, and PART.</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>
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>

Continuation of Table 25: Element Names for Sample		
DataItem Type	Element Name	Description
X_DIMENSION	XDimension	Measured dimension of an entity relative to the X direction of the referenced coordinate system. XDimension MUST be reported in units of MILLIMETER.
Y_DIMENSION	YDimension	Measured dimension of an entity relative to the Y direction of the referenced coordinate system. YDimension MUST be reported in units of MILLIMETER.
Z_DIMENSION	ZDimension	Measured dimension of an entity relative to the Z direction of the referenced coordinate system. ZDimension MUST be reported in units of MILLIMETER.

1112 Note: The Sample response format **MUST** be extended when the represen-
 1113 tation attribute for the data item is TIME_SERIES. See *Section 5.6.1 -*
 1114 *Observations for DataItem with representation of TIME_SERIES* for details on
 1115 extending the response format.

1116 6.2 Event Element Names

1117 Table 26 lists the XML elements that can be placed in the Events container of the Com-
 1118 ponentStream element.

1119 The Table 25 shows both the type for each EVENT category DataItem element defined
 1120 in the MTConnectDevices document and the corresponding *Element Name* for the
 1121 Data Entity that **MUST** be reported as an Event element in the MTConnectStreams

1122 document.

1123 The table also defines the *Valid Data Value* for those Event type data items where the
 1124 reported values are restricted to a *Controlled Vocabulary*.

Table 26: Element Names for Event

DataItem Type	Element Name	Description
ACTIVATION_ COUNT	ActivationCount	<p>Accumulation of the number of times a function has attempted to, or is planned to attempt to, activate or be performed.</p> <p>When the discrete attribute is <code>true</code>, the value represents the count since the previous occurrence of the event.</p> <p>Subtypes of ActivationCount are ALL, GOOD, BAD, TARGET, REMAINING, COMPLETE, FAILED, and ABORTED.</p> <p>The <i>Valid Data Value</i> MUST be numeric.</p>

Continuation of Table 26: 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 26: Element Names for Event		
DataItem Type	Element Name	Description
ACTUATOR_- STATE	ActuatorState	Represents the operational state of an apparatus for moving or controlling a mechanism or system. <i>Valid Data Values:</i> ACTIVE: The actuator is operating INACTIVE: The actuator is not operating
ADAPTER_- SOFTWARE_- VERSION	AdapterSoftwareVersion	The originator's software version of the <i>Adapter</i> . The <i>Valid Data Value</i> MUST be a string.
ADAPTER_URI	AdapterURI	The URI of the <i>Adapter</i> . The <i>Valid Data Value</i> MUST be a string.
ALARM	Alarm	DEPRECATED : Replaced with <code>CONDITION</code> category data items in Version 1.1.0.
ALARM_LIMIT	AlarmLimit	A set of limits used to trigger warning or alarm indicators. The <i>Valid Data Value</i> MUST be a float. The <code>Entry key</code> MUST be one or more from the following: <code>UPPER_LIMIT</code> , <code>UPPER_WARNING</code> , <code>LOWER_WARNING</code> , or <code>LOWER_LIMIT</code> .

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
APPLICATION	Application	<p>The application on a component.</p> <p>Subtypes of APPLICATION are LICENSE, VERSION, RELEASE_DATE, INSTALL_DATE, and MANUFACTURER.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</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> <p>AVAILABLE: The <i>Structural Element</i> is active and capable of providing data.</p> <p>UNAVAILABLE: The <i>Structural Element</i> is either inactive or not capable of providing data.</p>

Continuation of Table 26: 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> <p>TANDEM: The axes are physically connected to each other and operate as a single unit.</p> <p>SYNCHRONOUS: The axes are not physically connected to each other but are operating together in lockstep.</p> <p>MASTER: The axis is the master of the CoupledAxes</p> <p>SLAVE: The axis is a slave to the CoupledAxes</p>

Continuation of Table 26: 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 <code>AxisFeedrateOverride</code> is expressed as a percentage of the designated feedrate for the axis.</p> <p>Subtypes of <code>AxisFeedrateOverride</code> are JOG, PROGRAMMED, and RAPID.</p> <p>If a <code>subType</code> is not specified, the reported value for the data MUST default to the <code>subType</code> of PROGRAMMED.</p> <p>The <i>Valid Data Value</i> MUST be a floating-point number.</p>

Continuation of Table 26: 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 26: 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> <p>HOME: The axis is in its home position.</p> <p>TRAVEL: The axis is in motion</p> <p>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.</p> <p>STOPPED: The axis is stopped</p>
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 26: 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> <p>ACTIVE: The chuck cannot be unclamped</p> <p>INACTIVE: The chuck can be unclamped.</p>

Continuation of Table 26: 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> <p>OPEN: The CHUCK component or composition element is open to the point of a positive confirmation</p> <p>CLOSED: The CHUCK component or composition element is closed to the point of a positive confirmation</p> <p>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.</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.</p>
CODE	Code	DEPRECATED in Version 1.1.

Continuation of Table 26: 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 <code>Composition</code> type element.</p> <p>Subtypes of <code>CompositionState</code> are <code>ACTION</code>, <code>LATERAL</code>, <code>MOTION</code>, <code>SWITCHED</code>, and <code>VERTICAL</code>.</p> <p>A <code>subType</code> MUST be provided.</p> <p><i>Valid Data Values</i> for <code>subType ACTION</code> are:</p> <p>ACTIVE: The <code>Composition</code> element is operating</p> <p>INACTIVE: The <code>Composition</code> element is not operating.</p> <p><i>Valid Data Values</i> for <code>subType LATERAL</code> are:</p> <p>RIGHT : The position of the <code>Composition</code> element is oriented to the right to the point of a positive confirmation</p> <p>LEFT : The position of the <code>Composition</code> element is oriented to the left to the point of a positive confirmation</p> <p>TRANSITIONING : The position of the <code>Composition</code> element is not oriented to the right to the point of a positive confirmation and is not</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
COMPOSITION_ STATE (Continued)	CompositionState	<p>oriented to the left to the point of a positive confirmation. It is in an intermediate position.</p> <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 26: Element Names for Event		
DataItem Type	Element Name	Description
COMPOSITION_ STATE (Continued)	CompositionState	<p><i>Valid Data Values</i> for subType MOTION are:</p> <p>OPEN: The position of the Composition element is open to the point of a positive confirmation</p> <p>CLOSED: The position of the Composition element is closed to the point of a positive confirmation</p> <p>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.</p>
CONNECTION_ STATUS	ConnectionStatus	<p>The status of the connection between an <i>Adapter</i> and an <i>Agent</i>.</p> <p><i>Valid Data Values:</i></p> <p>CLOSED: represents no connection at all.</p> <p>LISTEN: represents the <i>Agent</i> waiting for a connection request from an <i>Adapter</i>.</p> <p>ESTABLISHED: represents an open connection. The normal state for the data transfer phase of the connection.</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
CONTROL_- LIMIT	ControlLimit	<p>A set of limits used to indicate whether a process variable is stable and in control.</p> <p>The <i>Valid Data Value</i> MUST be a float.</p> <p>The Entry key MUST be one or more from the following: UPPER_LIMIT, UPPER_WARNING, NOMINAL, LOWER_WARNING, or LOWER_LIMIT.</p>

Continuation of Table 26: 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> <p>AUTOMATIC: The controller is configured to automatically execute a program.</p> <p>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.</p> <p>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.</p> <p>SEMI_AUTOMATIC: The controller is operating in a mode that restricts the active program from processing its next process step without operator intervention.</p> <p>EDIT: The controller is currently functioning as a programming device and is not capable of executing an active program.</p>

Continuation of Table 26: 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> <p>ON : The indicator of the ControllerModeOverride is in the ON state and the mode override is active.</p> <p>OFF : The indicator of the ControllerModeOverride is in the OFF state and the mode override is inactive</p>

Continuation of Table 26: 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 <code>AxisCoupling</code> to describe how the <code>CoupledAxes</code> 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 <code>MTConnectDevices</code> document that this Event element represents. If name is not available, <code>nativeName</code> MUST be returned to identify the Linear or Rotary <i>Structural Elements</i>.</p> <p>Example:</p> <pre><CoupledAxes ...>Y1 Y2</CoupledAxes></pre>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
CYCLE_COUNT	CycleCount	<p>Accumulation of the number of times a cyclic function has attempted to, or is planned to attempt to execute.</p> <p>When the <code>discrete</code> attribute is <code>true</code>, the value represents the count since the previous occurrence of the event.</p> <p>Subtypes of <code>CycleCount</code> are ALL, GOOD, BAD, TARGET, REMAINING, COMPLETE, FAILED, and ABORTED.</p> <p>The <i>Valid Data Value</i> MUST be numeric.</p>
DATE_CODE	DateCode	<p>The time and date code associated with a material or other physical item.</p> <p>Subtypes of <code>DateCode</code> are MANUFACTURE, EXPIRATION, and FIRST_USE.</p> <p>A <code>subType</code> MUST always be specified.</p> <p><code>DateCode</code> MUST be reported in ISO 8601 format.</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
DEACTIVATION_COUNT	DeactivationCount	<p>Accumulation of the number of times a function has attempted to, or is planned to attempt to, de-activate or cease.</p> <p>When the discrete attribute is <code>true</code>, the value represents the count since the previous occurrence of the event.</p> <p>Subtypes of <code>DeactivationCount</code> are ALL, GOOD, BAD, TARGET, REMAINING, COMPLETE, FAILED, and ABORTED.</p> <p>The <i>Valid Data Value</i> MUST be numeric.</p>
DEVICE_ADDED	DeviceAdded	<p>DeviceAdded is an Event that provides the UUID of a new device added to an <i>MTConnect Agent</i>.</p> <p><i>Valid Data Value</i> is the value of the Device's UUID that was added to the <i>MTConnect Agent</i>.</p>
DEVICE_CHANGED	DeviceChanged	<p>DeviceChanged is an Event that provides the UUID of the device whose <i>Metadata</i> has changed.</p> <p><i>Valid Data Value</i> is the value of the Device's UUID for which the metadata has changed.</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
DEVICE_- REMOVED	DeviceRemoved	<p>DeviceRemoved is an Event that provides the UUID of a device removed from an <i>MTConnect Agent</i>.</p> <p><i>Valid Data Value</i> is the value of the Device's UUID that was removed from the <i>MTConnect Agent</i>.</p>
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 NMTOKEN XML type.</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
DIRECTION	Direction	<p>The direction of motion.</p> <p>Subtypes of <code>Direction</code> are <code>ROTARY</code> and <code>LINEAR</code>.</p> <p><i>Valid Data Values</i> for subType <code>ROTARY</code> are as follows:</p> <p><code>CLOCKWISE</code>: Clockwise rotation using the right-hand rule.</p> <p><code>COUNTER_CLOCKWISE</code>: Counter-clockwise rotation using the right-hand rule.</p> <p><code>NONE</code>: No direction.</p> <p><i>Valid Data Values</i> for subType <code>LINEAR</code> are as follows:</p> <p><code>POSITIVE</code>: Linear position is increasing.</p> <p><code>NEGATIVE</code>: Linear position is decreasing.</p> <p><code>NONE</code>: No direction.</p>

Continuation of Table 26: 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> <p>OPEN: The DOOR is open to the point of a positive confirmation</p> <p>CLOSED: The DOOR is closed to the point of a positive confirmation</p> <p>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.</p>
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> <p>ARMED : The emergency stop circuit is complete and the piece of equipment, component, or composition element is allowed to operate.</p> <p>TRIGGERED : The emergency stop circuit is open and the operation of the piece of equipment, component, or composition element is inhibited.</p>

Continuation of Table 26: 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> <p>YES : The EndOfBar has been reached.</p> <p>NO : The EndOfBar has not been reached.</p>
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 DELAY, LOADED, WORKING, OPERATING, and POWERED.</p> <p>A subType MUST always be specified.</p> <p><i>Valid Data Values:</i></p> <p>ON : The equipment is functioning in the mode designated by the subType.</p> <p>OFF : The equipment is not functioning in the mode designated by the subType.</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
EXECUTION	Execution	<p>The execution status of a component.</p> <p><i>Valid Data Values:</i></p> <p>READY: The component is ready to execute instructions. It is currently idle.</p> <p>ACTIVE: The component is actively executing an instruction.</p> <p>INTERRUPTED: The component suspends the execution of the program due to an external signal. Action is required to resume execution.</p> <p>WAIT: The component suspends execution while a secondary operation executes. Execution resumes automatically once the secondary operation completes.</p> <p>FEED_HOLD: The motion of the active axes are commanded to stop at their current position.</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
EXECUTION (continued)	Execution	<p>STOPPED: The component program is not READY to execute.</p> <p>OPTIONAL_STOP: A command from the program has intentionally interrupted execution. The component MAY have another state that indicates if the execution is interrupted or the execution ignores the interrupt instruction.</p> <p>PROGRAM_STOPPED: A command from the program has intentionally interrupted execution. Action is required to resume execution.</p> <p>PROGRAM_COMPLETED: The program completed execution.</p>
FIRMWARE	Firmware	<p>The embedded software of a component.</p> <p>Subtypes of FIRMWARE are LICENSE, VERSION, RELEASE_DATE, INSTALL_DATE, and MANUFACTURER.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p>

Continuation of Table 26: 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 <code>FunctionalMode</code> SHOULD be associated with the <i>Device 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> <p>PRODUCTION : The <i>Device</i> 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.</p> <p>SETUP : The <i>Device</i> element or another <i>Structural Element</i> is not currently producing product. It is being prepared or modified to begin production of product.</p> <p>TEARDOWN : The <i>Device</i> 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.</p>

Continuation of Table 26: 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 26: Element Names for Event		
DataItem Type	Element Name	Description
HARDWARE	Hardware	<p>The hardware of a component.</p> <p>Subtypes of HARDWARE are LICENSE, VERSION, RELEASE_DATE, INSTALL_DATE, and MANUFACTURER.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p>
INTERFACE_ STATE	InterfaceState	<p>The current functional or operational state of an Interface type element indicating whether the <i>Interface</i> is active or not currently functioning.</p> <p><i>Valid Data Values:</i></p> <p>ENABLED: The <i>Interface</i> is currently operational and performing as expected.</p> <p>DISABLED: The Interface is currently not operational.</p> <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>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
LIBRARY	Library	<p>The software library on a component.</p> <p>Subtypes of LIBRARY are LICENSE, VERSION, RELEASE_DATE, INSTALL_DATE, and MANUFACTURER.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</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>
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>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
LOAD_COUNT	LoadCount	<p>Accumulation of the number of times an operation has attempted to, or is planned to attempt to, load materials, parts, or other items.</p> <p>When the discrete attribute is <code>true</code>, the value represents the count since the previous occurrence of the event.</p> <p>Subtypes of <code>LoadCount</code> are <code>ALL</code>, <code>GOOD</code>, <code>BAD</code>, <code>TARGET</code>, <code>REMAINING</code>, <code>COMPLETE</code>, <code>FAILED</code>, and <code>ABORTED</code>.</p> <p>The <i>Valid Data Value</i> MUST be numeric.</p>
LOCK_STATE	LockState	<p>The state or operating mode of a <code>Lock</code>.</p> <p><i>Valid Data Values:</i></p> <p><code>LOCKED</code>: The mechanism is engaged and preventing the associated component from being opened or operated.</p> <p><code>UNLOCKED</code>: The mechanism is disengaged and the associated component is able to be opened or operated.</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>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
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>
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>
MTCONNECT_ VERSION	MTConnectVersion	<p>The reference version of the MTConnect Standard supported by the <i>Adapter</i>.</p> <p>The <i>Valid Data Value</i> MUST be a string.</p>
NETWORK	Network	<p>Network details of a component.</p> <p>Subtypes of NETWORK are IPV4_ADDRESS, IPV6_ADDRESS, GATEWAY, SUBNET_MASK, VLAN_ID, MAC_ADDRESS, and WIRELESS.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
OPERATING_ SYSTEM	OperatingSystem	<p>The Operating System of a component.</p> <p>Subtypes of OPERATING_SYSTEM are LICENSE, VERSION, RELEASE_DATE, INSTALL_DATE, and MANUFACTURER.</p> <p>The <i>Valid Data Value</i> MUST be a text string.</p> <p>When specified with no subType, use the following vocabulary or specify the name of the operating system:</p> <ul style="list-style-type: none"> - WINDOWS - LINUX - MACINTOSH - PROPRIETARY
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>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
PART_COUNT	PartCount	<p>The aggregate count of parts.</p> <p>When the <code>discrete</code> attribute is <code>true</code>, the value represents the number of parts since the previous occurrence of the event.</p> <p>Subtypes of <code>PartCount</code> are <code>ALL</code>, <code>GOOD</code>, <code>BAD</code>, <code>TARGET</code>, <code>REMAINING</code>, <code>COMPLETE</code>, <code>FAILED</code>, and <code>ABORTED</code>.</p> <p>The <i>Valid Data Value</i> MUST be numeric.</p>
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> <p><code>PRESENT</code>: if a part or work piece has been detected or is present.</p> <p><code>NOT_PRESENT</code>: if a part or work piece is not detected or is not present.</p>
PART_GROUP_ID	PartGroupId	<p>Identifier given to a collection of individual parts. If no <code>subType</code> is specified, <code>UUID</code> is default.</p> <p>Subtypes of <code>PartGroupId</code> are <code>UUID</code>, <code>LOT</code>, <code>BATCH</code>, <code>RAW_MATERIAL</code> and <code>HEAT_TREAT</code>.</p> <p>The <i>Valid Data Value</i> MUST be a string.</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
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_KIND_ID	PartKindId	<p>Identifier given to link the individual occurrence to a class of parts, typically distinguished by a particular part design. If no subType is specified, UUID is default.</p> <p>Subtypes of PartKindId are UUID, PART_NUMBER, PART_FAMILY and PART_NAME.</p> <p>The <i>Valid Data Value</i> MUST be a string.</p>
PART_NUMBER	PartNumber	<p>DEPRECATED in Version 1.7. PART_NUMBER is now a subType of PART_KIND_ID.</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
PART_ PROCESSING_ STATE	PartProcessingState	<p>The particular condition of the part occurrence at a specific time.</p> <p><i>Valid Data Values:</i></p> <p>NEEDS_PROCESSING: The part occurrence is not actively being processed, but the processing has not ended. Processing requirements exist that have not yet been fulfilled. This is the default entry state when the part occurrence is originally received. In some cases, the part occurrence may return to this state while it waits for additional processing to be performed.</p> <p>IN_PROCESS: The part occurrence is actively being processed.</p> <p>PROCESSING_ENDED: The part occurrence is no longer being processed. A general state when the reason for termination is unknown.</p> <p>PROCESSING_ENDED_COMPLETE: The part occurrence has completed processing successfully.</p> <p>PROCESSING_ENDED_STOPPED: The process has been stopped during the processing. The part occurrence will require special treatment.</p> <p>PROCESSING_ENDED_ABORTED: The processing of the part occurrence has come to a premature end.</p> <p>PROCESSING_ENDED_COMPLETED:</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
PART_STATUS	PartStatus	<p>State or condition of a part.</p> <p>If unique identifier is given, part status is for that individual. If group identifier is given without a unique identifier, then the status is assumed to be for the whole group.</p> <p>The <i>Valid Data Value</i> MUST be:</p> <p>PASS: The part does conform to given requirements.</p> <p>FAIL: The part does not conform to some given requirements.</p>
PART_- UNIQUE_ID	PartUniqueId	<p>Identifier given to a distinguishable, individual part. If no subType is specified, UUID is default.</p> <p>Subtypes of PartUniqueId are UUID, SERIAL_NUMBER and RAW_MATERIAL.</p> <p>The <i>Valid Data Value</i> MUST be a string.</p>

Continuation of Table 26: 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 26: 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> <p>INDEPENDENT : The path is operating independently and without the influence of another path.</p> <p>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</p> <p>SYNCHRONOUS: The axes associated with the path are following the motion of the MASTER type path.</p> <p>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 .</p>

Continuation of Table 26: 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 26: 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_- AGGREGATE_ID	ProcessAggregateId	<p>Identifier given to link the individual occurrence to a group of related occurrences, such as a process step in a process plan.</p> <p>Subtypes of ProcessAggregateId are PROCESS_STEP, PROCESS_PLAN and ORDER_NUMBER.</p> <p>The <i>Valid Data Value</i> MUST be a string.</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
PROCESS_ KIND_ID	ProcessKindId	<p>Identifier given to link the individual occurrence to a class of processes or process definition.</p> <p>Subtypes of ProcessKindId are UUID, PROCESS_NAME and ISO_STEP_EXECUTABLE.</p> <p>The <i>Valid Data Value</i> MUST be a string.</p>
PROCESS_ OCCURRENCE_ ID	ProcessOccurrenceId	<p>An identifier of a process being executed by the device.</p> <p>The <i>Valid Data Value</i> MUST be a string.</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
PROCESS_STATE	ProcessState	<p>The particular condition of the process occurrence at a specific time. <i>Valid Data Values:</i></p> <p>INITIALIZING: The device is preparing to execute the process occurrence.</p> <p>READY: The process occurrence is ready to be executed.</p> <p>ACTIVE: The process occurrence is actively executing.</p> <p>COMPLETE: The process occurrence is now finished.</p> <p>INTERRUPTED: The process occurrence has been stopped and may be resumed.</p> <p>ABORTED: The process occurrence has come to a premature end and cannot be resumed.</p>
PROCESS_TIME	ProcessTime	<p>The time and date associated with an activity or event.</p> <p>Subtypes of <code>ProcessTime</code> are <code>START</code>, <code>COMPLETE</code>, and <code>TARGET_COMPLETION</code>.</p> <p>A <code>subType</code> MUST always be specified.</p> <p><code>ProcessTime</code> MUST be reported in ISO 8601 format.</p>

Continuation of Table 26: 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 26: 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 26: Element Names for Event		
DataItem Type	Element Name	Description
PROGRAM_ HEADER	ProgramHeader	<p>The non-executable header section of the control program.</p> <p>Subtypes of PROGRAM_HEADER are SCHEDULE, MAIN, and ACTIVE.</p> <p>The <i>Valid Data Value</i> MUST be a 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 26: 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 are:</i></p> <p>LOCAL: Managed by the controller.</p> <p>EXTERNAL: Not managed by the controller.</p>
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 26: 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> <p>SPINDLE: The axis is functioning as a spindle. Generally, it is configured to rotate at a defined speed.</p> <p>INDEX: The axis is configured to index to a set of fixed positions or to incrementally index by a fixed amount.</p> <p>CONTOUR: The position of the axis is being interpolated as part of the PathPosition defined by the Controller <i>Structural Element</i>.</p>
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</i> MUST be a floating-point number.</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
ROTATION	Rotation	<p>A three space angular rotation relative to a coordinate system.</p> <p>The value MUST be three floating-point numbers representing rotations around the X, Y, and Z axes in degrees.</p> <p>The values in XML are space delimited.</p>
SENSOR_- ATTACHMENT	SensorAttachment	<p>A SensorAttachment is an Event defining an <i>Attachment</i> between a sensor and an entity.</p> <p>The <i>Valid Data Value</i> MUST be a string.</p> <p>The Entry key MUST be from the following: SENSOR_ID</p>
SERIAL_- NUMBER	SerialNumber	<p>The serial number associated with a Component, Asset, or Device. The <i>Valid Data Value</i> MUST be a text string.</p>
SPECIFICATION_- LIMIT	SpecificationLimit	<p>A set of limits defining a range of values designating acceptable performance for a variable.</p> <p>The <i>Valid Data Value</i> MUST be a float.</p> <p>The Entry key MUST be one or more from the following: UPPER_LIMIT, NOMINAL, or LOWER_LIMIT.</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
SPINDLE_ INTERLOCK	SpindleInterlock	<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><i>Valid Data Values:</i></p> <p>ACTIVE: Power has been removed and the spindle cannot be operated.</p> <p>INACTIVE: Spindle has not been deactivated.</p>
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	<p>An identifier for the tool group associated with a specific tool. Commonly used to designate spare tools.</p> <p>The <i>Valid Data Value</i> MUST be any text string.</p>
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.
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>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
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>
TRANSFER_COUNT	TransferCount	<p>Accumulation of the number of times an operation has attempted to, or is planned to attempt to, transfer materials, parts, or other items from one location to another.</p> <p>When the discrete attribute is true, the value represents the count since the previous occurrence of the event.</p> <p>Subtypes of TransferCount are ALL, GOOD, BAD, TARGET, REMAINING, COMPLETE, FAILED, and ABORTED.</p> <p>The <i>Valid Data Value</i> MUST be numeric.</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
TRANSLATION	Translation	<p>A three space linear translation relative to a coordinate system.</p> <p>The value MUST be three floating-point numbers translation along the X, Y, and Z axes in millimeters.</p> <p>The values in XML are space delimited.</p>
UNLOAD_COUNT	UnloadCount	<p>Accumulation of the number of times an operation has attempted to, or is planned to attempt to, unload materials, parts, or other items.</p> <p>When the discrete attribute is true, the value represents the count since the previous occurrence of the event.</p> <p>Subtypes of UnloadCount are ALL, GOOD, BAD, TARGET, REMAINING, COMPLETE, FAILED, and ABORTED.</p> <p>The <i>Valid Data Value</i> MUST be numeric.</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
USER	User	<p>The identifier of the person currently responsible for operating the piece of equipment.</p> <p>Subtypes of <code>User</code> are <code>OPERATOR</code>, <code>MAINTENANCE</code>, and <code>SET_UP</code>.</p> <p>A <code>subType</code> MUST always be specified.</p> <p>The <i>Valid Data Value</i> MUST be any text string.</p>

Continuation of Table 26: Element Names for Event		
DataItem Type	Element Name	Description
VALVE_STATE	ValveState	<p>The state of a valve that is one of open, closed, or transitioning between the states.</p> <p>Subtypes of ValveState are ACTUAL and PROGRAMMED.</p> <p><i>Valid Data Values:</i></p> <p>OPEN: ValveState where flow is allowed and the aperture is static.</p> <p>Note : For a binary value, OPEN indicates that the valve has the maximum possible aperture.</p> <p>OPENING: The VALVE is transitioning from a CLOSED state to an OPEN state.</p> <p>CLOSED: ValveState where flow is not possible, the aperture is static and the valve is completely shut.</p> <p>CLOSING: The VALVE is transitioning from an OPEN state to a CLOSED state.</p>
VARIABLE	Variable	<p>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.</p> <p>The <i>Valid Data Value</i> MUST be a string.</p>

Continuation of Table 26: 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 are:</i></p> <p>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.</p> <p>POWERING_DOWN: An indication that the execution is waiting while the equipment is powering down but has not fully reached a stopped state.</p> <p>PART_LOAD: An indication that the execution is waiting while one or more discrete workpieces are being loaded.</p> <p>PART_UNLOAD: An indication that the execution is waiting while one or more discrete workpieces are being unloaded.</p> <p>TOOL_LOAD: An indication that the execution is waiting while a tool or tooling is being loaded.</p> <p>TOOL_UNLOAD: An indication that the execution is waiting while a tool or tooling is being unloaded.</p>

Continuation of Table 26: 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 26: 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>

1125 6.3 Types of Condition Elements

1126 As described in *Section 5.8 - Condition Data Entity*, *Condition Data Entities* are re-
1127 ported differently from other data item types. They are reported based on the *Fault State*
1128 for each *Condition*. Unlike *Sample* and *Event* data items that are identified by their
1129 *Element Name*, *Condition* data items are defined by the type and subType (where
1130 applicable) attributes defined for each *Condition*.

1131 The type and subType (where applicable) attributes for a *Condition* element **MAY**
1132 be any of the type and subType attributes defined for *SAMPLE* category or *EVENT*
1133 category data item listed in the *Devices Information Model*.

1134 Table *Section 5.8.1 - Element Names for Condition* lists additional *Condition Data En-*
1135 *tities* that have been defined to represent the health and fault status of *Structural Elements*.
1136 The table defines the type attribute for each of these additional *Condition* category

1137 elements that **MAY** be reported in the `MTConnectStreams` document.

Table 27: 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 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.

1138 Appendices

1139 A Bibliography

- 1140 Engineering Industries Association. *EIA Standard - EIA-274-D*, Interchangeable Variable,
 1141 Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically
 1142 Controlled Machines. Washington, D.C. 1979.
- 1143 ISO TC 184/SC4/WG3 N1089. *ISO/DIS 10303-238*: Industrial automation systems and
 1144 integration Product data representation and exchange Part 238: Application Protocols: Ap-
 1145 plication interpreted model for computerized numerical controllers. Geneva, Switzerland,
 1146 2004.
- 1147 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
 1148 tems and integration – Physical device control – Data model for computerized numerical
 1149 controllers – Part 10: General process data. Geneva, Switzerland, 2004.
- 1150 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
 1151 tems and integration – Physical device control – Data model for computerized numerical
 1152 controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.
- 1153 International Organization for Standardization. *ISO 6983/1* – Numerical Control of ma-
 1154 chines – Program format and definition of address words – Part 1: Data format for posi-
 1155 tioning, line and contouring control systems. Geneva, Switzerland, 1982.
- 1156 Electronic Industries Association. *ANSI/EIA-494-B-1992*, 32 Bit Binary CL (BCL) and
 1157 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines.
 1158 Washington, D.C. 1992.
- 1159 National Aerospace Standard. *Uniform Cutting Tests - NAS Series: Metal Cutting Equip-*
 1160 *ment Specifications*. Washington, D.C. 1969.
- 1161 International Organization for Standardization. *ISO 10303-11*: 1994, Industrial automa-
 1162 tion systems and integration Product data representation and exchange Part 11: Descrip-
 1163 tion methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.
- 1164 International Organization for Standardization. *ISO 10303-21*: 1996, Industrial automa-
 1165 tion systems and integration – Product data representation and exchange – Part 21: Imple-
 1166 mentation methods: Clear text encoding of the exchange structure. Geneva, Switzerland,
 1167 1996.
- 1168 H.L. Horton, F.D. Jones, and E. Oberg. *Machinery's Handbook*. Industrial Press, Inc.

- 1169 New York, 1984.
- 1170 International Organization for Standardization. *ISO 841-2001: Industrial automation sys-*
1171 *tems and integration - Numerical control of machines - Coordinate systems and motion*
1172 *nomenclature*. Geneva, Switzerland, 2001.
- 1173 ASME B5.57: *Methods for Performance Evaluation of Computer Numerically Controlled*
1174 *Lathes and Turning Centers*, 1998.
- 1175 ASME/ANSI B5.54: *Methods for Performance Evaluation of Computer Numerically Con-*
1176 *trolled Machining Centers*. 2005.
- 1177 OPC Foundation. *OPC Unified Architecture Specification, Part 1: Concepts Version 1.00*.
1178 July 28, 2006.
- 1179 IEEE STD 1451.0-2007, *Standard for a Smart Transducer Interface for Sensors and Ac-*
1180 *tuators – Common Functions, Communication Protocols, and Transducer Electronic Data*
1181 *Sheet (TEDS) Formats*, IEEE Instrumentation and Measurement Society, TC-9, The In-
1182 *stitute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH99684,*
1183 *October 5, 2007.*
- 1184 IEEE STD 1451.4-1994, *Standard for a Smart Transducer Interface for Sensors and Ac-*
1185 *tuators – Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet*
1186 *(TEDS) Formats*, IEEE Instrumentation and Measurement Society, TC-9, The Institute of
1187 *Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH95225, December*
1188 *15, 2004.*



MTConnect[®] Standard

Part 4.0 – Assets Information Model

Version 1.8.0

Prepared for: MTConnect Institute
Prepared on: September 6, 2021

MTConnect[®] is a registered trademark of AMT - The Association for Manufacturing Technology. Use of *MTConnect* is limited to use as specified on <http://www.mtconnect.org/>.

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	9
3	MTConnect Assets	10
3.1	Overview	10
3.2	MTConnectAssets	11
3.2.1	MTConnectAssets Header	11
3.2.1.1	Header Attributes	12
3.2.2	Assets	14
3.2.3	Asset	14
3.2.3.1	Common Asset Attributes	15
3.2.3.2	Common Asset Elements	18
4	MTConnect Assets Architecture	19
4.1	Agent Asset Storage	19
4.2	Asset Protocol	20
4.2.1	Asset by assetId	20
4.2.2	Asset for a Given Type	21
4.2.3	Assets Including Removed Assets	21
4.2.4	Assets for a Piece of Equipment	22
5	Extensions to Part 2.0 - Devices Information Model	23
5.1	Data Item Types added for EVENT Category	23
5.1.1	ASSET_CHANGED Data Item Type	23
5.1.2	ASSET_REMOVED Data Item Type	24
6	Extensions to Part 3.0 - Streams Information Model	25
6.1	AssetChanged Extension to Events	25
6.1.1	AssetChanged event Attributes	26
6.2	AssetRemoved Extension to Events	27
6.2.1	AssetRemoved Attributes	28
	Appendices	29
A	Bibliography	29

Table of Figures

Figure 1: MTConnectAssets Schema	11
Figure 2: MTConnectAssets Header	12
Figure 3: Asset Schema	16
Figure 4: Description Schema	18
Figure 5: MTConnect Assets storage as First in First Out	19
Figure 6: MTConnect Assets storage as Key/Value pairs	20
Figure 7: AssetChanged Schema	25
Figure 8: AssetRemoved Schema	27

List of Tables

Table 1: MTConnectAssets Header 13

Table 2: MTConnect Assets Element 14

Table 3: MTConnect Asset Element 15

Table 4: Attributes for Asset 16

Table 5: Elements for Asset 18

Table 6: DataItem Type for EVENT category 23

Table 7: Attributes for AssetChanged 26

Table 8: Attributes for AssetRemoved 28

1 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 XML

31 Stands for eXtensible Markup Language.

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

34 XML is the language used for all code examples in the MTConnect Standard.

35 Refer to <http://www.w3.org/XML> for more information about XML.

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

software systems by providing a structured response in the form of a *Response Document* that is constructed using the *semantic data models* defined in the Standard.

Appears in the documents in the following form: *Agent*.

43 ***Asset***

item, thing or entity that has potential or actual value to an organization *Ref:ISO 55000:2014(en)*

Note 1 to entry: Value can be tangible or intangible, financial or non-financial, and includes consideration of risks and liabilities. It can be positive or negative at different stages of the asset life.

Note 2 to entry: Physical assets usually refer to equipment, inventory and properties owned by the organization. Physical assets are the opposite of intangible assets, which are non-physical assets such as leases, brands, digital assets, use rights, licences, intellectual property rights, reputation or agreements.

Note 3 to entry: A grouping of assets referred to as an asset system could also be considered as an asset.

56 ***Child Element***

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

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

60 ***Component***

General meaning:

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

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

Used in Information Models:

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

- When used as an XML container to organize *Lower Level Component* elements.

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

- 71 • When used as an abstract XML element. `Component` is replaced in a data
72 model by a type of *Component* element. `Component` is also an XML con-
73 tainer used to organize *Lower Level* `Component` elements, *Data Entities*, or
74 both.

75 Appears in the documents in the following form: `Component`.

76 ***Current Request***

77 A *Current Request* is a *Request* to an *Agent* to produce an *MTConnectStreams Re-*
78 *sponse Document* containing the *Observations Information Model* for a snapshot of
79 the latest *observations* at the moment of the *Request* or at a given *sequence number*.

80 ***Data Entity***

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

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

85 ***Devices Information Model***

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

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

89 ***Equipment Metadata***

90 See *Metadata*

91 ***Information Model***

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

94 For example, an information model is used to define the structure for each *MTCon-*
95 *nect Response Document*; the definition of each piece of information within those
96 documents and the relationship between pieces of information.

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

98 ***Lower Level***

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

100 ***Metadata***

101 Data that provides information about other data.

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

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

107 ***MTConnect Agent***

108 See definition for *Agent*.

109 ***MTConnect Asset***

110 An *MTConnect Asset* is an *Asset* used by the manufacturing process to perform
111 tasks.

112 Note 1 to entry: An *MTConnect Asset* relies upon an *MTConnect Device* to
113 provide *observations* and information about itself and the *MTConnect Device*
114 revises the information to reflect changes to the *MTConnect Asset* during their
115 interaction. Examples of *MTConnect Assets* are Cutting Tools, Part Information,
116 Manufacturing Processes, Fixtures, and Files.

117 Note 2 to entry: A singular `assetId` uniquely identifies an *MTConnect Asset*
118 throughout its lifecycle and is used to track and relate the *MTConnect Asset* to
119 other *MTConnect Devices* and entities.

120 Note 3 to entry: *MTConnect Assets* are temporally associated with a device and
121 can be removed from the device without damage or alteration to its primary
122 functions.

123

124 ***MTConnect Device***

125 An *MTConnect Device* is a piece of equipment or a manufacturing system that pro-
126 duces *observations* about itself and/or publishes data using the *MTConnect Infor-
127 mation Model*.

128 ***MTConnect Information Model***

129 See *Information Model*

130 ***MTConnectDevices Response Document***

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

133 ***MTConnectStreams Response Document***

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

136 ***observation***

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

138 ***Observations Information Model***

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

141 ***Parent Element***

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

144 Appears in the documents in the following form: *Parent Element*.

145 ***Probe Request***

146 A *Probe Request* is a *Request* to an *Agent* to produce an *MTConnectDevices Re-*
147 *sponse Document* containing the *Devices Information Model*.

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

155 ***Sample Request***

156 A *Sample Request* is a *Request* to an *Agent* to produce an *MTConnectStreams Re-*
157 *sponse Document* containing the *Observations Information Model* for a set of time-
158 stamped *observations* made by *Components*.

159 ***semantic data model***

160 A methodology for defining the structure and meaning for data in a specific logical
161 way.

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

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

165 ***sequence number***

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

168 *sequence number* is a monotonically increasing number within an instance of an
169 *Agent*.

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

171 ***Streaming Data***

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

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

175 ***Structural Element***

176 General meaning:

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

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

180 Used to indicate hierarchy of Components:

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

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

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

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

187 ***Top Level***

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

190 ***Valid Data Value***

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

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

194 ***XML Schema***

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

197 2.2 Acronyms

198 **AMT**

199 The Association for Manufacturing Technology

200 2.3 MTConnect References

201 [MTConnect Part 1.0] *MTConnect Standard Part 1.0 - Overview and Fundamentals*. Ver-
202 sion 1.8.0.

203 [MTConnect Part 3.0] *MTConnect Standard: Part 3.0 - Streams Information Model*. Ver-
204 sion 1.8.0.

205 [MTConnect Part 4.0] *MTConnect Standard: Part 4.0 - Assets Information Model*. Ver-
206 sion 1.8.0.

207 3 MTConnect Assets

208 3.1 Overview

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

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

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

229 3.2 MTConnectAssets

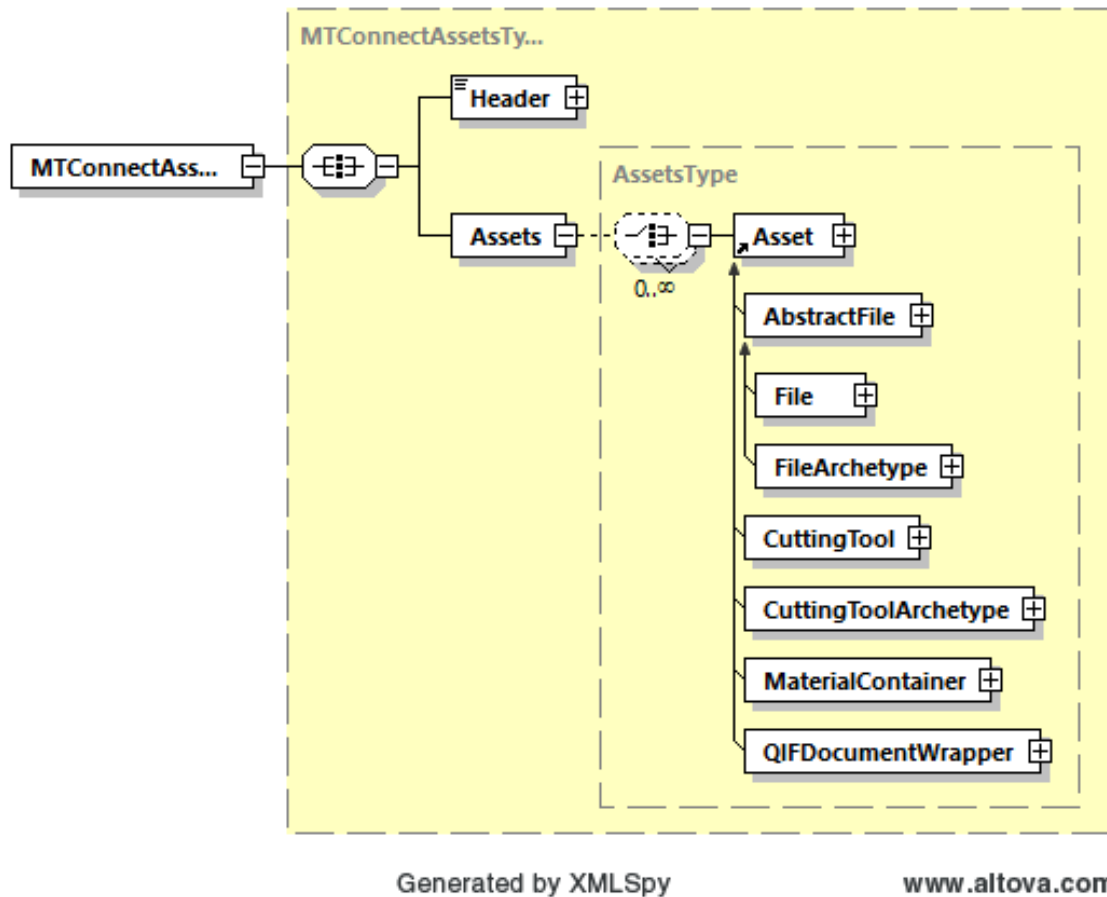


Figure 1: MTConnectAssets Schema

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

234 In the remaining Part 4.x sub-part documents of *MTConnect Assets*, various types of *Assets*
 235 will be introduced such as cutting tools and other *Asset* types.

236 3.2.1 MTConnectAssets Header

237 The `MTConnectAssets` header is where the protocol sequence information **MUST** be
 238 provided. The *XML Schema* in *Figure 2* represents the structure of the `MTConnectAs-`
 239 `sets` header showing the attributes defined for `MTConnectAssets`.

240 Refer to *MTConnect Standard Part 1.0 - Overview and Fundamentals* for more informa-
 241 tion on headers.

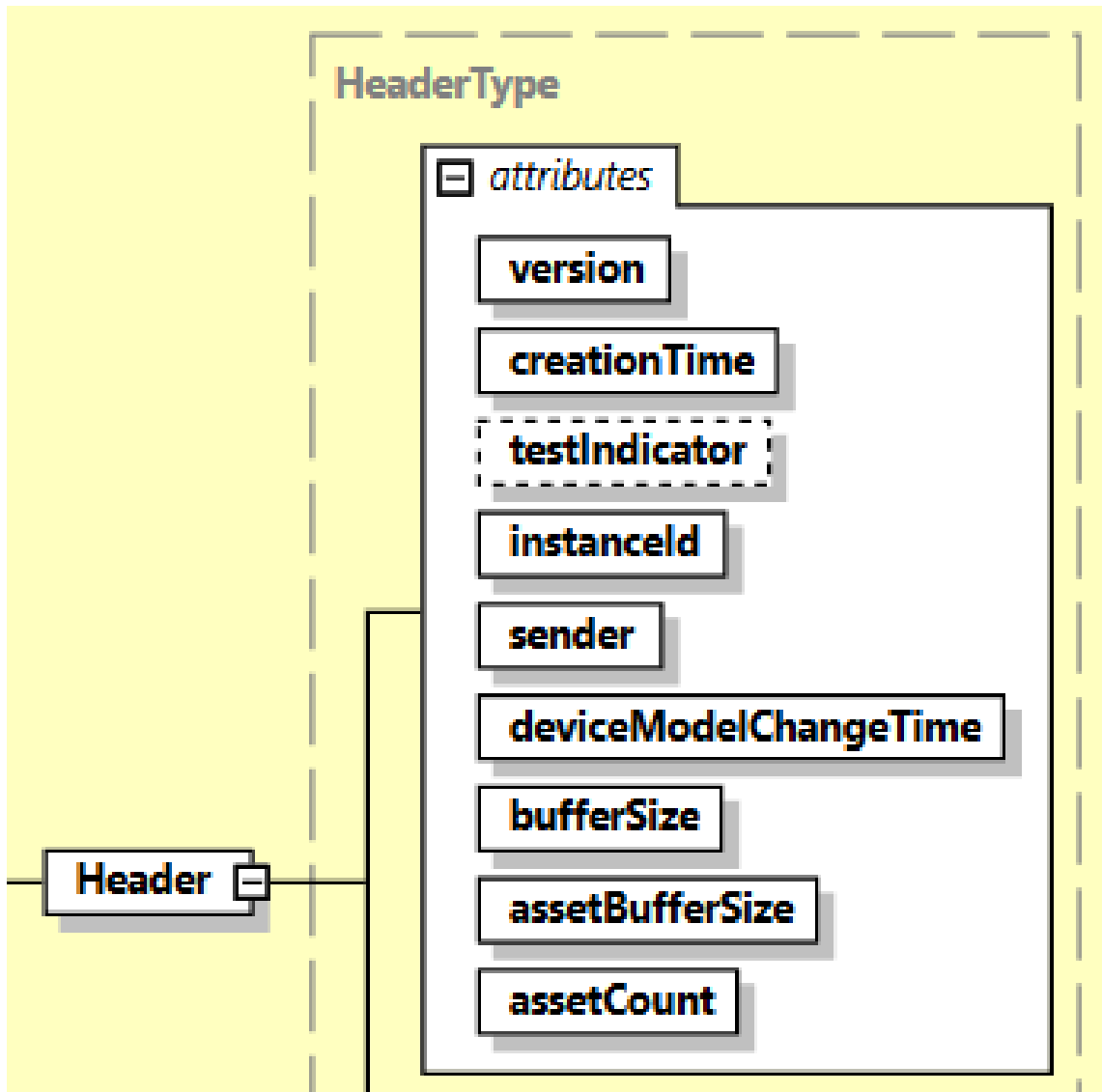


Figure 2: MTConnectAssets Header

242 3.2.1.1 Header Attributes

243 *Table 1* defines the attributes used to provide information for an `MTConnectAssets`
 244 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

Continuation of Table 1		
Attribute	Description	Occurrence
assetCount	The total number of <i>MTConnect Assets</i> in an <i>Agent</i> . This MUST be an unsigned positive integer value with a maximum value of $2^{32} - 1$. This value MUST NOT be greater than <code>assetBufferSize</code> . <code>assetCount</code> is a required attribute.	1
deviceModelChangeTime	A timestamp in 8601 format of the last update of the <code>Device</code> information for any device.	1

Example 1: MTConnectAssets Header Example

```

245 1 <Header creationTime="2010-03-13T07:59:11+00:00"
246 2     sender="localhost" instanceId="1268463594"
247 3     assetBufferSize="1024" version="1.1"
248 4     assetCount="12" />

```

3.2.2 Assets

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

Table 2: MTConnect Assets Element

Element	Description	Occurrence
<code>Assets</code>	An XML container that consists of one or more types of <code>Asset</code> XML elements.	0..1

3.2.3 Asset

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

256 type XML element, `Asset` will be replaced in the XML document by specific *MTConnect*
 257 *Asset* type.

Table 3: MTConnect Asset Element

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

258 There are various types of entities or *Asset* types. Each type of *Asset* is described in sub-
 259 parts of *MTConnect Standard: Part 4.0 - Assets Information Model*. These sub-parts are
 260 designated by a Part 4.x document number.

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

263 3.2.3.1 Common Asset Attributes

264 The *XML Schema* in *Figure 3* represents the structure of `Asset` showing the attributes
 265 defined for `Asset`.

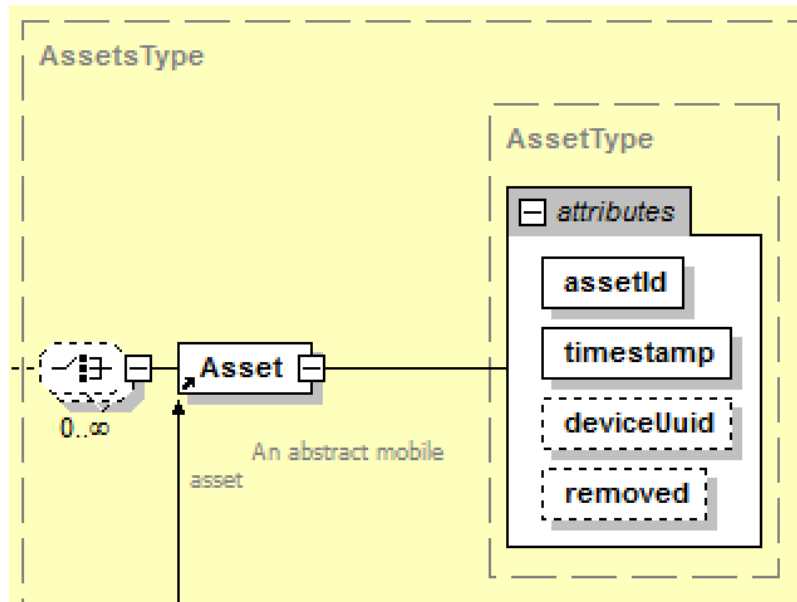


Figure 3: Asset Schema

266 *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
timestamp	The time this <i>MTConnect Asset</i> was last modified. Always given in UTC. The timestamp MUST be provided in UTC (Universal Time Coordinate, also known as GMT). This is the time the <i>Asset</i> data was last modified. timestamp is a required attribute.	1

Continuation of Table 4		
Attribute	Description	Occurrence
deviceUuid	The piece of equipments UUID that supplied this data. This is an optional element references to the UUID attribute given in the <i>Device</i> element. This can be any series of numbers and letters as defined by the XML type NMToken.	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

267 All *MTConnect Assets* **MUST** have a unique value for `assetId` and it **SHOULD** be
 268 globally unique, such as a RFC 4122 UUID.

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

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

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

285 3.2.3.2 Common Asset Elements

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

287 The *XML Schema* in *Figure 4* represents the structure of `Description`.

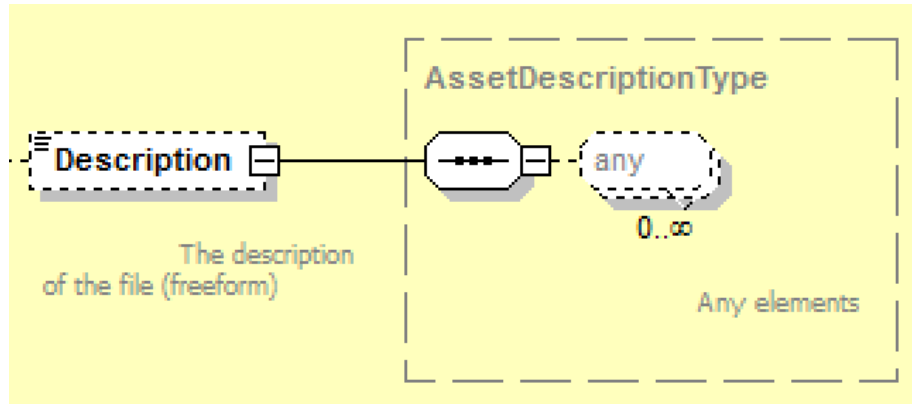


Figure 4: Description Schema

288 *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

289 4 MTConnect Assets Architecture

290 4.1 Agent Asset Storage

291 The *Agent* stores *MTConnect Assets* in a similar fashion as the *Agent* data storage de-
 292 scribed in *MTConnect Standard Part 1.0 - Overview and Fundamentals*. The storage of
 293 information is contained in the *asset buffer*. The *Agent* provides a limited number of *As-*
 294 *sets* that can be stored at one time and uses the same method of pushing out the oldest
 295 *Asset* when the *asset buffer* is full. The *asset buffer* size for the *Asset* storage is maintained
 296 separately from the *Sample*, *Event*, and *Condition* storage.

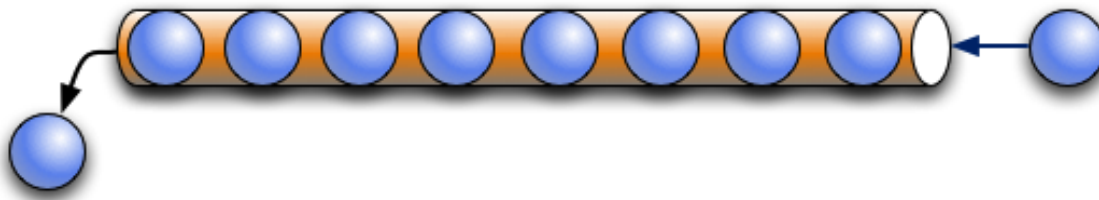


Figure 5: MTConnect Assets storage as First in First Out

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

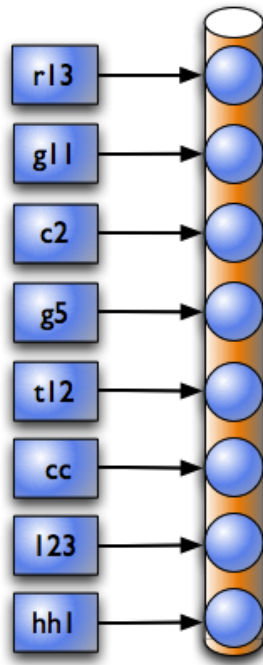


Figure 6: MTConnect Assets storage as Key/Value pairs

303 As in *Figure 6* , each of the *Assets* is referred to by their key. The key is independent of
 304 the order in the *asset buffer* storage.

305 4.2 Asset Protocol

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

309 The URL format is similar to the `probe` and `sample` structure. Reference each as-
 310 setId directly to request an *MTConnect Asset* by `assetId`.

311 4.2.1 Asset by assetId

Example 2: Asset by assetId Example

```
312 1 url: http://example.com/asset/e39d23ba-ef2d-
313 2     11e6-b12c15028cfe91a82ef
```

314 *Example 2* returns the `MTConnectAssets` document for *Asset* `e39d23ba-ef2d-`
 315 `11e6-b12c-28cfe91a82ef`

316 Request multiple *Assets* by each `assetId`:

Example 3: Assets by assetId Example

317 1 url: `http://example.com/asset/e39d23ba-ef2d-11e6-b12c155;`
 318 2 `8cfe91a82ef;e46d5256-ef2d-11e6-96aa-28cfe91a82ef`

319 *Example 3* returns the `MTConnectAssets` document for *Assets* `e39d23ba-ef2d-`
 320 `11e6-b12c-28cfe91a82ef` and `e46d5256-ef2d-11e6-96aa-28cfe91a82ef`.

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

Example 4: Get all Assets Example

322 1 url: `http://example.com/assets`

323 *Example 4* returns all available *MTConnect Assets* in the *Agent*. The *Agent* **MAY** return
 324 a limited set if there are too many *Asset* records. The *Assets* **MUST** be added to the
 325 beginning with the most recently modified *Asset*.

326 4.2.2 Asset for a Given Type

Example 5: Asset for a Given Type Example

327 1 url: `http://example.com/assets?type="CuttingTool"`

328 *Example 5* returns all available `CuttingTool Assets` from the *Agent* of the type `Cut-`
 329 `tingTool`. The *Agent* **MAY** return a limited set if there are too many *Asset* records. The
 330 *Assets* **MUST** be added to the beginning with the most recently modified assets.

331 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

332 1 url: `http://example.com/assets?type="CuttingTool"`

333 *Example 6* returns all available `CuttingTool Assets` from the *Agent*. The *Agent* **MUST**
 334 return up to 1000 *Assets* beginning with the most recently modified *Assets* if they exist.

335 4.2.3 Assets Including Removed Assets

Example 7: Assets Including Removed Assets Example

336 1 url: `http://example.com/assets?type=CuttingTool&removed=true`

337 *Example 7* returns all available `CuttingTool Assets` from the *Agent*. With the removed
 338 flag, *Assets* that have been removed but are included in the result set.

339 4.2.4 Assets for a Piece of Equipment

340 If no `assetId` is provided with a general *Assets* request, it would be as shown in *Exam-*
 341 *ple 8*:

Example 8: Assets For a Piece of Equipment Example

342 1 url: `http://example.com/Mill123/assets`

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

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

Example 9: Assets For a Piece of Equipment For a Given Type Example

348 1 url: `http://example.com/Mill123/asset/`
 349 2 `?type=CuttingTool&count=100`

350 The request in *Example 9* will get the newest 100 Cutting Tool Instance *Assets* from the
 351 *Agent* for `Mill123`. Similarly:

Example 10: Assets For a Piece of Equipment For a Given Type Example 2

352 1 url: `http://example.com/Mill123/asset/`
 353 2 `?type=CuttingToolArchetype`

354 *Example 10* will provide all Cutting Tool Archetype *Assets* with the `deviceUuid` of
 355 `Mill123`.

356 5 Extensions to Part 2.0 - Devices Information Model

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

361 5.1 Data Item Types added for EVENT Category

Table 6: `DataItem` Type for EVENT category

DataItem Type SubType	Description
ASSET_CHANGED	The event generated when an asset is added or changed. <code>AssetChanged</code> MUST be discrete and the value of the <code>DataItem</code> 's discrete attribute MUST be <code>true</code> .
ASSET_REMOVED	The value of the <code>CData</code> for the event MUST be the <code>assetId</code> of the asset that has been removed. The asset will still be visible if requested with the <code>includeRemoved</code> parameter as described in the protocol section. When assets are removed they are not moved to the beginning of the most recently modified list.

362 5.1.1 ASSET_CHANGED Data Item Type

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

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

372 5.1.2 ASSET_REMOVED Data Item Type

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

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

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

387 6 Extensions to Part 3.0 - Streams Information Model

388 The associated modifications **MUST** be added to *MTConnect Standard: Part 3.0 - Streams*
 389 *Information Model* to add the following event to the `Events` in the streams.

390 6.1 AssetChanged Extension to Events

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

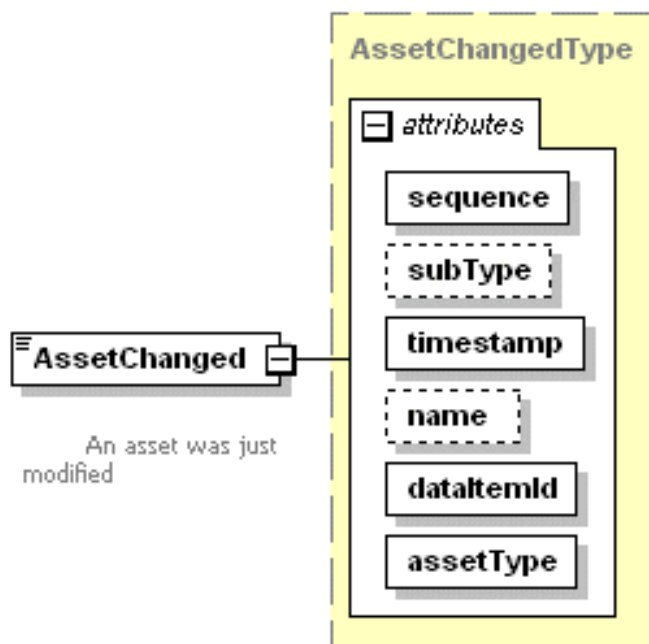


Figure 7: AssetChanged Schema

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

399 6.1.1 AssetChanged event Attributes**Table 7:** Attributes for AssetChanged

Attribute	Description	Occurrence
assetType	<p>The type of asset changed.</p> <p>assetType is a required attribute.</p> <p><i>Valid Data Values:</i></p> <p>CuttingTool</p> <p>File</p> <p>QIFDocumentWrapper</p> <p>MaterialContainer</p>	1

400 6.2 AssetRemoved Extension to Events

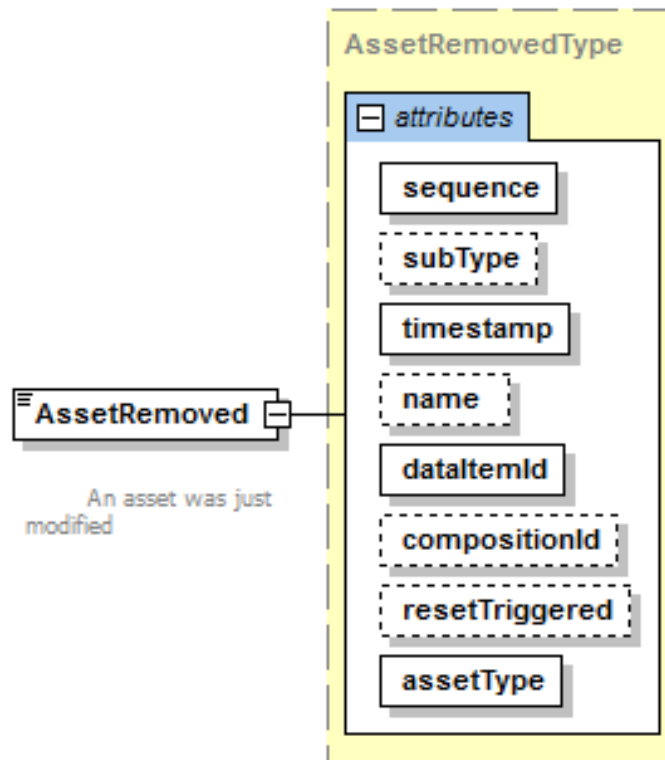


Figure 8: AssetRemoved Schema

401 `AssetRemoved`: An *MTConnect Asset* has been removed. The CDATA for the `As-`
 402 `setRemoved` element **MUST** be the `assetId` of the *Asset* that has been removed.

403 6.2.1 AssetRemoved Attributes

Table 8: Attributes for AssetRemoved

Attribute	Description	Occurrence
assetType	<p>The type of asset that was removed.</p> <p>assetType is a required attribute.</p> <p><i>Valid Data Values:</i></p> <p>CuttingTool</p> <p>File</p> <p>QIFDocumentWrapper</p> <p>MaterialContainer</p>	1

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

407 Appendices

408 A Bibliography

- 409 Engineering Industries Association. *EIA Standard - EIA-274-D*, Interchangeable Variable,
 410 Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically
 411 Controlled Machines. Washington, D.C. 1979.
- 412 ISO TC 184/SC4/WG3 N1089. *ISO/DIS 10303-238*: Industrial automation systems and
 413 integration Product data representation and exchange Part 238: Application Protocols: Ap-
 414 plication interpreted model for computerized numerical controllers. Geneva, Switzerland,
 415 2004.
- 416 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
 417 tems and integration – Physical device control – Data model for computerized numerical
 418 controllers – Part 10: General process data. Geneva, Switzerland, 2004.
- 419 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
 420 tems and integration – Physical device control – Data model for computerized numerical
 421 controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.
- 422 International Organization for Standardization. *ISO 6983/1* – Numerical Control of ma-
 423 chines – Program format and definition of address words – Part 1: Data format for posi-
 424 tioning, line and contouring control systems. Geneva, Switzerland, 1982.
- 425 Electronic Industries Association. *ANSI/EIA-494-B-1992*, 32 Bit Binary CL (BCL) and
 426 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines.
 427 Washington, D.C. 1992.
- 428 National Aerospace Standard. *Uniform Cutting Tests - NAS Series: Metal Cutting Equip-*
 429 *ment Specifications*. Washington, D.C. 1969.
- 430 International Organization for Standardization. *ISO 10303-11*: 1994, Industrial automa-
 431 tion systems and integration Product data representation and exchange Part 11: Descrip-
 432 tion methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.
- 433 International Organization for Standardization. *ISO 10303-21*: 1996, Industrial automa-
 434 tion systems and integration – Product data representation and exchange – Part 21: Imple-
 435 mentation methods: Clear text encoding of the exchange structure. Geneva, Switzerland,
 436 1996.
- 437 H.L. Horton, F.D. Jones, and E. Oberg. *Machinery's Handbook*. Industrial Press, Inc.

- 438 New York, 1984.
- 439 International Organization for Standardization. *ISO 841-2001: Industrial automation sys-*
440 *tems and integration - Numerical control of machines - Coordinate systems and motion*
441 *nomenclature*. Geneva, Switzerland, 2001.
- 442 *ASME B5.59-2 Version 9c: Data Specification for Properties of Machine Tools for Milling*
443 *and Turning*. 2005.
- 444 *ASME/ANSI B5.54: Methods for Performance Evaluation of Computer Numerically Con-*
445 *trolled Machining Centers*. 2005.
- 446 OPC Foundation. *OPC Unified Architecture Specification, Part 1: Concepts Version 1.00*.
447 July 28, 2006.
- 448 International Organization for Standardization. *ISO 13399: Cutting tool data representa-*
449 *tion and exchange*. Geneva, Switzerland, 2000.



MTConnect[®] Standard

Part 4.1 – Cutting Tools

Version 1.8.0

Prepared for: MTConnect Institute
Prepared on: September 6, 2021

MTConnect[®] is a registered trademark of AMT - The Association for Manufacturing Technology. Use of *MTConnect* is limited to use as specified on <http://www.mtconnect.org/>.

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	9
3	Cutting Tool and Cutting Tool Archetype	10
3.1	XML Schema Structure for CuttingTool and CuttingToolArchetype	10
3.2	Common Attributes for CuttingTool and CuttingToolArchetype	12
3.3	Common Elements for CuttingTool and CuttingToolArchetype	14
3.3.1	Description Element for CuttingTool and CuttingToolArchetype .	14
4	CuttingToolArchetype Information Model	15
4.1	Attributes for CuttingToolArchetype	19
4.2	Elements for CuttingToolArchetype	19
4.2.1	CuttingToolDefinition Element for CuttingToolArchetype	20
4.2.1.1	Attributes for CuttingToolDefinition	20
4.2.1.1.1	format Attribute for CuttingToolDefinition . .	20
4.2.1.2	Elements for CuttingToolDefinition	21
4.2.1.3	ISO13399 Standard	21
4.2.2	CuttingToolLifeCycle Element for CuttingToolArchetype	21
5	CuttingTool Information model	22
5.1	Attributes for CuttingTool	22
5.2	Elements for CuttingTool	22
5.2.1	CuttingToolLifeCycle Elements for CuttingTool Only	23
5.2.1.1	CutterStatus Element for CuttingToolLifeCycle	23
5.2.1.1.1	Status Element for CutterStatus	24
5.2.1.2	ToolLife Element for CuttingToolLifeCycle	26
5.2.1.2.1	Attributes for ToolLife	27
5.2.1.2.2	type Attribute for ToolLife	27
5.2.1.2.3	countDirection Attribute for ToolLife	28
5.2.1.3	Location Element for CuttingToolLifeCycle	28
5.2.1.3.1	Attributes for Location	29
5.2.1.3.2	type Attribute for Location	31
5.2.1.3.3	postiveOverlap Attribute for Location	31
5.2.1.3.4	negativeOverlap Attribute for Location	31
5.2.1.4	ReconditionCount Element for CuttingToolLifeCycle .	32
5.2.1.4.1	Attributes for ReconditionCount	32
5.2.2	CuttingToolArchetypeReference Element for Cutting Tool	33

5.2.2.1	source Attribute for CuttingToolArcheTypeReference	33
6	Common Entity CuttingToolLifeCycle	34
6.1	CuttingToolLifeCycle	34
6.1.1	XML Schema Structure for CuttingToolLifeCycle	34
6.2	Elements for CuttingToolLifeCycle	36
6.2.1	ProgramToolGroup Element for CuttingToolLifeCycle	37
6.2.2	ProgramToolNumber Element for CuttingToolLifeCycle	37
6.2.3	ProcessSpindleSpeed Element for CuttingToolLifeCycle	38
6.2.3.1	Attributes for ProcessSpindleSpeed	38
6.2.4	ProcessFeedRate Element for CuttingToolLifeCycle	39
6.2.4.1	Attributes for ProcessFeedRate	39
6.2.5	ConnectionCodeMachineSide Element for CuttingToolLifeCycle	40
6.2.6	xs:any Element for CuttingToolLifeCycle	40
6.2.7	Measurements Element for CuttingToolLifeCycle	40
6.2.8	Measurement	41
6.2.8.1	Attributes for Measurement	42
6.2.8.2	Measurement Subtypes for CuttingToolLifeCycle	43
6.2.9	CuttingItems Element for CuttingToolLifeCycle	47
6.2.9.1	Attributes for CuttingItems	48
6.2.10	CuttingItem	48
6.2.10.1	Attributes for CuttingItem	50
6.2.10.1.1	indices Attribute for CuttingItem	50
6.2.10.1.2	itemId Attribute for CuttingItem	50
6.2.10.1.3	manufacturers Attribute for CuttingItem	50
6.2.10.1.4	grade Attribute for CuttingItem	51
6.2.10.2	Elements for CuttingItem	51
6.2.10.2.1	Description Element for CuttingItem	51
6.2.10.2.2	Locus Element for CuttingItem	52
6.2.10.2.3	ItemLife Element for CuttingItem	53
6.2.10.2.4	Attributes for ItemLife	54
6.2.10.2.5	type Attribute for ItemLife	54
6.2.10.2.6	countDirection Attribute for ItemLife	55
6.2.10.3	Measurement Subtypes for CuttingItem	55
	Appendices	62
A	Bibliography	62
B	Additional Illustrations	64
C	Cutting Tool Example	68
C.1	Shell Mill	68
C.2	Step Drill	71
C.3	Shell Mill with Individual Loci	73

C.4	Drill with Individual Loci	75
C.5	Shell Mill with Different Inserts on First Row	77

Table of Figures

Figure 1: Cutting Tool Schema	11
Figure 2: Cutting Tool Parts	15
Figure 3: Cutting Tool Composition	16
Figure 4: Cutting Tool, Tool Item, and Cutting Item	17
Figure 5: Cutting Tool, Tool Item, and Cutting Item 2	17
Figure 6: Cutting Tool Measurements	18
Figure 7: Cutting Tool Asset Structure	18
Figure 8: CuttingToolDefinition Schema	20
Figure 9: CutterStatus Schema	23
Figure 10: ToolLife Schema	26
Figure 11: Location Schema	29
Figure 12: ReconditionCount Schema	32
Figure 13: CuttingToolArcheTypeReference Schema	33
Figure 14: CuttingToolLifeCycle Schema	35
Figure 15: ProcessSpindleSpeed Schema	38
Figure 16: ProcessFeedRate Schema	39
Figure 17: Measurement Schema	41
Figure 18: Cutting Tool Measurement Diagram 1	43
Figure 19: Cutting Tool Measurement Diagram 2	44
Figure 20: CuttingItems Schema	47
Figure 21: CuttingItem Schema	49
Figure 22: ItemLife Schema	53
Figure 23: Cutting Tool	56
Figure 24: Cutting Item	56
Figure 25: Cutting Item Measurement Diagram 3	57
Figure 26: Cutting Item Drive Angle	57
Figure 27: Cutting Tool Measurement Diagram 1 (Cutting Tool, Cutting Item, and Assembly Item – ISO 13399)	64
Figure 28: Cutting Tool Measurement Diagram 2 (Cutting Tool, Cutting Item, and Assembly Item – ISO 13399)	65
Figure 29: Cutting Tool Measurement Diagram 3 (Cutting Item – ISO 13399)	65
Figure 30: Cutting Tool Measurement Diagram 4 (Cutting Item – ISO 13399)	66
Figure 31: Cutting Tool Measurement Diagram 5 (Cutting Item – ISO 13399)	66
Figure 32: Cutting Tool Measurement Diagram 6 (Cutting Item – ISO 13399)	67
Figure 33: Shell Mill Side View	68
Figure 34: Indexable Insert Measurements	68
Figure 35: Step Mill Side View	71
Figure 36: Shell Mill with Explicate Loci	73
Figure 37: Step Drill with Explicate Loci	75
Figure 38: Shell Mill with Different Inserts on First Row	77

List of Tables

Table 1: Attributes for CuttingTool and CuttingToolArchetype	12
Table 2: Common Elements for CuttingTool and CuttingToolArchetype	14
Table 3: Elements for CuttingToolArchetype	19
Table 4: Attributes for CuttingToolDefinition	20
Table 5: Values for format attribute of CuttingToolDefinition	21
Table 6: Elements for CuttingTool	22
Table 7: Elements for CutterStatus	24
Table 8: Values for Status Element of CutterStatus	24
Table 9: Attributes for ToolLife	27
Table 10: Values for type of ToolLife	28
Table 11: Values for countDirection	28
Table 12: Attributes for Location	30
Table 13: Values for type of Location	31
Table 14: Attributes for ReconditionCount	32
Table 15: Attributes for CuttingToolArchetypeReference	33
Table 16: Elements for CuttingToolLifeCycle	36
Table 17: Attributes for ProcessSpindleSpeed	38
Table 18: Attributes for ProcessFeedRate	39
Table 19: Attributes for Measurement	42
Table 20: Measurement Subtypes for CuttingTool	44
Table 21: Attributes for CuttingItems	48
Table 22: Attributes for CuttingItem	50
Table 23: Elements for CuttingItem	51
Table 24: Attributes for ItemLife	54
Table 25: Values for type of ItemLife	55
Table 26: Values for countDirection	55
Table 27: Measurement Subtypes for CuttingItem	57

1 1 Purpose of This Document

2 This document, *MTConnect Standard: Part 4.1 - Cutting Tools* of the MTConnect Stan-
3 dard, 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 NMTOKEN

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: NMTOKEN.

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 Doc-*
40 *ument* 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 item, thing or entity that has potential or actual value to an organization *Ref:ISO*
44 *55000:2014(en)*

45 Note 1 to entry: Value can be tangible or intangible, financial or non-financial,
46 and includes consideration of risks and liabilities. It can be positive or negative
47 at different stages of the asset life.

48 Note 2 to entry: Physical assets usually refer to equipment, inventory and prop-
49 erties owned by the organization. Physical assets are the opposite of intangible
50 assets, which are non-physical assets such as leases, brands, digital assets, use
51 rights, licences, intellectual property rights, reputation or agreements.

52 Note 3 to entry: A grouping of assets referred to as an asset system could also
53 be considered as an asset.

54

55 ***Attribute***

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

57 Appears in the documents in the following form: *attribute*.

58 ***Child Element***

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

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

62 ***Component***

63 General meaning:

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

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

67 Used in *Information Models*:

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

- When used as an XML container to organize *Lower Level* Component elements.

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

- When used as an abstract XML element. *Component* is replaced in a data model by a type of *Component* element. *Component* is also an XML container used to organize *Lower Level* Component elements, *Data Entities*, or both.

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

Current Request

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

Data Entity

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

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

Devices Information Model

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

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

Equipment Metadata

See *Metadata*

Information Model

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

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

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

Lower Level

A nested element that is below a higher level element.

Metadata

Data that provides information about other data.

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

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

MTConnect Agent

See definition for *Agent*.

MTConnect Asset

An *MTConnect Asset* is an *Asset* used by the manufacturing process to perform tasks.

Note 1 to entry: An *MTConnect Asset* relies upon an *MTConnect Device* to provide *observations* and information about itself and the *MTConnect Device* revises the information to reflect changes to the *MTConnect Asset* during their interaction. Examples of *MTConnect Assets* are Cutting Tools, Part Information, Manufacturing Processes, Fixtures, and Files.

Note 2 to entry: A singular `assetId` uniquely identifies an *MTConnect Asset* throughout its lifecycle and is used to track and relate the *MTConnect Asset* to other *MTConnect Devices* and entities.

Note 3 to entry: *MTConnect Assets* are temporally associated with a device and can be removed from the device without damage or alteration to its primary functions.

MTConnect Device

An *MTConnect Device* is a piece of equipment or a manufacturing system that produces *observations* about itself and/or publishes data using the *MTConnect Information Model*.

MTConnect Information Model

See *Information Model*

MTConnectDevices Response Document

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

135 ***MTConnectStreams Response Document***

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

138 ***observation***

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

140 ***Observations Information Model***

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

143 ***Parent Element***

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

146 Appears in the documents in the following form: *Parent Element*.

147 ***Probe Request***

148 A *Probe Request* is a *Request* to an *Agent* to produce an *MTConnectDevices Re-*
149 *sponse Document* containing the *Devices Information Model*.

150 ***Request***

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

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

154 ***Response Document***

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

157 ***Sample Request***

158 A *Sample Request* is a *Request* to an *Agent* to produce an *MTConnectStreams Re-*
159 *sponse Document* containing the *Observations Information Model* for a set of time-
160 stamped *observations* made by *Components*.

161 ***semantic data model***

162 A methodology for defining the structure and meaning for data in a specific logical
163 way.

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

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

167 ***sequence number***

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

170 *sequence number* is a monotonically increasing number within an instance of an
171 *Agent*.

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

173 ***Spindle***

174 A mechanism that provides rotational capabilities to a piece of equipment.

175 Typically used for either work holding, materials or cutting tools.

176 ***Streaming Data***

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

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

180 ***Structural Element***

181 General meaning:

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

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

185 Used to indicate hierarchy of Components:

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

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

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

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

192 ***Top Level***

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

195 ***Valid Data Value***

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

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

199 ***XML Schema***

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

202 **2.2 Acronyms**

203 ***AMT***

204 The Association for Manufacturing Technology

205 **2.3 MTConnect References**

206 [MTConnect Part 1.0] *MTConnect Standard Part 1.0 - Overview and Fundamentals*. Ver-
207 sion 1.8.0.

208 [MTConnect Part 2.0] *MTConnect Standard: Part 2.0 - Devices Information Model*. Ver-
209 sion 1.8.0.

210 [MTConnect Part 3.0] *MTConnect Standard: Part 3.0 - Streams Information Model*. Ver-
211 sion 1.8.0.

212 [MTConnect Part 4.1] *MTConnect Standard: Part 4.1 - Cutting Tools*. Version 1.8.0.

213 3 Cutting Tool and Cutting Tool Archetype

214 There are two *Information Models* used to represent a cutting tool, `CuttingToolArchetype`
215 and `CuttingTool`. The `CuttingToolArchetype` represent the static cutting tool
216 geometries and nominal values as one would expect from a tool catalog and the `Cut-`
217 `tingTool` represents the use or application of the tool on the shop floor with actual
218 measured values and process data. In Version 1.3.0 of the MTConnect Standard it was de-
219 cided to separate out these two concerns since not all pieces of equipment will have access
220 to both sets of information. In this way, a generic definition of the cutting tool can coexist
221 with a specific assembly *Information Model* with minimal redundancy of data.

222 3.1 XML Schema Structure for CuttingTool and CuttingToolArchetype

223 The *Figure 1* shows the *XML Schema* that applies to both the `CuttingTool` *Information*
224 *Model* and the `CuttingToolArchetype` *Information Model*.

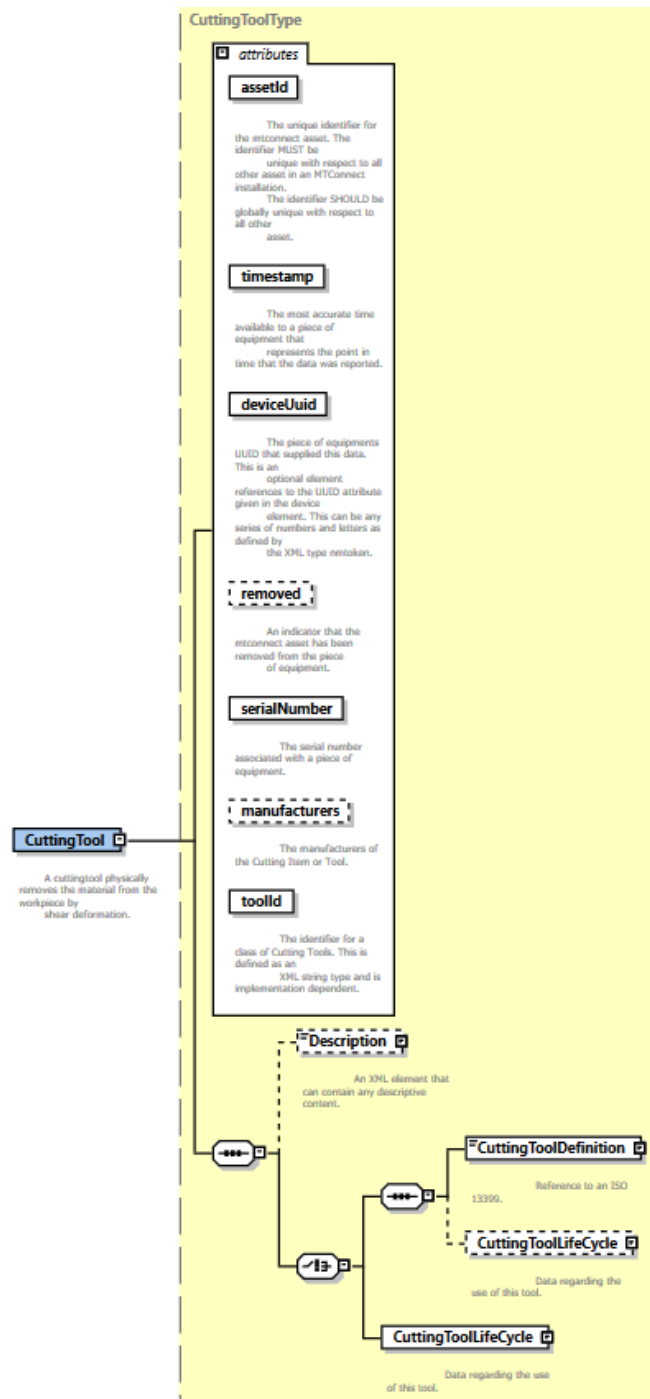


Figure 1: Cutting Tool Schema

225 Note: The use of the XML element `CuttingToolDefinition` has been **DEP-**
 226 **RECATED** in the `CuttingTool` schema, but remains in the `Cutting-`
 227 `ToolArchetype` schema.

228 The following sections contain the definitions of `CuttingTool` and `CuttingToolArchetype`
 229 and describe their unique components. The following are the common entities for both el-
 230 ements.

231 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	<p>The identifier for a class of Cutting Tools. This is defined as an XML string type and is implementation dependent.</p> <p>toolId is a required attribute.</p>	1
deviceUuid	<p>A reference to the Device's uuid that created the Asset information. The deviceUuid MUST be an NMTOKEN XML type.</p>	1
manufacturers	<p>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.</p>	0..1
removed	<p>This is an indicator that the Cutting Tool has been removed from the piece of equipment.</p> <p>removed is a required attribute.</p> <p>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.</p>	0..1

232 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

233 3.3.1 Description Element for CuttingTool and CuttingToolArchetype

234 Description **MAY** contain mixed content, meaning that an additional XML element
235 or plain text may be provided as part of the content of the description tag. Currently
236 Description contains no attributes.

237 4 CuttingToolArchetype Information Model

238 The CuttingToolArchetype *Information Model* will have the identical structure as
 239 the CuttingTool *Information Model* illustrated in *Figure 1* , except for a few entities.
 240 The CuttingTool will no longer carry the CuttingToolDefinition, this **MUST**
 241 only appear in the CuttingToolArchetype. The CuttingToolArchetype **MUST**
 242 **NOT** have measured values and **MUST NOT** have any of the following items: Cutter-
 243 Status, ToolLife values, Location, or a ReconditionCount.

244 MTConnect Standard will adopt the ISO 13399 structure when formulating the vocabulary
 245 for Cutting Tool geometries and structure to be represented in the CuttingToolArchetype.
 246 The nominal values provided in the CuttingToolLifeCycle section are only concerned
 247 with two aspects of the Cutting Tool, the Cutting Tool and the Cutting Item. The
 248 Tool Item, Adaptive Item, and Assembly Item will only be covered in the Cutting-
 249 ToolDefinition section of this document since this section contains the full ISO
 250 13399 information about a Cutting Tool.



Figure 2: Cutting Tool Parts

251 The *Figure 2* illustrates the parts of a Cutting Tool. The Cutting Tool is the aggregate of
 252 all the components and the Cutting Item is the part of the tool that removes the material
 253 from the workpiece. These are the primary focus of the MTConnect Standard.

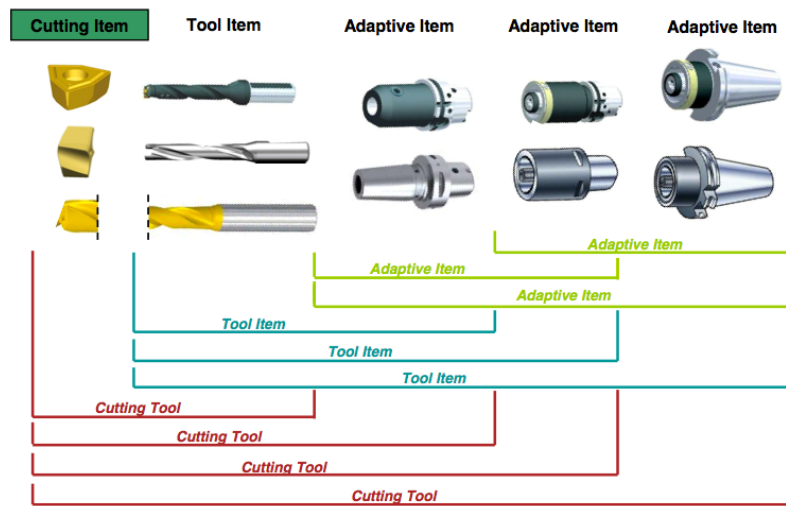


Figure 3: Cutting Tool Composition

254 *Figure 3* provides another view of the composition of a Cutting Tool. The Adaptive Items
 255 and Tool Items will be used for measurements, but will not be modeled as separate entities.
 256 When we are referencing the Cutting Tool we are referring to the entirety of the assembly
 257 and when we provide data regarding the Cutting Item we are referencing each individual
 258 item as illustrated on the left of the previous diagram.

259 *Figure 4* and *Figure 5* further illustrates the components of the Cutting Tool. As we
 260 compose the Tool Item, Cutting Item, Adaptive Item, we get a Cutting Tool. The Tool Item,
 261 Adaptive Item, and Assembly Item will only be in the `CuttingToolDefinition`
 262 section that will contain the full ISO 13399 information.

Reference ISO13399

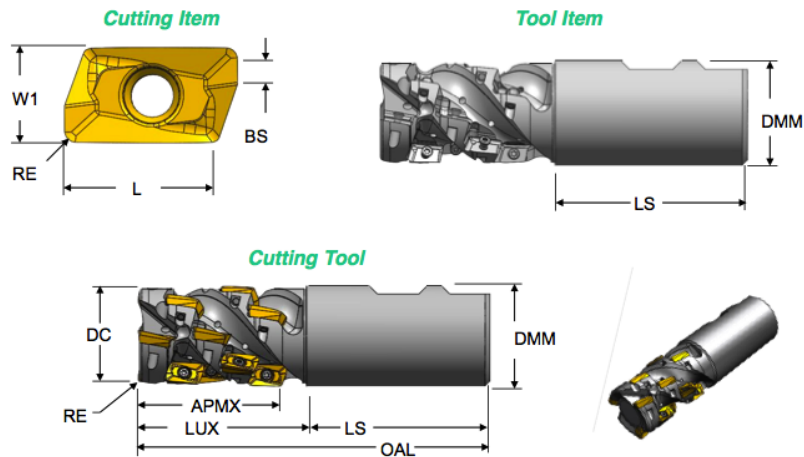


Figure 4: Cutting Tool, Tool Item, and Cutting Item

Reference ISO13399

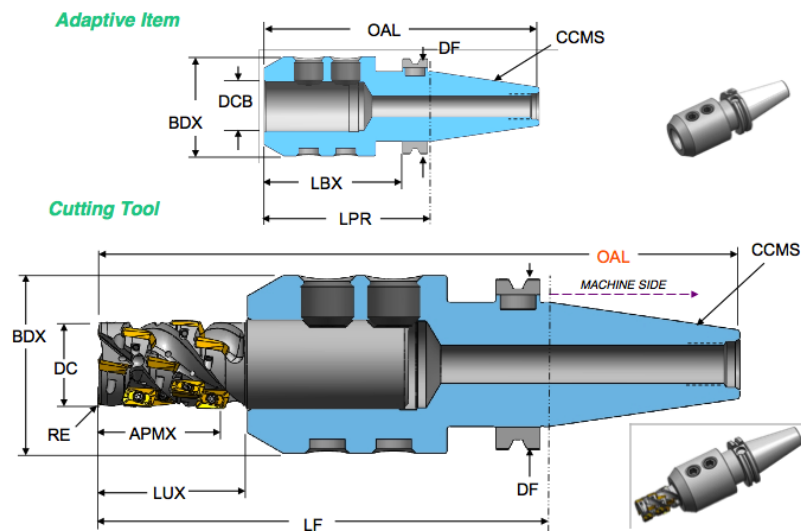


Figure 5: Cutting Tool, Tool Item, and Cutting Item 2

263 *Figure 4* and *Figure 5* use the ISO 13399 codes for each of the measurements. These
 264 codes will be translated into the MTConnect Standard vocabulary as illustrated below.
 265 The measurements will have a maximum, minimum, and nominal value representing the
 266 tolerance of allowable values for this dimension. See below for a full discussion.

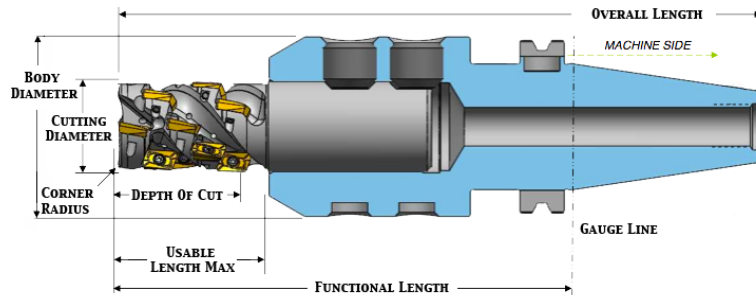


Figure 6: Cutting Tool Measurements

267 The MTConnect Standard will not define the entire geometry of the Cutting Tool, but will
 268 provide the information necessary to use the tool in the manufacturing process. Addi-
 269 tional information can be added to the definition of the Cutting Tool by means of schema
 270 extensions.

271 Additional diagrams will reference these dimensions by their codes that will be defined in
 272 the measurement tables. The codes are consistent with the codes used in ISO 13399 and
 273 have been standardized. MTConnect Standard will use the full text name for clarity in the
 274 XML document.

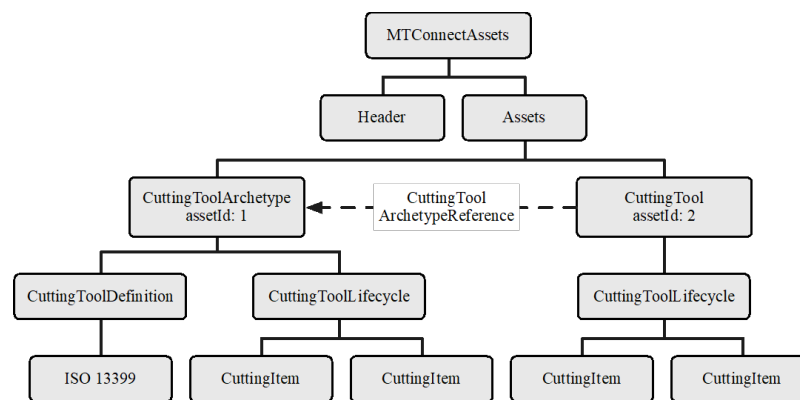


Figure 7: Cutting Tool Asset Structure

275 The structure of the MTConnectAssets header is defined in *MTConnect Standard Part*
 276 *1.0 - Overview and Fundamentals* of the Standard. A finite number of *MTConnect Assets*
 277 will be stored in the *Agent*. This finite number is implementation specific and will depend
 278 on memory and storage constraints. The standard will not prescribe the number or capacity
 279 requirements for an implementation.

280 4.1 Attributes for CuttingToolArchetype

281 Refer to *Section 3.2 - Common Attributes for CuttingTool and CuttingToolArchetype* for a
 282 full description of the attributes for CuttingToolArchetype *Information Model*.

283 4.2 Elements for CuttingToolArchetype

284 The elements associated with CuttingToolArchetype are given in *Table 3*. Each
 285 element will be described in more detail below and any possible values will be presented
 286 with full definitions. The elements **MUST** be provided in the following order as prescribed
 287 by XML. At least one of CuttingToolDefinition or CuttingToolLifeCycle
 288 **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

289 4.2.1 CuttingToolDefinition Element for CuttingToolArchetype

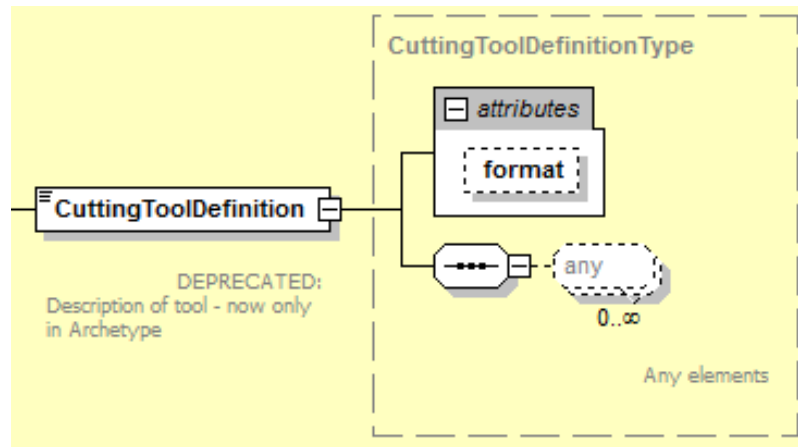


Figure 8: CuttingToolDefinition Schema

290 The `CuttingToolDefinition` contains the detailed structure of the Cutting Tool.
 291 The information contained in this element will be static during its lifecycle. Currently we
 292 are referring to the external ISO 13399 standard to provide the complete definition and
 293 composition of the Cutting Tool as defined in *Section 6.1 - CuttingToolLifeCycle*.

294 4.2.1.1 Attributes for CuttingToolDefinition

Table 4: Attributes for CuttingToolDefinition

Attribute	Description	Occurrence
<code>format</code>	<p>Identifies the expected representation of the enclosed data.</p> <p><code>format</code> is an optional attribute.</p> <p>Valid values of <code>format</code> are – XML, EXPRESS, TEXT, or UNDEFINED.</p> <p>If <code>format</code> is not specified, the assumed format is XML.</p>	0..1

295 4.2.1.1.1 `format` Attribute for CuttingToolDefinition

296 The `format` attribute describes the expected representation of the enclosed data. If no
 297 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 confirm 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.

298 4.2.1.2 Elements for CuttingToolDefinition

299 The only acceptable Cutting Tool definition at present is defined by the ISO 13399 stan-
300 dard. Additional formats **MAY** be considered in the future.

301 4.2.1.3 ISO13399 Standard

302 The ISO 13399 data **MUST** be presented in either XML (ISO 10303-28) or EXPRESS
303 format (ISO 10303-21). An *XML Schema* will be preferred as this will allow for easier
304 integration with the MTConnect Standard XML tools. EXPRESS will also be supported,
305 but software tools will need to be provided or made available for handling this data repre-
306 sentation.

307 There will be the root element of the ISO13399 document when XML is used. When
308 EXPRESS is used the XML element will be replaced by the text representation.

309 4.2.2 CuttingToolLifeCycle Element for CuttingToolArchetype

310 Refer to *Section 6 - Common Entity CuttingToolLifeCycle* for a complete description of
311 CuttingToolLifeCycle element.

312 5 CuttingTool Information model

313 The CuttingTool *Information Model* illustrated in *Figure 1* has the identical struc-
 314 ture as the CuttingToolArchetype *Information Model* except for the XML ele-
 315 ment CuttingToolDefinition that has been **DEPRECATED** in the Cutting-
 316 Tool schema.

317 5.1 Attributes for CuttingTool

318 Refer to *Section 3.2 - Common Attributes for CuttingTool and CuttingToolArchetype* for a
 319 full description of the *Attributes for CuttingTool Information Model*.

320 5.2 Elements for CuttingTool

321 The elements associated with CuttingTool are given below. The elements **MUST** be
 322 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 <code>assetId</code> of the <code>CuttingToolArchetype</code> document. It MAY also contain a <code>source</code> attribute that gives the URL of the archetype data as well.	0..1

323 5.2.1 CuttingToolLifeCycle Elements for CuttingTool Only

324 The following `CuttingToolLifeCycle` elements are used only in the `Cutting-`
325 `Tool Information Model` and are not part of the `CuttingToolArchetype Informa-`
326 `tion Model`. Refer to *Section 6 - Common Entity CuttingToolLifeCycle* for a complete
327 description of the remaining elements for `CuttingToolLifeCycle` that are common
328 in both *Information Models*. Refer also to the `CuttingToolLifeCycle` schema illus-
329 trated in *Figure 14*.

330 5.2.1.1 CutterStatus Element for CuttingToolLifeCycle

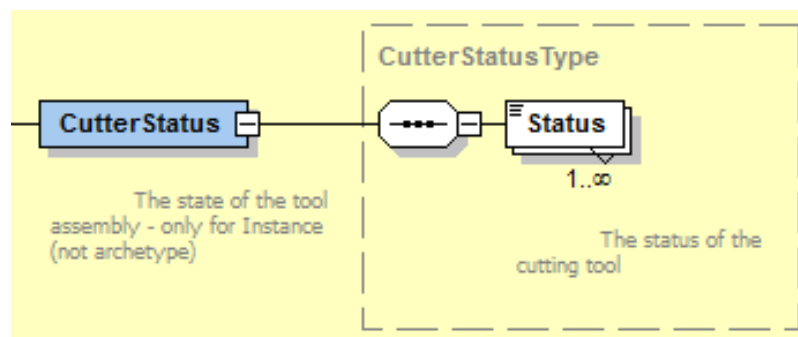


Figure 9: CutterStatus Schema

331 The elements of the `CutterStatus` element can be a combined set of `Status` ele-
332 ments. The *MTConnect Standard* allows any set of statuses to be combined, but only
333 certain combinations make sense. A `CuttingTool` **SHOULD** not be both `NEW` and

334 USED at the same time. There are no rules in the schema to enforce this, but this is left to
 335 the implementer. The following combinations **MUST NOT** occur:

- 336 • NEW **MUST NOT** be used with USED, RECONDITIONED, or EXPIRED.
- 337 • UNKNOWN **MUST NOT** be used with any other status.
- 338 • ALLOCATED and UNALLOCATED **MUST NOT** be used together.
- 339 • AVAILABLE and UNAVAILABLE **MUST NOT** be used together.
- 340 • If the tool is EXPIRED, BROKEN, or NOT_REGISTERED it **MUST NOT** be AVAIL-
 341 ABLE.
- 342 • 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..*

343 5.2.1.1.1 Status Element for CutterStatus

344 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 <code>ReconditionCount</code> 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.

345 5.2.1.2 ToolLife Element for CuttingToolLifeCycle

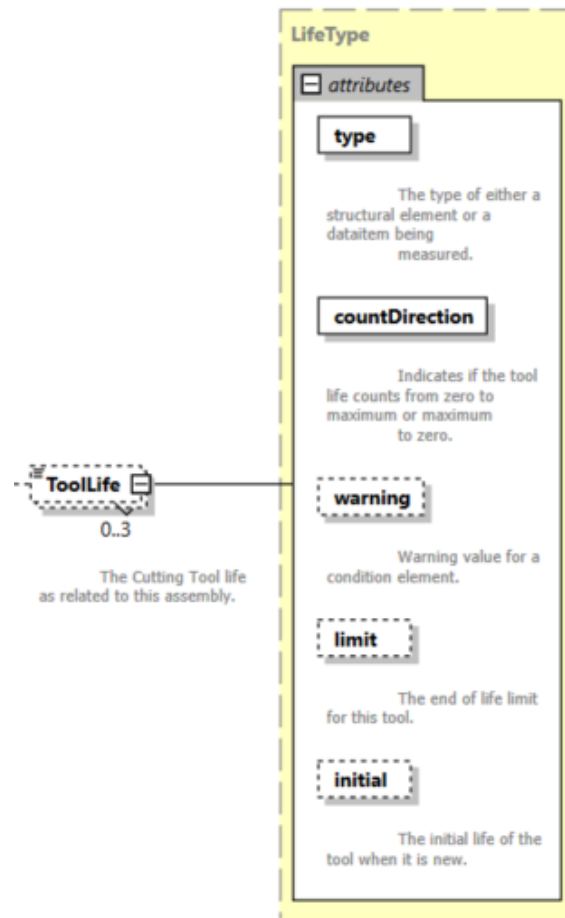


Figure 10: ToolLife Schema

346 The value is the current value for the ToolLife. The value **MUST** be numeric. Tool-
 347 Life is an option element which can have three types, either minutes for time based, part
 348 count for parts based, or wear based using a distance measure. One ToolLife element
 349 can appear for each type, but there cannot be two entries of the same type. Additional
 350 types can be added in the future.

351 **5.2.1.2.1 Attributes for ToolLife**

352 ToolLife has the following attributes that can be used to indicate the behavior of the
 353 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

354 **5.2.1.2.2 type Attribute for ToolLife**

355 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.

356 5.2.1.2.3 countDirection Attribute for ToolLife

357 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.

358 5.2.1.3 Location Element for CuttingToolLifeCycle

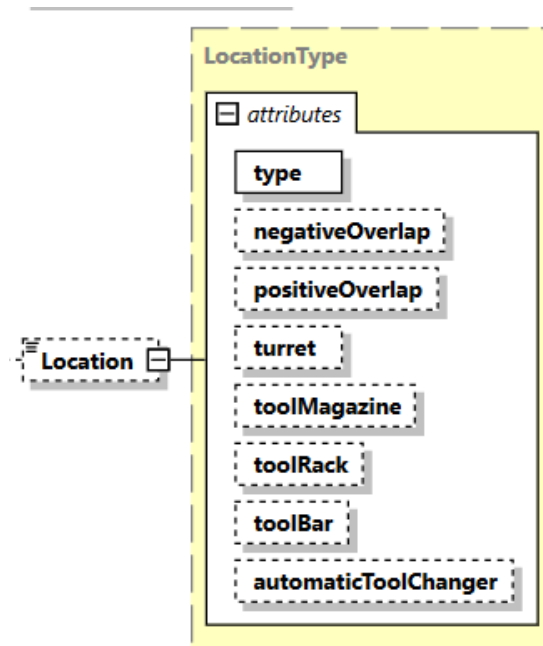


Figure 11: Location Schema

Location element identifies the specific location where a tool resides in a piece of equipment tool storage or in a tool crib. This can be any series of numbers and letters as defined by the XML type NMToken. When a POT or STATION type is used, the value **MUST** be a numeric value. If a negativeOverlap or the positiveOverlap is provided, the tool reserves additional locations on either side, otherwise if they are not given, no additional locations are required for this tool. If the pot occupies the first or last location, a rollover to the beginning or the end of the index-able values may occur. For example, if there are 64 pots and the tool is in pot 64 with a positiveOverlap of 1, the first pot **MAY** be occupied as well.

5.2.1.3.1 Attributes for Location

Table 12: Attributes for Location

Attribute	Description	Occurrence
type	<p>The type of location being identified.</p> <p>type MUST be one of POT, STATION, CRIB, SPINDLE, TRANSFER_POT, RETURN_POT, STAGING_POT, REMOVAL_POT, EXPIRED_POT, or END_EFFECTOR.</p> <p>type is a required attribute.</p>	1
positiveOverlap	<p>The number of locations at higher index value from this location.</p> <p>positiveOverlap is a optional attribute.</p>	0..1
negativeOverlap	<p>The number of location at lower index values from this location.</p> <p>negativeOverlap is an optional attribute.</p>	0..1
turret	<p>The turret associated with a tool.</p> <p>turret MUST be an XML NMTOKEN type.</p>	0..1
toolMagazine	<p>The tool magazine associated with a tool.</p> <p>toolMagazine MUST be an XML NMTOKEN type.</p>	0..1
toolBar	<p>The tool bar associated with a tool.</p> <p>toolBar MUST be an XML NMTOKEN type.</p>	0..1
toolRack	<p>The tool rack associated with a tool.</p> <p>toolRack MUST be an XML NMTOKEN type.</p>	0..1
automaticToolChanger	<p>The automatic tool changer associated with a tool.</p> <p>automaticToolChanger MUST be an XML NMTOKEN type.</p>	0..1

369 5.2.1.3.2 type Attribute for Location

370 The type of location being identified.

Table 13: Values for type of Location

Value	Description
POT	A location in a tool magazine.
STATION	A location in a turret, tool bar, or tool rack.
CRIB	A location within a tool crib.
SPINDLE	A location associated with a <i>Spindle</i> .
TRANSFER_POT	A location for a tool awaiting transfer from a tool magazine to spindle or a turret.
RETURN_POT	A location for a tool removed from a <i>Spindle</i> or turret and awaiting return to a tool magazine.
STAGING_POT	A location for a tool awaiting transfer to a tool magazine or turret from outside of the piece of equipment.
REMOVAL_POT	A location for a tool removed from a tool magazine or turret awaiting transfer to a location outside of the piece of equipment.
EXPIRED_POT	A location for a tool that is no longer useable and is awaiting removal from a tool magazine or turret.
END_EFFECTOR	A location associated with an end effector.

371 5.2.1.3.3 positiveOverlap Attribute for Location

372 The number of locations at higher index values that the `CuttingTool` occupies due to
 373 interference. The value **MUST** be an integer. If not provided it is assumed to be 0.

374 5.2.1.3.4 negativeOverlap Attribute for Location

375 The number of locations at lower index values that the `CuttingTool` occupies due to
 376 interference. The value **MUST** be an integer. If not provided it is not assumed to be 0.

377 The tool number assigned in the part program and is used for cross referencing this tool
 378 information with the process parameters. The value **MUST** be an integer.

379 5.2.1.4 ReconditionCount Element for CuttingToolLifeCycle

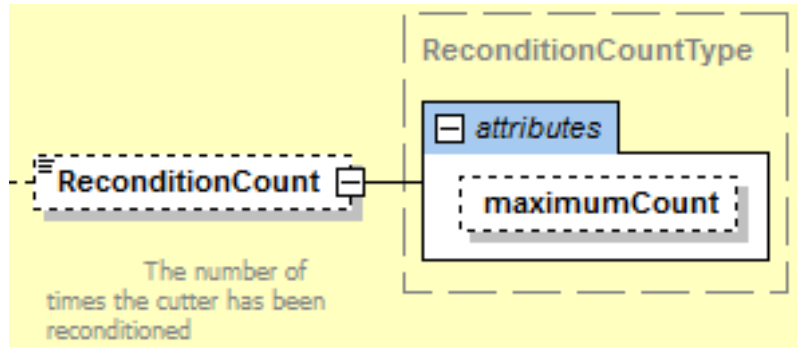


Figure 12: ReconditionCount Schema

380 This element **MUST** contain an integer value as the CDATA that represents the number of
 381 times the cutter has been reconditioned.

382 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

383 5.2.2 CuttingToolArchetypeReference Element for Cutting Tool

384



Figure 13: CuttingToolArcheTypeReference Schema

385 This optional element references another *MTConnect Asset* document providing the static
 386 geometries and nominal values for all the measurements. This reduces the amount of data
 387 duplication as well as providing a mechanism for asset definitions to be provided before
 388 complete measurement has occurred.

389 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

390 6 Common Entity CuttingToolLifeCycle

391 6.1 CuttingToolLifeCycle

392 The life cycle refers to the data pertaining to the application or the use of the tool. This
 393 data is provided by various pieces of equipment (i.e. machine tool, presetter) and statis-
 394 tical process control applications. Life cycle data will not remain static, but will change
 395 periodically when a tool is used or measured. The life cycle has three conceptual parts;
 396 CuttingTool and CuttingItem identity, properties, and measurements. A measure-
 397 ment is defined as a constrained value that is reported in defined units and as a W3C
 398 floating point format.

399 The CuttingToolLifeCycle contains data for the entire tool assembly. The specific
 400 CuttingItems that are part of the CuttingToolLifeCycle are contained in the
 401 CuttingItems element. Each Cutting Item has similar properties as the assembly;
 402 identity, properties, and Measurements.

403 The units for all Measurements have been predefined in the *MTConnect Standard* and
 404 will be consistent with *MTConnect Standard: Part 2.0 - Devices Information Model* and
 405 *MTConnect Standard: Part 3.0 - Streams Information Model*. This means that all lengths
 406 and distances will be given in millimeters and all angular measures will be given in de-
 407 grees. Quantities like ProcessSpindleSpeed will be given in RPM, the same as the
 408 ROTARY_VELOCITY in *MTConnect Standard: Part 3.0 - Streams Information Model*.

409 6.1.1 XML Schema Structure for CuttingToolLifeCycle

410 The CuttingToolLifeCycle schema shown in *Figure 14* is used in both the Cut-
 411 tingToolArchetype and CuttingTool *Information Models*. The only difference
 412 is that the elements CutterStatus, ToolLife, Location, and Recondition-
 413 Count are used only in the CuttingTool *Information Model*.

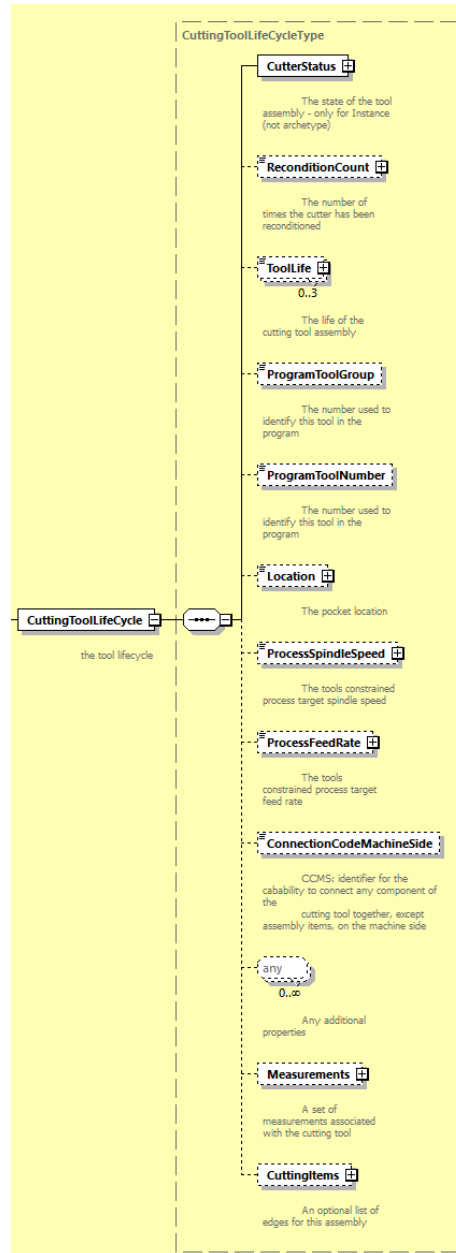


Figure 14: CuttingToolLifeCycle Schema

414 6.2 Elements for CuttingToolLifeCycle

415 The elements associated with this Cutting Tool are given in *Table 16*. The elements **MUST**
 416 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 CuttingTool <i>Information Model</i>.</p>	1
ReconditionCount	<p>The number of times this cutter has been reconditioned.</p> <p>MUST only be used in the CuttingTool <i>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 CuttingTool <i>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 CuttingTool <i>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

417 6.2.1 ProgramToolGroup Element for CuttingToolLifeCycle

418 The optional identifier for the group of Cutting Tools when multiple tools can be used
 419 interchangeably. This is defined as an XML string type and is implementation dependent.

420 6.2.2 ProgramToolNumber Element for CuttingToolLifeCycle

421 The tool number assigned in the part program and is used for cross referencing this tool
 422 information with the process parameters. The value **MUST** be a string.

423 6.2.3 ProcessSpindleSpeed Element for CuttingToolLifeCycle

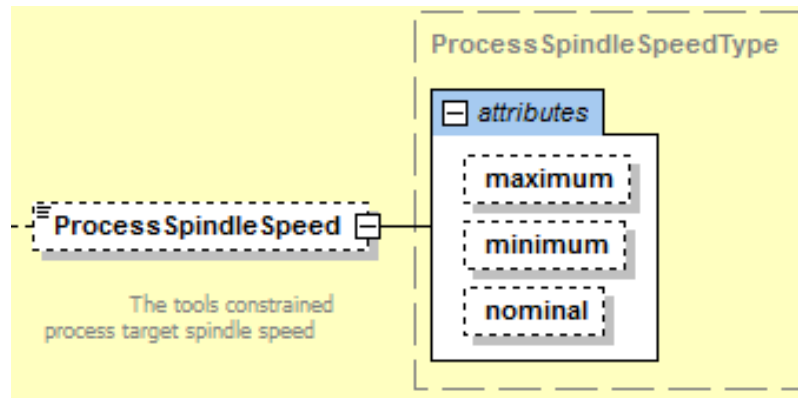


Figure 15: ProcessSpindleSpeed Schema

424 The ProcessSpindleSpeed **MUST** be specified in revolutions/minute (RPM). The
 425 CDATA **MAY** contain the nominal process target spindle speed if available. The maximum
 426 and minimum speeds **MAY** be provided as attributes. If ProcessSpindleSpeed is
 427 provided, at least one value of maximum, nominal, or minimum **MUST** be specified.

428 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

429 6.2.4 ProcessFeedRate Element for CuttingToolLifeCycle

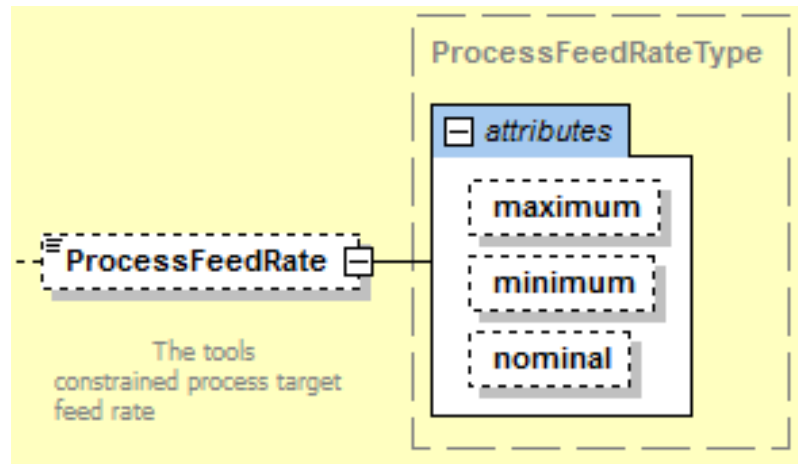


Figure 16: ProcessFeedRate Schema

430 The `ProcessFeedRate` **MUST** be specified in millimeters/second (mm/s). The CDATA
 431 **MAY** contain the nominal process target feed rate if available. The maximum and mini-
 432 mum rates **MAY** be provided as attributes. If `ProcessFeedRate` is provided, at least
 433 one value of maximum, nominal, or minimum **MUST** be specified.

434 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

435 6.2.5 ConnectionCodeMachineSide Element for CuttingToolLifeCy- 436 cle

437 This is an optional identifier for implementation specific connection component of the
438 Cutting Tool on the machine side. Code: CCMS. The CDATA **MAY** be any valid string
439 according to the referenced connection code standards.

440 6.2.6 xs:any Element for CuttingToolLifeCycle

441 Utilizing *XML Schema* 1.1, extension points are available where an additional element
442 can be added to the document without being part of a substitution group. The new ele-
443 ments **MUST NOT** be part of the *MTConnect namespace* and **MUST NOT** be one of the
444 predefined elements mentioned above.

445 This allows additional properties to be defined for `CuttingTool` without having to
446 change the definition of the definition of the `CuttingTool` or modify the standard, but
447 requires *XML Schema* Version 1.1.

448 6.2.7 Measurements Element for CuttingToolLifeCycle

449 The `Measurements` element is a collection of one or more constrained scalar values
450 associated with this Cutting Tool. The XML element **MUST** be a type extension of the
451 base types `CommonMeasurement` or `AssemblyMeasurement`. The following sec-
452 tion defines the abstract `Measurement` type used in both `CuttingToolLifeCycle`
453 and `CuttingItem`. This subsequent sections describe the `AssemblyMeasurement`
454 types followed by the `CuttingItemMeasurement` types.

455 A `Measurement` is specific to the tool management policy at a particular shop. The tool
456 zero reference point or gauge line will be different depending on the particular implemen-
457 tation and will be assumed to be consistent within the shop. *MTConnect Standard* does
458 not standardize the manufacturing process or the definition of the zero point.

459 6.2.8 Measurement

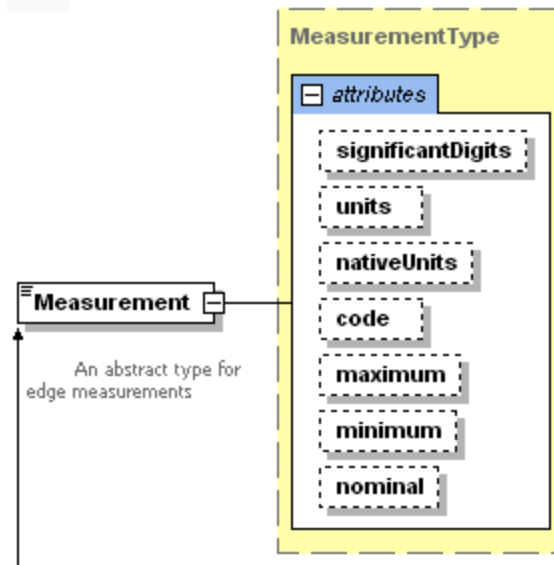


Figure 17: Measurement Schema

460 A Measurement **MUST** be a scalar floating-point value that **MAY** be constrained to a
 461 maximum and minimum value. Since the `CuttingToolLifeCycle`'s main responsi-
 462 bility is to track aspects of the tool that change over its use in the shop, *MTConnect* repre-
 463 sents the current value of the Measurement **MUST** be in the CDATA (text between the
 464 start and end element) as the most current valid value.

465 The minimum and maximum **MAY** be supplied if they are known or relevant to the
 466 Measurement. A nominal value **MAY** be provided to show the reference value for
 467 this Measurement.

468 There are three abstract subtypes of Measurement: `CommonMeasurement`, `Assem-`
 469 `blyMeasurement`, and `CuttingItemMeasurement`. These abstract types **MUST**
 470 **NOT** appear in an *MTConnectAssets* document, but are used in the schema as a way
 471 to separate which measurements **MAY** appear in the different sections of the document.
 472 Only subtypes that have extended these types **MAY** appear in the *MTConnectAssets*
 473 XML.

474 Measurements in the `CuttingToolLifeCycle` section **MUST** refer to the en-
 475 tire assembly and not to an individual `CuttingItem`. `CuttingItem` measurements
 476 **MUST** be located in the measurements associated with the individual `CuttingItem`.

477 Measurements **MAY** provide an optional `units` attribute to reinforce the given units.
 478 The units **MUST** always be given in the predefined *MTConnect* units. If `units` are

479 provided, they are only for documentation purposes. `nativeUnits` **MAY** optionally be
 480 provided to indicate the original units provided for the measurements.

481 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 MTConnect <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

6.2.8.2 Measurement Subtypes for CuttingToolLifeCycle

These Measurements for CuttingTool are specific to the entire assembly and **MUST NOT** be used for the Measurement pertaining to a CuttingItem. *Figure 18* and *Figure 19* will be used to reference the assembly specific Measurements.

The Code in *Table 20* will refer to the acronyms in the diagrams. We will be referring to many diagrams to disambiguate all measurements of the CuttingTool and CuttingItem.

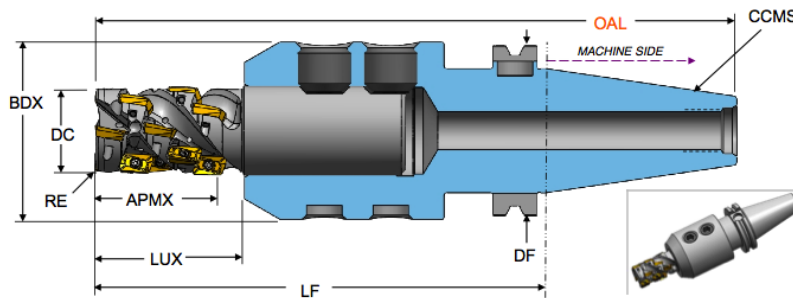


Figure 18: Cutting Tool Measurement Diagram 1

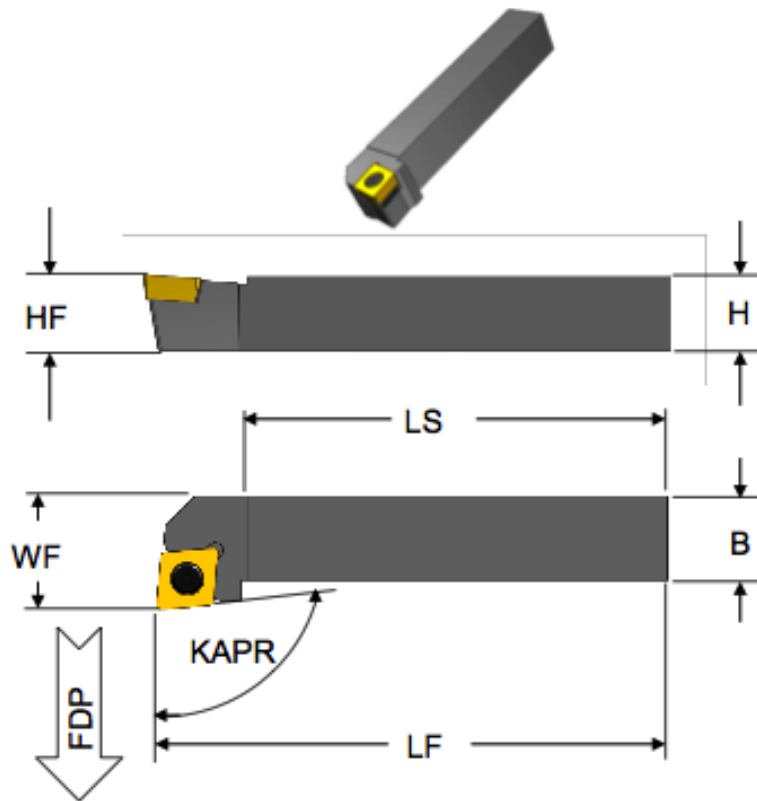


Figure 19: Cutting Tool Measurement Diagram 2

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 <code>CuttingTool</code> functional length will be the length of the entire tool, not a single <code>CuttingItem</code> . Each <code>CuttingItem</code> can have an independent <code>FunctionalLength</code> represented in its measurements.	MILLIMETER

489 6.2.9 CuttingItems Element for CuttingToolLifeCycle

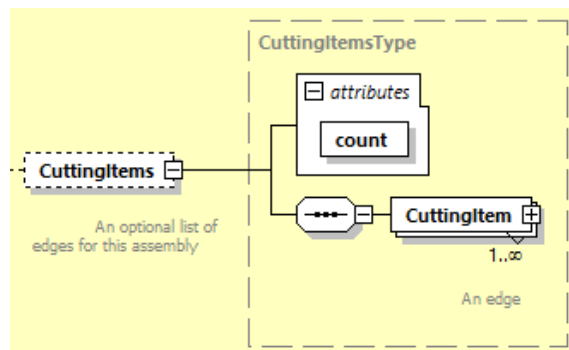


Figure 20: CuttingItems Schema

490 An optional collection of `CuttingItems` that **SHOULD** be provided for each indepen-
 491 dent edge or insert. If the `CuttingItems` are not present; it indicates there is no specific
 492 information with respect to each of the `CuttingItems`. This does not imply there are no
 493 `CuttingItems` – there **MUST** be at least one `CuttingItem` – but there is no specific
 494 information.

495 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

496 6.2.10 CuttingItem

497 A CuttingItem is the portion of the tool that physically removes the material from the
 498 workpiece by shear deformation. The Cutting Item can be either a single piece of mate-
 499 rial attached to the CuttingItem or it can be one or more separate pieces of material
 500 attached to the CuttingItem using a permanent or removable attachment. A Cut-
 501 tingItem can be comprised of one or more cutting edges. CuttingItems include:
 502 replaceable inserts, brazed tips and the cutting portions of solid CuttingTools.

503 MTConnect Standard considers CuttingItems as part of the CuttingTool. A Cut-
 504 tingItems **MUST NOT** exist in MTConnect unless it is attached to a CuttingTool.
 505 Some of the measurements, such as FunctionalLength, **MUST** be made with refer-
 506 ence to the entire CuttingTool to be meaningful.

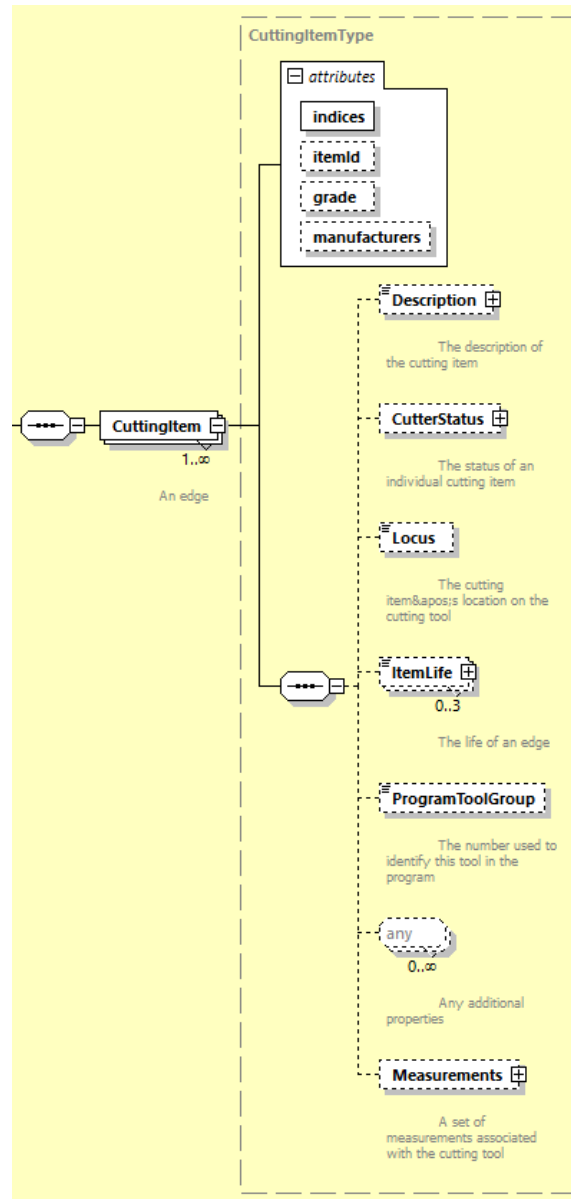


Figure 21: CuttingItem Schema

507 **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

508 **6.2.10.1.1 indices Attribute for CuttingItem**

509 An identifier that indicates the CuttingItem or CuttingItems these data are as-
510 sociated with. The value **MUST** be a single number ("1") or a comma separated set of
511 individual elements ("1,2,3,4"), or as a inclusive range of values as in ("1-10") or any
512 combination of ranges and numbers as in "1-4,6-10,22". There **MUST NOT** be spaces or
513 non-integer values in the text representation.

514 Indices **SHOULD** start numbering with the inserts or CuttingItem furthest from the
515 gauge line and increasing in value as the items get closer to the gauge line. Items at the
516 same distance **MAY** be arbitrarily numbered.

517 **6.2.10.1.2 itemId Attribute for CuttingItem**

518 The manufactures' identifier for this CuttingItem that **MAY** be its catalog or reference
519 number. The value **MUST** be an XML NMTOKEN value of numbers and letters.

520 **6.2.10.1.3 manufacturers Attribute for CuttingItem**

521 This optional element references the manufacturers of this tool. At this level the manufac-

522 turers will reference the `CuttingItem` specifically. The representation will be a comma
 523 (,) delimited list of manufacturer names. This can be any series of numbers and letters as
 524 defined by the XML type `string`.

525 **6.2.10.1.4 grade Attribute for CuttingItem**

526 This provides an implementation specific designation for the material composition of this
 527 `CuttingItem`.

528 **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
CutterStatus	The status of this item. CutterStatus MUST one of the following values: NEW, AVAILABLE, UNAVAILABLE, ALLOCATED, UNALLOCATED, MEASURED, RECONDITIONED, NOT_REGISTERED, USED, EXPIRED, BROKEN, or UNKNOWN.	0..1
ProgramToolGroup	The tool group the part program assigned this item.	0..1

529 **6.2.10.2.1 Description Element for CuttingItem**

530 An optional free form text description of this `CuttingItem`.

531 6.2.10.2.2 Locus Element for CuttingItem

532 Locus represents the location of the CuttingItem with respect to the Cutting Tool.
 533 For clarity, the words FLUTE, INSERT, and CARTRIDGE **SHOULD** be used to assist in
 534 noting the location of a CuttingItem. The Locus **MAY** be any free form text, but
 535 **SHOULD** adhere to the following rules:

- 536 • The location numbering **SHOULD** start at the furthest CuttingItem (#1) and
 537 work it's way back to the Cutting Item closest to the gauge line.
- 538 • Flutes **SHOULD** be identified as such using the word FLUTE:. For example: FLUTE:
 539 1, INSERT: 2 - would indicate the first flute and the second furthest insert from the
 540 end of the tool on that flute.
- 541 • Other designations such as CARTRIDGE **MAY** be included, but should be identified
 542 using upper case and followed by a colon (:).

543 6.2.10.2.3 ItemLife Element for CuttingItem

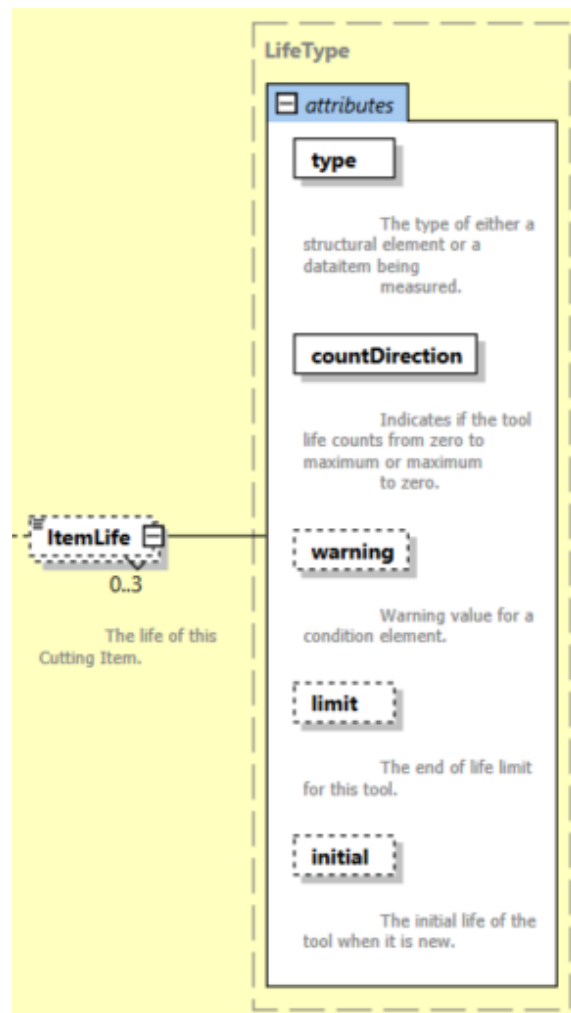


Figure 22: ItemLife Schema

544 The value is the current value for the ItemLife. The value **MUST** be numeric. Item-
 545 Life is an option element which can have three types, either minutes for time based, part
 546 count for parts based, or wear based using a distance measure. One ItemLife can ap-
 547 pear for each type, but there cannot be two entries of the same type. Additional types can
 548 be added in the future.

549 6.2.10.2.4 Attributes for ItemLife

550 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

551 6.2.10.2.5 type Attribute for ItemLife

552 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.

553 6.2.10.2.6 countDirection Attribute for ItemLife

554 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.

555 6.2.10.3 Measurement Subtypes for CuttingItem

556 These Measurements for CuttingItem are specific to an individual CuttingItem
 557 and **MUST NOT** be used for the Measurements pertaining to an assembly. The *Fig-*
 558 *ure 23* , *Figure 24* , *Figure 25* and *Figure 26* will be used to for reference for the Cut-
 559 tingItem specific Measurements .

560 The Code in *Table 27* will refer to the acronym in the diagram. We will be referring to
 561 many diagrams to disambiguate all Measurements of the CuttingTools and Cut-
 562 tingItems. We will present a few here; please refer to Appendix B for additional
 563 reference material.

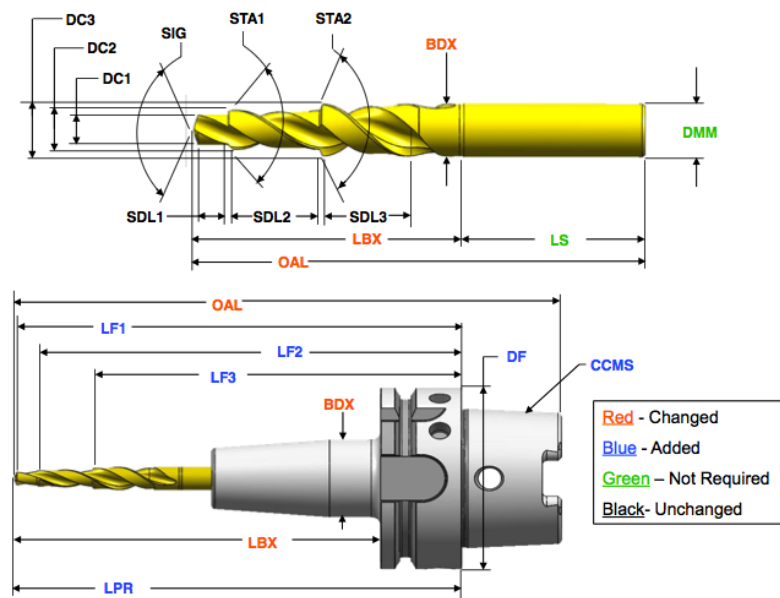


Figure 23: Cutting Tool

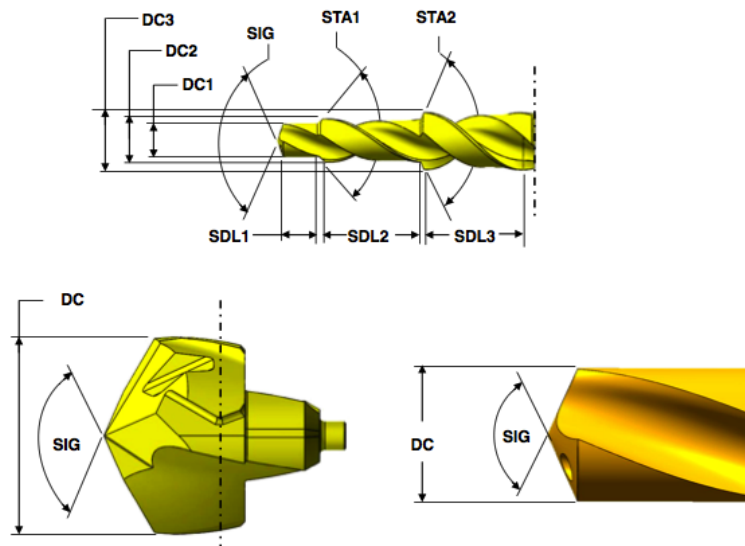


Figure 24: Cutting Item

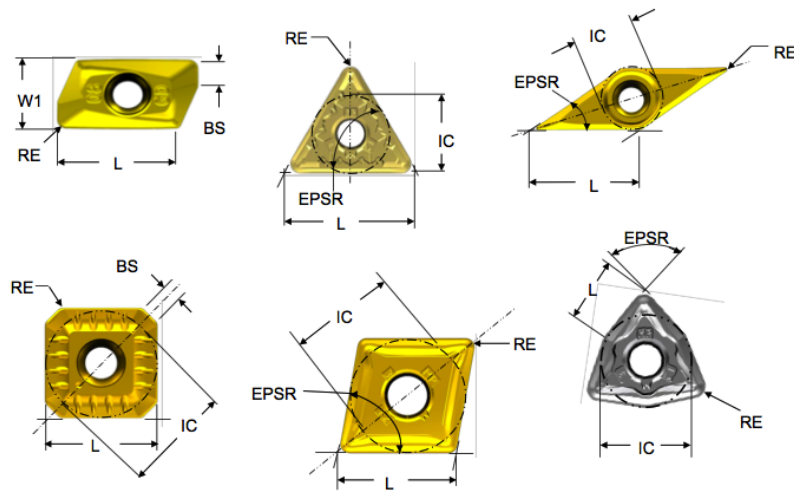


Figure 25: Cutting Item Measurement Diagram 3

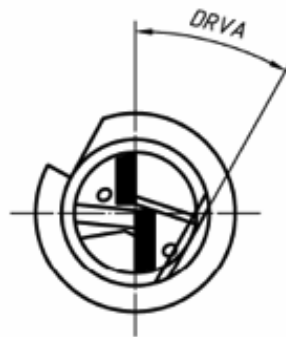


Figure 26: Cutting Item Drive Angle

564 The CuttingItem Measurements in Table 27 will refer the Figure 23 , Figure 24 ,
565 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
IncribedCircleDiameter	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

566 Appendices

567 A Bibliography

- 568 Engineering Industries Association. *EIA Standard - EIA-274-D*, Interchangeable Variable,
569 Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically
570 Controlled Machines. Washington, D.C. 1979.
- 571 ISO TC 184/SC4/WG3 N1089. *ISO/DIS 10303-238*: Industrial automation systems and
572 integration Product data representation and exchange Part 238: Application Protocols: Ap-
573 plication interpreted model for computerized numerical controllers. Geneva, Switzerland,
574 2004.
- 575 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
576 tems and integration – Physical device control – Data model for computerized numerical
577 controllers – Part 10: General process data. Geneva, Switzerland, 2004.
- 578 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
579 tems and integration – Physical device control – Data model for computerized numerical
580 controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.
- 581 International Organization for Standardization. *ISO 6983/1* – Numerical Control of ma-
582 chines – Program format and definition of address words – Part 1: Data format for posi-
583 tioning, line and contouring control systems. Geneva, Switzerland, 1982.
- 584 Electronic Industries Association. *ANSI/EIA-494-B-1992*, 32 Bit Binary CL (BCL) and
585 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines.
586 Washington, D.C. 1992.
- 587 National Aerospace Standard. *Uniform Cutting Tests - NAS Series*: Metal Cutting Equip-
588 ment Specifications. Washington, D.C. 1969.
- 589 International Organization for Standardization. *ISO 10303-11*: 1994, Industrial automa-
590 tion systems and integration Product data representation and exchange Part 11: Descrip-
591 tion methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.
- 592 International Organization for Standardization. *ISO 10303-21*: 1996, Industrial automa-
593 tion systems and integration – Product data representation and exchange – Part 21: Imple-
594 mentation methods: Clear text encoding of the exchange structure. Geneva, Switzerland,
595 1996.
- 596 H.L. Horton, F.D. Jones, and E. Oberg. *Machinery's Handbook*. Industrial Press, Inc.

- 597 New York, 1984.
- 598 International Organization for Standardization. *ISO 841-2001: Industrial automation sys-*
599 *tems and integration - Numerical control of machines - Coordinate systems and motion*
600 *nomenclature*. Geneva, Switzerland, 2001.
- 601 *ASME B5.59-2 Version 9c: Data Specification for Properties of Machine Tools for Milling*
602 *and Turning*. 2005.
- 603 *ASME/ANSI B5.54: Methods for Performance Evaluation of Computer Numerically Con-*
604 *trolled Machining Centers*. 2005.
- 605 OPC Foundation. *OPC Unified Architecture Specification, Part 1: Concepts Version 1.00*.
606 July 28, 2006.
- 607 International Organization for Standardization. *ISO 13399: Cutting tool data representa-*
608 *tion and exchange*. Geneva, Switzerland, 2000.

609 B Additional Illustrations

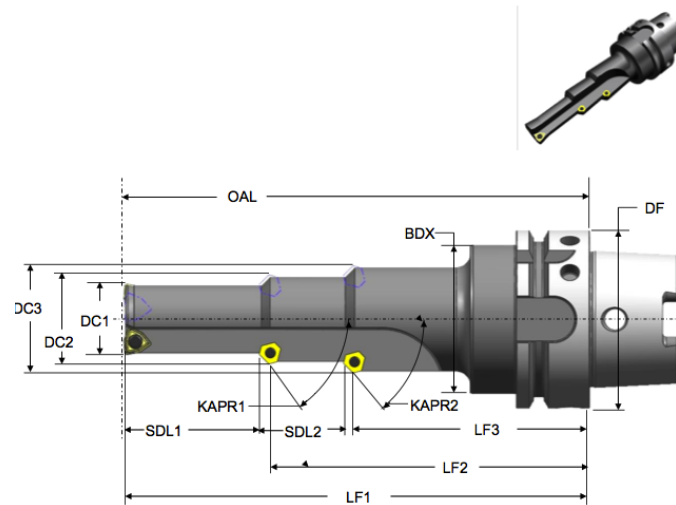


Figure 27: Cutting Tool Measurement Diagram 1
(Cutting Tool, Cutting Item, and Assembly Item – ISO 13399)

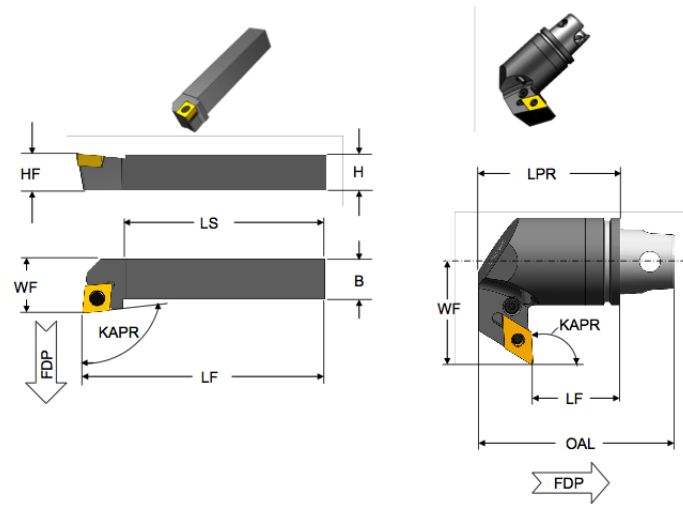


Figure 28: Cutting Tool Measurement Diagram 2
(Cutting Tool, Cutting Item, and Assembly Item – ISO 13399)

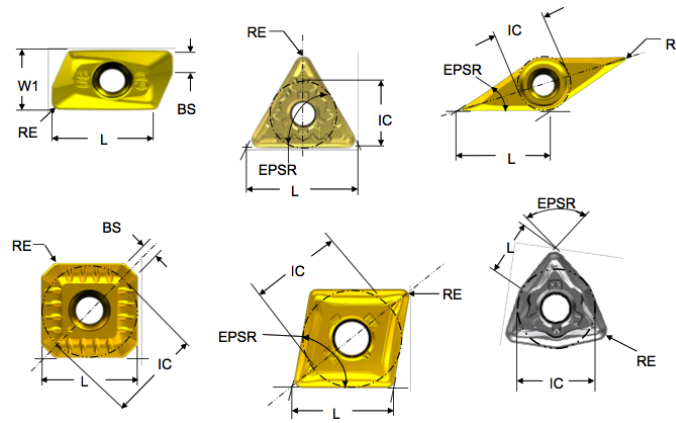


Figure 29: Cutting Tool Measurement Diagram 3
(Cutting Item – ISO 13399)

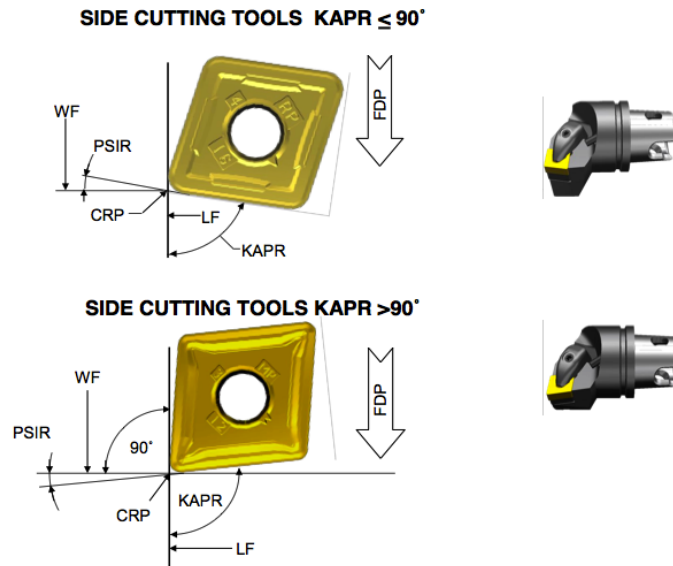


Figure 30: Cutting Tool Measurement Diagram 4
(Cutting Item – ISO 13399)

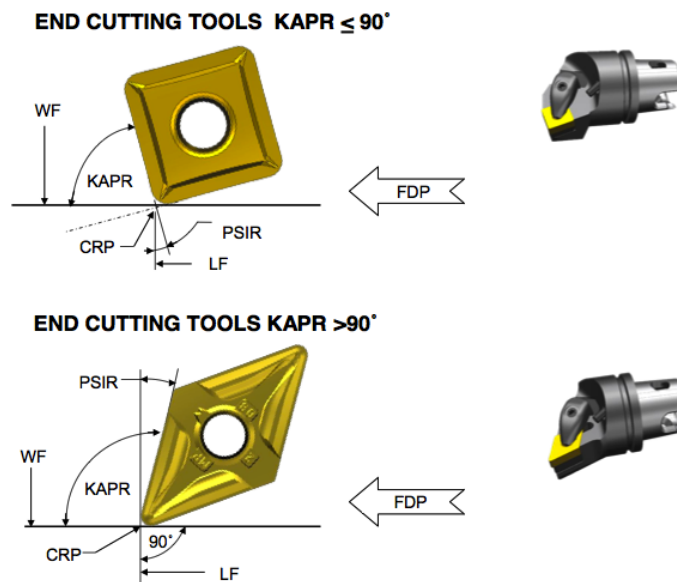


Figure 31: Cutting Tool Measurement Diagram 5
(Cutting Item – ISO 13399)

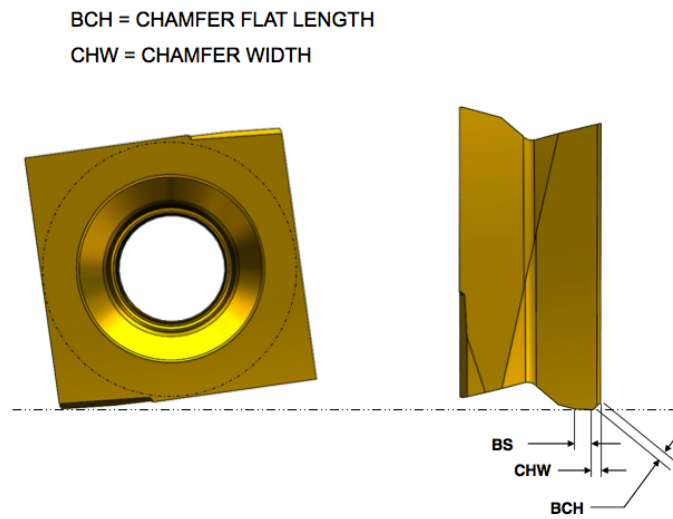


Figure 32: Cutting Tool Measurement Diagram 6
(Cutting Item – ISO 13399)

610 C Cutting Tool Example

611 C.1 Shell Mill

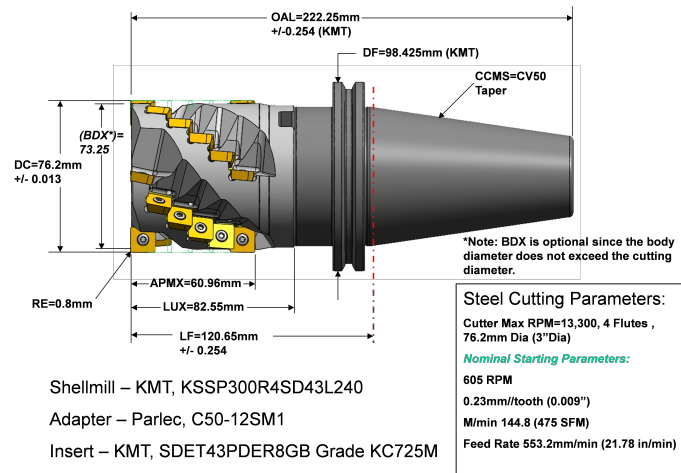


Figure 33: Shell Mill Side View

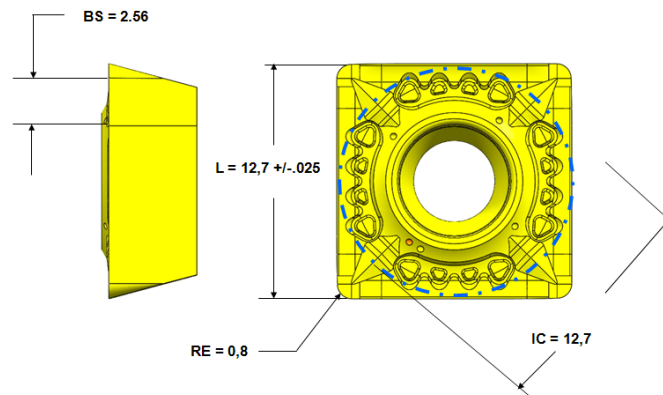


Figure 34: Indexable Insert Measurements

Example 1: Example for Indexable Insert Measurements

```

612 1 <?xml version="1.0" encoding="UTF-8"?>
613 2 <MTConnectAssets
614 3 xmlns:m="urn:mtconnect.org:MTConnectAssets:1.2"
615 4 xmlns="urn:mtconnect.org:MTConnectAssets:1.2"
616 5 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
617 6 xsi:schemaLocation="urn:mtconnect.org:MTConnectAssets:1.2
618 7 http://mtconnect.org/schemas/MTConnectAssets\_1.2.xsd">
619 8 <Header creationTime="2011-05-11T13:55:22"
620 9 assetBufferSize="1024" sender="localhost"

```

```

621 10  assetCount="2" version="1.2" instanceId="1234"/>
622 11  <Assets>
623 12  <CuttingTool serialNumber="1" toolId="KSSP300R4SD43L240"
624 13  timestamp="2011-05-11T13:55:22" assetId="KSSP300R4SD43L240.1"
625 14  manufacturers="KMT,Parlec">
626 15    <CuttingToolLifeCycle>
627 16    <CutterStatus><Status>NEW</Status></CutterStatus>
628 17    <ProcessSpindleSpeed maximum="13300"
629 18    nominal="605">10000</ProcessSpindleSpeed>
630 19    <ProcessFeedRate
631 20    nominal="9.22">9.22</ProcessSpindleSpeed>
632 21    <ConnectionCodeMachineSide>CV50
633 22    </ConnectionCodeMachineSide>
634 23    <Measurements>
635 24      <BodyDiameterMax code="BDX">73.25
636 25      </BodyDiameterMax>
637 26      <OverallToolLength nominal="222.25"
638 27      minimum="221.996" maximum="222.504"
639 28      code="OAL">222.25</OverallToolLength>
640 29      <UsableLengthMax code="LUX" nominal="82.55">82.55
641 30      </UsableLengthMax>
642 31      <CuttingDiameterMax code="DC" nominal="76.2"
643 32      maximum="76.213" minimum="76.187">76.2
644 33      </CuttingDiameterMax>
645 34      <BodyLengthMax code="LF" nominal="120.65"
646 35      maximum="120.904" minimum="120.404">120.65
647 36      </BodyLengthMax>
648 37      <DepthOfCutMax code="APMX"
649 38      nominal="60.96">60.95</DepthOfCutMax>
650 39      <FlangeDiameterMax code="DF"
651 40      nominal="98.425">98.425</FlangeDiameterMax>
652 41    </Measurements>
653 42    <CuttingItems count="24">
654 43      <CuttingItem indices="1-24" itemId="SDET43PDER8GB"
655 44      manufacturers="KMT" grade="KC725M">
656 45        <Measurements>
657 46          <CuttingEdgeLength code="L" nominal="12.7"
658 47          minimum="12.675" maximum="12.725">12.7
659 48          </CuttingEdgeLength>
660 49          <WiperEdgeLength code="BS" nominal="
661 50          "2.56">2.56</WiperEdgeLength>
662 51          <IncribedCircleDiameter code="IC"
663 52          nominal="12.7">12.7
664 53          </IncribedCircleDiameter>
665 54          <CornerRadius code="RE" nominal="0.8">
666 55          0.8</CornerRadius>
667 56        </Measurements>
668 57      </CuttingItem>
669 58    </CuttingItems>
670 59    </CuttingToolLifeCycle>
671 60  </CuttingTool>

```

```
672 61    </Assets>
673 62    </MTConnectAssets>
```

674 C.2 Step Drill

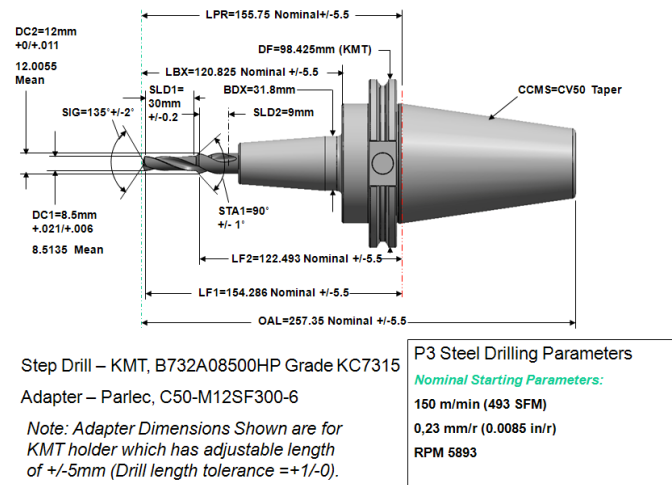


Figure 35: Step Mill Side View

Example 2: Example for Step Mill Side View

```

675 1 <?xml version="1.0" encoding="UTF-8"?>
676 2 <MTConnectAssets xmlns:m="urn:mtconnect.org:MTConnectAssets:1.2"
677 3 xmlns="urn:mtconnect.org:MTConnectAssets:1.2"
678 4 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
679 5 xsi:schemaLocation="urn:mtconnect.org:MTConnectAssets:1.2
680 6 http://mtconnect.org/schemas/MTConnectAssets/_1.2.xsd">
681 7   <Header creationTime="2011-05-
682 8   11T13:55:22" assetBufferSize="1024"
683 9   sender="localhost" assetCount="2" version="1.2" instanceId="1234"/>
684 10  <Assets>
685 11    <CuttingTool serialNumber="1_" toolId="B732A08500HP"
686 12    timestamp="2011-05-11T13:55:22" assetId="B732A08500HP_"
687 13    manufacturers="KMT,Parlec">
688 14      <Description>
689 15        Step Drill - KMT, B732A08500HP Grade KC7315
690 16        Adapter - Parlec, C50-M12SF300-6
691 17      </Description>
692 18      <CuttingToolLifeCycle>
693 19        <CutterStatus><Status>NEW</Status></CutterStatus>
694 20        <ProcessSpindleSpeed nominal="5893">5893</ProcessSpindleSpeed>
695 21        <ProcessFeedRate nominal="2.5">2.5</ProcessFeedRate>
696 22        <ConnectionCodeMachineSide>CV50 Taper</ConnectionCodeMachineSide>
697 23        <Measurements>
698 24          <BodyDiameterMax code="BDX">31.8</BodyDiameterMax>
699 25          <BodyLengthMax code="LBX" nominal="120.825" maximum="126.325"
700 26          minimum="115.325">120.825</BodyLengthMax>
701 27          <ProtrudingLength code="LPR" nominal="155.75" maximum="161.25"
702 28          minimum="150.26">155.75</ProtrudingLength>

```

```

703 29      <FlangeDiameterMax code="DF"
704 30      nominal="98.425">98.425</FlangeDiameterMax>
705 31      <OverallToolLength nominal="257.35" minimum="251.85"
706 32      maximum="262.85" code="OAL">257.35</OverallToolLength>
707 33  </Measurements>
708 34  <CuttingItems count="2">
709 35      <CuttingItem indices="1" manufacturers="KMT" grade="KC7315">>
710 36          <Measurements>
711 37              <CuttingDiameter code="DC1" nominal="8.5" maximum="8.521"
712 38              minimum="8.506">8.5135</CuttingDiameter>
713 39              <StepIncludedAngle code="STA1" nominal="90" maximum="91"
714 40              minimum="89">90</StepIncludedAngle>
715 41              <FunctionallLength code="LF1" nominal="154.286"
716 42              minimum="148.786"
717 43              maximum="159.786">154.286</FunctionallLength>
718 44              <StepDiameterLength code="SDL1"
719 45              nominal="9">9</StepDiameterLength>
720 46              <PointAngle code="SIG" nominal="135" minimum="133"
721 47              maximum="137">135</PointAngle>
722 48          </Measurements>
723 49      </CuttingItem>
724 50      <CuttingItem indices="2" manufacturers="KMT" grade="KC7315">>
725 51          <Measurements>
726 52              <CuttingDiameter code="DC2" nominal="12" maximum="12.011"
727 53              minimum="12">12</CuttingDiameter>
728 54              <FunctionallLength code="LF2" nominal="122.493"
729 55              maximum="127.993"
730 56              minimum="116.993">122.493</FunctionallLength>
731 57              <StepDiameterLength code="SDL2"
732 58              nominal="9">9</StepDiameterLength>
733 59          </Measurements>
734 60      </CuttingItem>
735 61  </CuttingItems>
736 62  </CuttingToolLifeCycle>
737 63  </CuttingTool>
738 64  </Assets>
739 65 </MTConnectAssets>

```

740 C.3 Shell Mill with Individual Loci

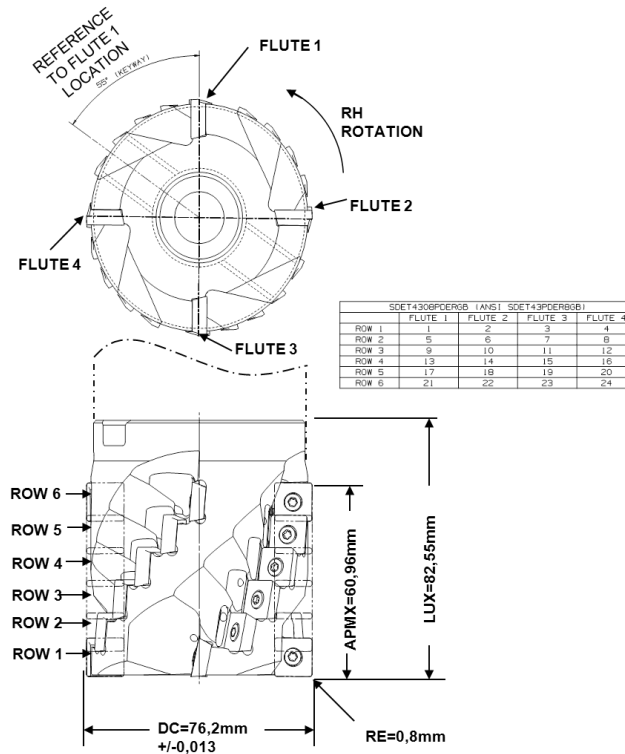


Figure 36: Shell Mill with Explicate Loci

Example 3: Example for Shell Mill with Explicate Loci

```

741 1 <?xml version="1.0" encoding="UTF-8"?>
742 2 <MTConnectAssets xmlns:m="urn:mtconnect.org:MTConnectAssets:1.2"
743 3 xmlns="urn:mtconnect.org:MTConnectAssets:1.2"
744 4 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
745 5 xsi:schemaLocation="urn:mtconnect.org:MTConnectAssets:1.2
746 6 http://mtconnect.org/schemas/MTConnectAssets/_1.2.xsd">
747 7 <Header creationTime="2011-05-11T13:55:22" assetBufferSize="1024"
748 8 sender="localhost" assetCount="2" version="1.2" instanceId="1234"/>
749 9 <Assets>
750 10 <CuttingTool serialNumber="1" toolId="KSSP300R4SD43L240"
751 11 timestamp="2011-05-11T13:55:22" assetId="KSSP300R4SD43L240.1"
752 12 manufacturers="KMT,Parlec">
753 13 <Description>Keyway: 55 degrees</Description>
754 14 <CuttingToolLifeCycle>
755 15 <CutterStatus><Status>NEW</Status></CutterStatus>
756 16 <Measurements>
757 17 <UsableLengthMax code="LUX"
758 18 nominal="82.55">82.55</UsableLengthMax>
759 19 <CuttingDiameterMax code="DC" nominal="76.2" maximum="76.213"

```



```

760 20         minimum="76.187">76.2</CuttingDiameterMax>
761 21         <DepthOfCutMax code="APMX" nominal="60.96">60.95</DepthOfCutMax>
762 22     </Measurements>
763 23     <CuttingItems count="24">
764 24         <CuttingItem indices="1" itemId="SDET43PDER8GB"
765 25         manufacturers="KMT">
766 26             <Locus>FLUTE: 1, ROW: 1</Locus>
767 27             <Measurements>
768 28                 <DriveAngle code="DRVA" nominal="55">55</DriveAngle>
769 29             </Measurements>
770 30         </CuttingItem>
771 31         <CuttingItem indices="2-24" itemId="SDET43PDER8GB"
772 32         manufacturers="KMT">
773 33             <Locus>FLUTE: 2-4, ROW: 1; FLUTE: 1-4, ROW 2-6</Locus>
774 34         </CuttingItem>
775 35     </CuttingItems>
776 36 </CuttingToolLifeCycle>
777 37 </CuttingTool>
778 38 </Assets>
779 39 </MTConnectAssets>

```

780 C.4 Drill with Individual Loci

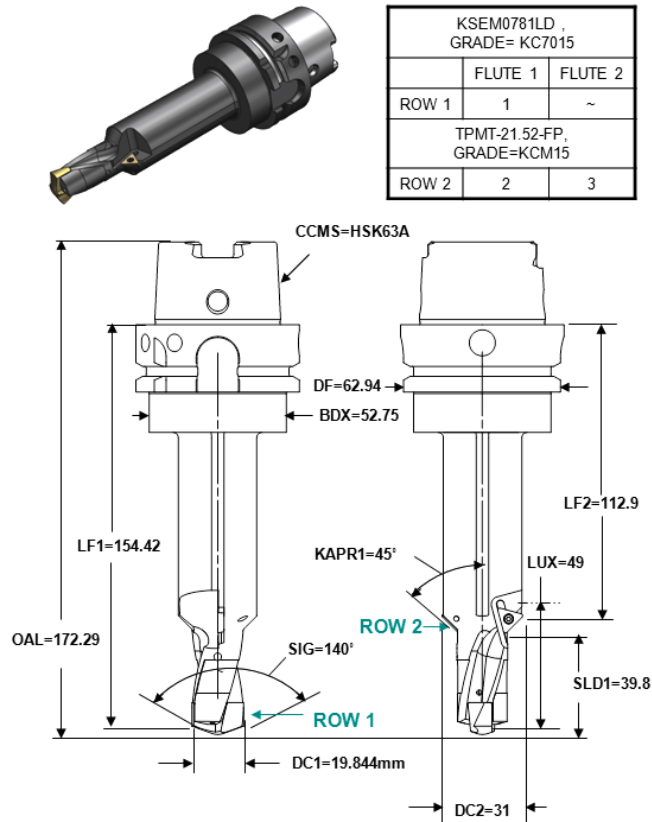


Figure 37: Step Drill with Explicate Loci

Example 4: Example for Step Drill with Explicate Loci

```

781 1 <?xml version="1.0" encoding="UTF-8"?>
782 2 <MTConnectAssets xmlns:m="urn:mtconnect.org:MTConnectAssets:1.2"
783 3 xmlns="urn:mtconnect.org:MTConnectAssets:1.2"
784 4 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
785 5 xsi:schemaLocation="urn:mtconnect.org:MTConnectAssets:1.2
786 6 http://mtconnect.org/schemas/MTConnectAssets/_1.2.xsd">
787 7   <Header creationTime="2011-05-11T13:55:22" assetBufferSize="1024"
788 8   sender="localhost" assetCount="2" version="1.2" instanceId="1234"/>
789 9   <Assets>
790 10     <CuttingTool serialNumber="1" toolId="KSEM0781LD"
791 11     timestamp="2011-05-11T13:55:22" assetId="KSEM0781LD.1" manufacturers="KMT">
792 12       <CuttingToolLifeCycle>
793 13         <CutterStatus><Status>NEW</Status></CutterStatus>
794 14         <ConnectionCodeMachineSide>HSK63A</ConnectionCodeMachineSide>
795 15         <Measurements>
796 16           <BodyDiameterMax code="BDX">52.75</BodyDiameterMax>
797 17           <OverallToolLength nominal="172.29"

```

```

798 18         code="OAL">172.29</OverallToolLength>
799 19     <UsableLengthMax code="LUX" nominal="49">49</UsableLengthMax>
800 20     <FlangeDiameterMax code="DF"
801 21         nominal="62.94">62.94</FlangeDiameterMax>
802 22 </Measurements>
803 23 <CuttingItems count="3">
804 24     <CuttingItem indices="1" itemId="KSEM0781LD" manufacturers="KMT"
805 25         grade="KC7015">
806 26         <Locus>FLUTE: 1, ROW: 1</Locus>
807 27         <Measurements>
808 28             <FunctionalLength code="LF1" nominal="154.42">154.42</FunctionalLength>
809 29             <CuttingDiameter code="DC1" nominal="19.844">19.844</CuttingDiameter>
810 30             <PointAngle code="SIG" nominal="140">140</PointAngle>
811 31             <ToolCuttingEdgeAngle code="KAPR1" nominal="45">45</ToolCuttingEdgeAngle>
812 32             <StepDiameterLength code="SLD1" nominal="39.8">39.8</StepDiameterLength>
813 33         </Measurements>
814 34     </CuttingItem>
815 35     <CuttingItem indices="2-3" itemId="TPMT-21.52-FP"
816 36         manufacturers="KMT" grade="KCM15">
817 37         <Locus>FLUTE: 1-2, ROW: 2</Locus>
818 38         <Measurements>
819 39             <FunctionalLength code="LF2" nominal="112.9">119.2</FunctionalLength>
820 40             <CuttingDiameter code="DC2" nominal="31">31</CuttingDiameter>
821 41         </Measurements>
822 42     </CuttingItem>
823 43 </CuttingItems>
824 44 </CuttingToolLifeCycle>
825 45 </CuttingTool>
826 46 </Assets>
827 47 </MTConnectAssets>

```

828 C.5 Shell Mill with Different Inserts on First Row

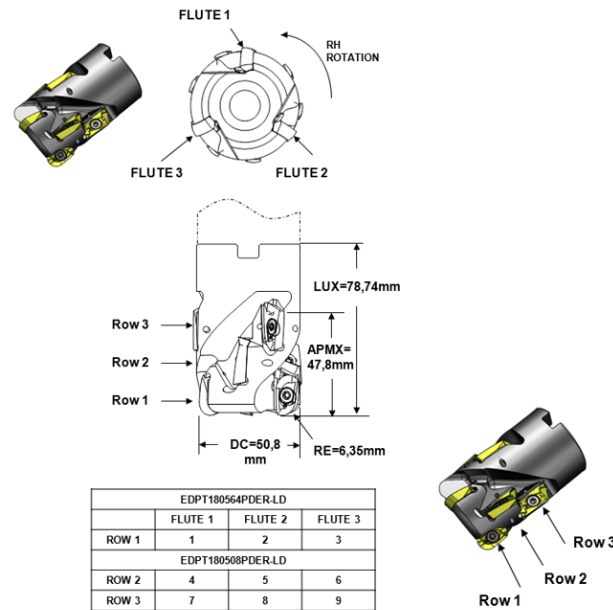


Figure 38: Shell Mill with Different Inserts on First Row

Example 5: Example for Shell Mill with Different Inserts on First Row

```

829 1 <?xml version="1.0" encoding="UTF-8"?>
830 2 <MTConnectAssets xmlns:m="urn:mtconnect.org:MTConnectAssets:1.2"
831 3 xmlns="urn:mtconnect.org:MTConnectAssets:1.2"
832 4 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
833 5 xsi:schemaLocation="urn:mtconnect.org:MTConnectAssets:1.2
834 6 http://mtconnect.org/schemas/MTConnectAssets/_1.2.xsd">
835 7   <Header creationTime="2011-05-11T13:55:22" assetBufferSize="1024"
836 8   sender="localhost" assetCount="2" version="1.2" instanceId="1234"/>
837 9   <Assets>
838 10    <CuttingTool serialNumber="1" toolId="XXX" timestamp="2011-05-11T13:55:22"
839 11    assetId="XXX.1" manufacturers="KMT">
840 12      <CuttingToolLifeCycle>
841 13        <CutterStatus><Status>NEW</Status></CutterStatus>
842 14        <Measurements>
843 15          <DepthOfCutMax code="APMX" nominal="47.8">47.8</DepthOfCutMax>
844 16          <CuttingDiameterMax code="DC"
845 17          nominal="50.8">50.8</CuttingDiameterMax>
846 18          <UsableLengthMax code="LUX"
847 19          nominal="78.74">78.74</UsableLengthMax>
848 20        </Measurements>
849 21        <CuttingItems count="9">
850 22          <CuttingItem indices="1-3" itemId="EDPT180564PDER-LD"
851 23          manufacturers="KMT">
852 24            <Locus>FLUTE: 1-3, ROW: 1</Locus>

```

```

853 25         <Measurements>
854 26             <CornerRadius code="RE" nominal="6.25">6.35</CornerRadius>
855 27         </Measurements>
856 28     </CuttingItem>
857 29     <CuttingItem indices="4-9" itemId="EDPT180508PDER-LD"
858 30         manufacturers="KMT">
859 31         <Locus>FLANGE: 1-4, ROW: 2-3</Locus>
860 32     </CuttingItem>
861 33 </CuttingItems>
862 34 </CuttingToolLifeCycle>
863 35 </CuttingTool>
864 36 </Assets>
865 37 </MTConnectAssets>

```



MTConnect[®] Standard

Part 4.2 – File Asset Information Model

Version 1.8.0

Prepared for: MTConnect Institute
Prepared on: September 6, 2021

MTConnect[®] is a registered trademark of AMT - The Association for Manufacturing Technology. Use of *MTConnect* is limited to use as specified on <http://www.mtconnect.org/>.

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	Files Information Model	9
3.1	AbstractFile	9
3.1.1	Attributes for AbstractFile	11
3.1.1.1	AbstractFile applicationCategory types	11
3.1.1.2	AbstractFile applicationType types	12
3.1.2	Elements for AbstractFile	12
3.1.3	FileProperty	12
3.1.3.1	Attributes for FileProperty	12
3.1.4	FileComment	13
3.1.4.1	Attributes for FileComment	13
3.2	File	13
3.2.1	Attributes for File	15
3.2.1.1	File states	15
3.2.2	Elements for File	15
3.2.3	FileLocation	16
3.2.3.1	Attributes for FileLocation	16
3.2.4	Destination	17
3.2.4.1	Attributes for Destination	17
3.3	FileArchetype	17
	Appendices	18
A	Bibliography	18

Table of Figures

Figure 1: AbstractFile Diagram	10
Figure 2: File Diagram	14

List of Tables

Table 1: Attributes for AbstractFile	11
Table 2: AbstractFile applicationCategory types	11
Table 3: AbstractFile applicationType types	12
Table 4: Elements for AbstractFile	12
Table 5: Attributes for FileProperty	13
Table 6: Attributes for FileComment	13
Table 7: Attributes for File	15
Table 8: File states	15
Table 9: Elements for File	16
Table 10: Attributes for FileLocation	16
Table 11: Attributes for Destination	17

1 1 Purpose of This Document

2 This document, *MTConnect Standard: Part 4.2 - File Asset Information Model* of the
3 MTConnect Standard, establishes the rules and terminology to be used by designers to
4 describe the function and operation of files used within manufacturing and to define the
5 data that is provided by an *Agent* from a piece of equipment. This part of the Standard also
6 defines the structure for the XML document that is returned from an *Agent* in response to
7 a `probe` request.

8 The data associated with these files will be retrieved from multiple sources that are respon-
9 sible 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 URL

16 Stands for Uniform Resource Locator.

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

18 W3C

19 The World Wide Web Consortium (W3C) is an international community that devel-
 20 ops open standards to ensure the long-term growth of the Web.

21 See <https://www.w3.org/>.

22 *Agent*

23 Refers to an MTConnect Agent.

24 Software that collects data published from one or more piece(s) of equipment, orga-
 25 nizes that data in a structured manner, and responds to requests for data from client
 26 software systems by providing a structured response in the form of a *Response Doc-*
 27 *ument* that is constructed using the *semantic data models* defined in the Standard.

28 Appears in the documents in the following form: *Agent*.

29 *Asset*

30 item, thing or entity that has potential or actual value to an organization *Ref:ISO*
 31 *55000:2014(en)*

32 Note 1 to entry: Value can be tangible or intangible, financial or non-financial,
 33 and includes consideration of risks and liabilities. It can be positive or negative
 34 at different stages of the asset life.

35 Note 2 to entry: Physical assets usually refer to equipment, inventory and prop-
 36 erties owned by the organization. Physical assets are the opposite of intangible
 37 assets, which are non-physical assets such as leases, brands, digital assets, use
 38 rights, licences, intellectual property rights, reputation or agreements.

Note 3 to entry: A grouping of assets referred to as an asset system could also be considered as an asset.

Child Element

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

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

Component

General meaning:

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

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

Used in *Information Models*:

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

- When used as an XML container to organize *Lower Level Component* elements.

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

- When used as an abstract XML element. *Component* is replaced in a data model by a type of *Component* element. *Component* is also an XML container used to organize *Lower Level Component* elements, *Data Entities*, or both.

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

Current Request

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

Data Entity

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

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

71 ***Devices Information Model***

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

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

75 ***Equipment Metadata***

76 See *Metadata*

77 ***Information Model***

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

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

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

84 ***Lower Level***

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

86 ***Metadata***

87 Data that provides information about other data.

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

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

93 ***MTConnect Agent***

94 See definition for *Agent*.

95 ***MTConnect Asset***

96 An *MTConnect Asset* is an *Asset* used by the manufacturing process to perform
97 tasks.

98 Note 1 to entry: An *MTConnect Asset* relies upon an *MTConnect Device* to
99 provide *observations* and information about itself and the *MTConnect Device*
100 revises the information to reflect changes to the *MTConnect Asset* during their
101 interaction. Examples of *MTConnect Assets* are Cutting Tools, Part Information,
102 Manufacturing Processes, Fixtures, and Files.

103 Note 2 to entry: A singular `assetId` uniquely identifies an *MTConnect Asset*
 104 throughout its lifecycle and is used to track and relate the *MTConnect Asset* to
 105 other *MTConnect Devices* and entities.

106 Note 3 to entry: *MTConnect Assets* are temporally associated with a device and
 107 can be removed from the device without damage or alteration to its primary
 108 functions.

109

110 ***MTConnect Device***

111 An *MTConnect Device* is a piece of equipment or a manufacturing system that pro-
 112 duces *observations* about itself and/or publishes data using the *MTConnect Infor-*
 113 *mation Model*.

114 ***MTConnect Information Model***

115 See *Information Model*

116 ***MTConnectDevices Response Document***

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

119 ***MTConnectStreams Response Document***

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

122 ***observation***

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

124 ***Observations Information Model***

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

127 ***organize***

128 The act of containing and owning one or more elements.

129 ***Parent Element***

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

132 Appears in the documents in the following form: *Parent Element*.

133 ***Probe Request***

134 A *Probe Request* is a *Request* to an *Agent* to produce an *MTConnectDevices Re-*
135 *response Document* containing the *Devices Information Model*.

136 ***Request***

137 A communications method where a client software application transmits a message
138 to an *Agent*. That message instructs the *Agent* to respond with specific information.
139 Appears in the documents in the following form: *Request*.

140 ***Response Document***

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

143 ***Sample Request***

144 A *Sample Request* is a *Request* to an *Agent* to produce an *MTConnectStreams Re-*
145 *response Document* containing the *Observations Information Model* for a set of time-
146 stamped *observations* made by *Components*.

147 ***semantic data model***

148 A methodology for defining the structure and meaning for data in a specific logical
149 way.
150 It provides the rules for encoding electronic information such that it can be inter-
151 preted by a software system.
152 Appears in the documents in the following form: *semantic data model*.

153 ***sequence number***

154 The primary key identifier used to manage and locate a specific piece of *Streaming*
155 *Data* in an *Agent*.
156 *sequence number* is a monotonically increasing number within an instance of an
157 *Agent*.
158 Appears in the documents in the following form: *sequence number*.

159 ***Streaming Data***

160 The values published by a piece of equipment for the *Data Entities* defined by the
161 *Equipment Metadata*.
162 Appears in the documents in the following form: *Streaming Data*.

Structural Element

General meaning:

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

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

Used to indicate hierarchy of Components:

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

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

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

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

Top Level

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

2.2 Acronyms

AMT

The Association for Manufacturing Technology

2.3 MTConnect References

[MTConnect Part 1.0] *MTConnect Standard Part 1.0 - Overview and Fundamentals*. Version 1.8.0.

[MTConnect Part 4.2] *MTConnect Standard: Part 4.2 - File Asset Information Model*. Version 1.8.0.

186 3 Files Information Model

187 Manufacturing processes require various documents, programs, setup sheets, and digital
188 media available at the device for a given process. The `File` and `FileArchetype` As-
189 sets provide a mechanism to communicate specific "Files" that are relevant to a process
190 where the media is located on a server and represented by a Universal Resource Locator
191 (URL).

192 The `FileArchetype` contains metadata common to all `File` Assets for a certain
193 purpose. The `File` Asset references the file specific to a given device or set of devices.
194 The `File` Asset does not hold the contents of the file, it contains a reference to the
195 location (URL) used to access the information. The metadata associated with the `File`
196 provides semantic information about the representation (mime-type) and the application
197 associated with the `File`. The application of the file is an extensible controlled vocabulary
198 with common manufacturing uses provided.

199 3.1 AbstractFile

200 An `AbstractFile` is an abstract `Asset` type model that contains the common proper-
201 ties of the `File` and `FileArchetype` types.

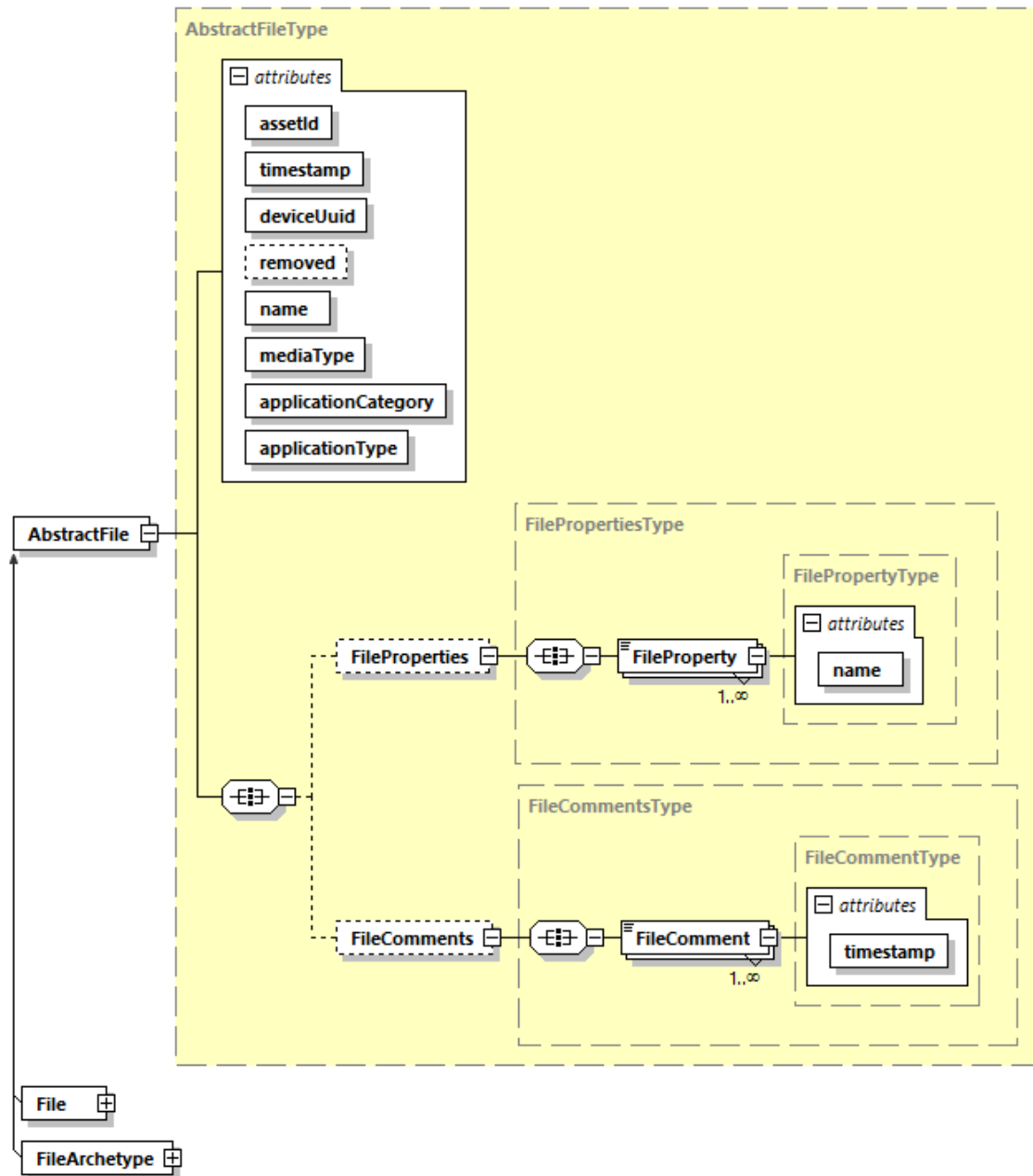


Figure 1: AbstractFile Diagram

202 3.1.1 Attributes for AbstractFile

203 *Table 1* lists the attributes for an `AbstractFile` element in addition to attributes inher-
 204 ited from `Asset` element.

Table 1: Attributes for AbstractFile

Attribute	Description	Occurrence
name	The name of the file. The value of <code>name</code> MUST be a string.	1
mediaType	The mime type of the file. The value of <code>mediaType</code> MUST be a string.	1
applicationCategory	The category of application that will use this file.	1
applicationType	The type of application that will use this file.	1

205 3.1.1.1 AbstractFile applicationCategory types

206 *Table 2* lists the types for `applicationCategory` attribute of `AbstractFile` ele-
 207 ment.

Table 2: AbstractFile applicationCategory types

type	Description
ASSEMBLY	Files regarding the fully assembled product.
DEVICE	Device related files.
HANDLING	Files relating to the handling of material.
MAINTENANCE	File relating to equipment maintenance.
PART	Files relating to a part.
PROCESS	Files related to the manufacturing process.
INSPECTION	Files related to the quality inspection.
SETUP	Files related to the setup of a process.

208 3.1.1.2 AbstractFile applicationType types

209 *Table 3* lists the types for applicationType attribute of AbstractFile element.

Table 3: AbstractFile applicationType types

type	Description
DESIGN	Computer aided design files or drawings.
DATA	Generic data.
DOCUMENTATION	Documentation regarding a category of file.
INSTRUCTIONS	User instructions regarding the execution of a task.
LOG	The data related to the history of a machine or process.
PRODUCTION_PROGRAM	Machine instructions to perform a process.

210 3.1.2 Elements for AbstractFile

211 *Table 4* lists the elements for an AbstractFile element.

Table 4: Elements for AbstractFile

Element	Description	Occurrence
FileProperties	FileProperties <i>organizes</i> one or more FileProperty entities for Files.	0..1
FileComments	FileComments <i>organizes</i> one or more FileComment entities for Files.	0..1

212 3.1.3 FileProperty

213 A key-value pair providing additional metadata about a File.

214 The value for FileProperty **MUST** be a string.

215 3.1.3.1 Attributes for FileProperty

216 *Table 5* lists the attributes for a `FileProperty` element.

Table 5: Attributes for `FileProperty`

Attribute	Description	Occurrence
name	The name of the <code>FileProperty</code>	1

217 3.1.4 FileComment

218 A remark or interpretation for human interpretation associated with a `File` or `FileArchetype`.

219 The value for `FileComment` **MUST** be a string.

220 3.1.4.1 Attributes for `FileComment`

221 *Table 6* lists the attributes for a `FileComment` element.

Table 6: Attributes for `FileComment`

Attribute	Description	Occurrence
timestamp	The time the comment was made. The value for <code>timestamp</code> MUST be reported in ISO 8601 format.	1

222 3.2 File

223 The `File Asset` is an `AbstractFile` with information about the `File` instance and
224 its URL.

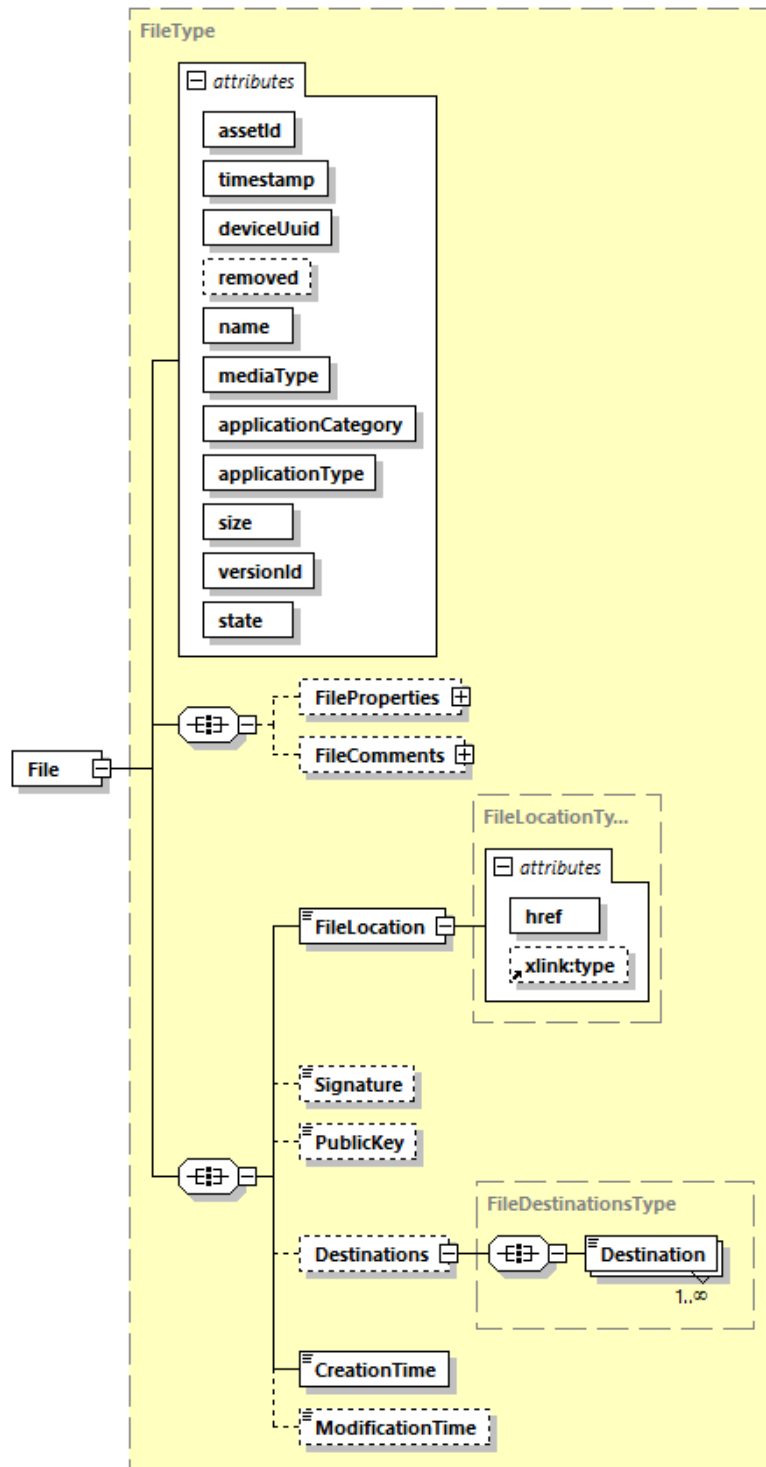


Figure 2: File Diagram

225 3.2.1 Attributes for File

226 *Table 7* lists the attributes for a `File` element in addition to attributes inherited from
 227 `AbstractFile` (See *Section 3.1 - AbstractFile*).

Table 7: Attributes for File

Attribute	Description	Occurrence
<code>size</code>	The size of the file in bytes. The value of <code>size</code> MUST be an integer.	1
<code>versionId</code>	The version identifier of the file. The value of <code>versionId</code> MUST be a string.	1
<code>state</code>	The state of the file.	1

228 3.2.1.1 File states

229 *Table 8* lists the values for `state` attribute of `File` element.

Table 8: File states

type	Description
EXPERIMENTAL	Used for processes other than production or otherwise defined.
PRODUCTION	Used for production processes.
REVISION	The content is modified from PRODUCTION or EXPERIMENTAL.

230 3.2.2 Elements for File

231 *Table 9* lists the elements for a `File` element.

Table 9: Elements for File

Element	Description	Occurrence
Signature	A secure hash of the file. The value for <code>Signature</code> MUST be an x509 data block.	0..1
PublicKey	The public key used to verify the signature. The value for <code>PublicKey</code> MUST be an x509 data block.	0..1
CreationTime	The time the file was created. The value for <code>CreationTime</code> MUST be reported in ISO 8601 format.	1
ModificationTime	The time the file was modified. The value for <code>ModificationTime</code> MUST be reported in ISO 8601 format.	0..1
FileLocation	The URL reference to the file location.	1
Destinations	<code>Destinations</code> <i>organizes</i> one or more <code>Destination</code> elements.	0..1

232 3.2.3 FileLocation

233 The URL reference to the file location.

234 3.2.3.1 Attributes for FileLocation

235 *Table 10* lists the attributes for a `FileLocation` element.

Table 10: Attributes for FileLocation

Attribute	Description	Occurrence
href	A URL reference to the file. <code>href</code> is of type <code>xlink:href</code> from the W3C XLink specification.	1

Continuation of Table 10		
Attribute	Description	Occurrence
xlink:type	The type of href for the xlink href type. MUST be locator referring to a URL.	0..1

236 3.2.4 Destination

237 The Destination is a reference to the target Device for this File.

238 3.2.4.1 Attributes for Destination

239 Table 11 lists the attributes for a Destination element.

Table 11: Attributes for Destination

Attribute	Description	Occurrence
deviceUuid	uuid of the target device or application.	1

240 3.3 FileArchetype

241 FileArchetype Asset is an AbstractFile providing information common to all
242 versions of a file.

243 See Section 3.1 - AbstractFile for details on the FileArchetype model.

244 Appendices

245 A Bibliography

- 246 Engineering Industries Association. *EIA Standard - EIA-274-D*, Interchangeable Variable,
 247 Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically
 248 Controlled Machines. Washington, D.C. 1979.
- 249 ISO TC 184/SC4/WG3 N1089. *ISO/DIS 10303-238*: Industrial automation systems and
 250 integration Product data representation and exchange Part 238: Application Protocols: Ap-
 251 plication interpreted model for computerized numerical controllers. Geneva, Switzerland,
 252 2004.
- 253 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
 254 tems and integration – Physical device control – Data model for computerized numerical
 255 controllers – Part 10: General process data. Geneva, Switzerland, 2004.
- 256 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
 257 tems and integration – Physical device control – Data model for computerized numerical
 258 controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.
- 259 International Organization for Standardization. *ISO 6983/1* – Numerical Control of ma-
 260 chines – Program format and definition of address words – Part 1: Data format for posi-
 261 tioning, line and contouring control systems. Geneva, Switzerland, 1982.
- 262 Electronic Industries Association. *ANSI/EIA-494-B-1992*, 32 Bit Binary CL (BCL) and
 263 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines.
 264 Washington, D.C. 1992.
- 265 National Aerospace Standard. *Uniform Cutting Tests* - NAS Series: Metal Cutting Equip-
 266 ment Specifications. Washington, D.C. 1969.
- 267 International Organization for Standardization. *ISO 10303-11*: 1994, Industrial automa-
 268 tion systems and integration Product data representation and exchange Part 11: Descrip-
 269 tion methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.
- 270 International Organization for Standardization. *ISO 10303-21*: 1996, Industrial automa-
 271 tion systems and integration – Product data representation and exchange – Part 21: Imple-
 272 mentation methods: Clear text encoding of the exchange structure. Geneva, Switzerland,
 273 1996.
- 274 H.L. Horton, F.D. Jones, and E. Oberg. *Machinery's Handbook*. Industrial Press, Inc.

275 New York, 1984.

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

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

281 *ASME/ANSI B5.54: Methods for Performance Evaluation of Computer Numerically Con-*
282 *trolled Machining Centers*. 2005.

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

285 International Organization for Standardization. *ISO 13399: Cutting tool data representa-*
286 *tion and exchange*. Geneva, Switzerland, 2000.



MTConnect[®] Standard
Part 4.3 – Raw Material Asset Information
Model
Version 1.8.0

Prepared for: MTConnect Institute
Prepared on: September 6, 2021

MTConnect[®] is a registered trademark of AMT - The Association for Manufacturing Technology. Use of *MTConnect* is limited to use as specified on <http://www.mtconnect.org/>.

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	Raw Material Information Model	9
3.1	RawMaterial	9
3.1.1	Attributes for RawMaterial	11
3.1.2	Elements for RawMaterial	11
3.2	Material	14
3.2.1	Attributes for Material	14
3.2.2	Elements for Material	15
	Appendices	16
A	Bibliography	16

Table of Figures

Figure 1: RawMaterial Diagram	10
Figure 2: Material Diagram	14

List of Tables

Table 1: Attributes for RawMaterial 11

Table 2: Elements for RawMaterial 12

Table 3: Attributes for Material 14

Table 4: Elements for Material 15

1 1 Purpose of This Document

2 This document, *MTConnect Standard: Part 4.3 - Raw Material Asset Information Model*
3 of the MTConnect Standard, establishes the rules and terminology to be used by designers
4 to describe the function and operation of *raw material* used within manufacturing and to
5 define the data that is provided by an *Agent* from a piece of equipment.

6 The data associated with these *raw material* will be retrieved from multiple sources that
7 are responsible for providing their knowledge of an *MTConnect Asset*.

8 2 Terminology and Conventions

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

12 2.1 Glossary

13 ***Agent***

14 Refers to an MTConnect Agent.

15 Software that collects data published from one or more piece(s) of equipment, orga-
 16 nizes that data in a structured manner, and responds to requests for data from client
 17 software systems by providing a structured response in the form of a *Response Doc-*
 18 *ument* that is constructed using the *semantic data models* defined in the Standard.

19 Appears in the documents in the following form: *Agent*.

20 ***Asset***

21 item, thing or entity that has potential or actual value to an organization *Ref:ISO*
 22 *55000:2014(en)*

23 Note 1 to entry: Value can be tangible or intangible, financial or non-financial,
 24 and includes consideration of risks and liabilities. It can be positive or negative
 25 at different stages of the asset life.

26 Note 2 to entry: Physical assets usually refer to equipment, inventory and prop-
 27 erties owned by the organization. Physical assets are the opposite of intangible
 28 assets, which are non-physical assets such as leases, brands, digital assets, use
 29 rights, licences, intellectual property rights, reputation or agreements.

30 Note 3 to entry: A grouping of assets referred to as an asset system could also
 31 be considered as an asset.

33 ***Child Element***

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

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

37 **Component**

38 General meaning:

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

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

42 Used in *Information Models*:

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

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

47 Appears in the documents in the following form: *Components*.

- 48 • When used as an abstract XML element. *Component* is replaced in a data
49 model by a type of *Component* element. *Component* is also an XML con-
50 tainer used to organize *Lower Level* Component elements, *Data Entities*, or
51 both.

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

53 **Current Request**

54 A *Current Request* is a *Request* to an *Agent* to produce an *MTConnectStreams Re-*
55 *sponse Document* containing the *Observations Information Model* for a snapshot of
56 the latest *observations* at the moment of the *Request* or at a given *sequence number*.

57 **Data Entity**

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

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

62 **Devices Information Model**

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

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

66 **Equipment Metadata**

67 See *Metadata*

68 ***Information Model***

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

71 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
72 documents and the relationship between pieces of information.
73

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

75 ***Lower Level***

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

77 ***Metadata***

78 Data that provides information about other data.

79 For example, *Equipment Metadata* defines both the *Structural Elements* that represent the physical and logical parts and sub-parts of each piece of equipment, the
80 relationships between those parts and sub-parts, and the definitions of the *Data Entities* associated with that piece of equipment.
82

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

84 ***MTConnect Agent***

85 See definition for *Agent*.

86 ***MTConnect Asset***

87 An *MTConnect Asset* is an *Asset* used by the manufacturing process to perform
88 tasks.

89 Note 1 to entry: An *MTConnect Asset* relies upon an *MTConnect Device* to
90 provide *observations* and information about itself and the *MTConnect Device*
91 revises the information to reflect changes to the *MTConnect Asset* during their
92 interaction. Examples of *MTConnect Assets* are Cutting Tools, Part Information,
93 Manufacturing Processes, Fixtures, and Files.

94 Note 2 to entry: A singular `assetId` uniquely identifies an *MTConnect Asset*
95 throughout its lifecycle and is used to track and relate the *MTConnect Asset* to
96 other *MTConnect Devices* and entities.

97 Note 3 to entry: *MTConnect Assets* are temporally associated with a device and
98 can be removed from the device without damage or alteration to its primary
99 functions.

100

101 ***MTConnect Device***

102 An *MTConnect Device* is a piece of equipment or a manufacturing system that pro-
 103 duces *observations* about itself and/or publishes data using the *MTConnect Infor-*
 104 *mation Model*.

105 ***MTConnect Information Model***

106 See *Information Model*

107 ***MTConnectDevices Response Document***

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

110 ***MTConnectStreams Response Document***

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

113 ***observation***

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

115 ***Observations Information Model***

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

118 ***Parent Element***

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

121 Appears in the documents in the following form: *Parent Element*.

122 ***Probe Request***

123 A *Probe Request* is a *Request* to an *Agent* to produce an *MTConnectDevices Re-*
 124 *sponse Document* containing the *Devices Information Model*.

125 ***raw material***

126 Crude or processed material that can be converted by manufacture, processing, or
 127 combination into a new and useful product.

128 ***Request***

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

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

Response Document

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

Sample Request

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

semantic data model

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

It provides the rules for encoding electronic information such that it can be interpreted by a software system.

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

sequence number

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

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

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

Streaming Data

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

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

Structural Element

General meaning:

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

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

Used to indicate hierarchy of Components:

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

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

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

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

Top Level

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

2.2 Acronyms

AMT

The Association for Manufacturing Technology

ASTM

American Society for Testing and Materials

2.3 MTConnect References

[MTConnect Part 1.0] *MTConnect Standard Part 1.0 - Overview and Fundamentals*. Version 1.8.0.

[MTConnect Part 4.3] *MTConnect Standard: Part 4.3 - Raw Material Asset Information Model*. Version 1.8.0.

180 **3 Raw Material Information Model**

181 Raw material represents the source of material for immediate use and sources of material
182 that may or may not be used during the manufacturing process.

183 The `RawMaterial Asset` holds the references to the content stored in the actual `Raw-`
184 `Material` container or derived about the `RawMaterial` by the system during opera-
185 tion.

186 **3.1 RawMaterial**

187 `RawMaterial` is an `Asset` that represents *raw material*.

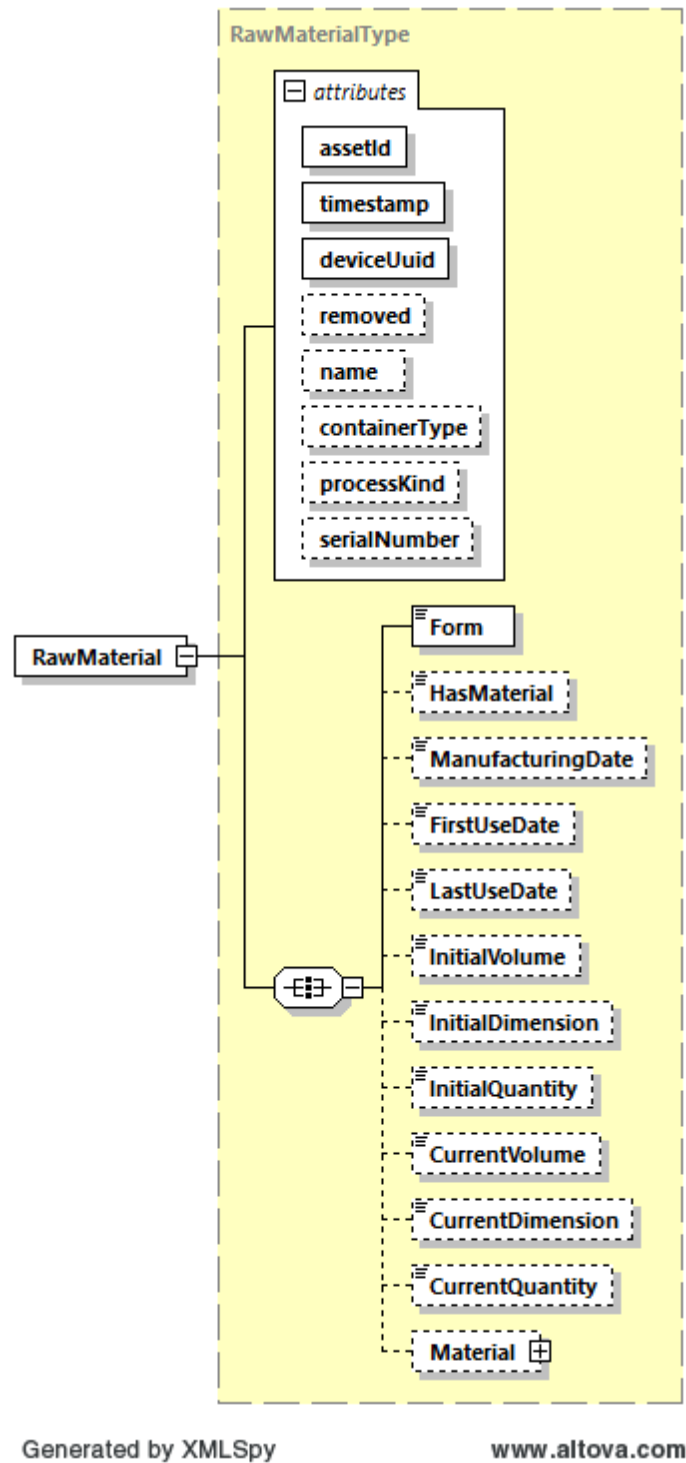


Figure 1: RawMaterial Diagram

188 3.1.1 Attributes for RawMaterial

189 *Table 1* lists the attributes for a RawMaterial element in addition to attributes inherited
 190 from Asset element.

Table 1: Attributes for RawMaterial

Attribute	Description	Occurrence
name	The <i>raw material</i> name. Examples: Container1 and AcrylicContainer. The value of name MUST be a string.	0..1
containerType	The type of container holding the <i>raw material</i> . Examples: Pallet, Canister, Cartridge, Tank, Bin, Roll and Spool. The value of type MUST be a string.	0..1
processKind	The ISO process type supported by this <i>raw material</i> . Examples include: VAT_POLYMERIZATION, BINDER_JETTING, MATERIAL_EXTRUSION, MATERIAL_JETTING, SHEET_LAMINATION, POWDER_BED_FUSION, or DIRECTED_ENERGY_DEPOSITION. <i>Ref: ASTM F2792-12a</i> The value of processId MUST be a string.	0..1
serialNumber	The serial number of the <i>raw material</i> . The value of serialNumber MUST be a string.	0..1

191 3.1.2 Elements for RawMaterial

192 *Table 2* lists the elements for a RawMaterial element.

Table 2: Elements for RawMaterial

Element	Description	Occurrence
Form	The form of the <i>raw material</i> . The value MUST be BAR, SHEET, BLOCK, CASTING, POWDER, LIQUID, GEL, FILAMENT, or GAS.	1
HasMaterial	Material has existing usable volume. The value of HasMaterial MUST be boolean.	0..1
ManufacturingDate	The date the <i>raw material</i> was created. The value of ManufacturingDate MUST be reported in ISO 8601 format.	0..1
FirstUseDate	The date <i>raw material</i> was first used. The value of FirstUseDate MUST be reported in ISO 8601 format.	0..1
LastUseDate	The date <i>raw material</i> was last used. The value of LastUseDate MUST be reported in ISO 8601 format.	0..1
InitialVolume	The amount of material initially placed in <i>raw material</i> when manufactured. The value of InitialVolume MUST be reported in CUBIC_MILLIMETER.	0..1
InitialDimension	The dimension of material initially placed in <i>raw material</i> when manufactured. The value of InitialDimension MUST be reported in MILLIMETER_3D.	0..1
InitialQuantity	The quantity of material initially placed in <i>raw material</i> when manufactured. The value MUST be an integer.	0..1

Continuation of Table 2		
Element	Description	Occurrence
CurrentVolume	<p>The amount of material currently in <i>raw material</i>.</p> <p>The value of CurrentVolume MUST be reported in CUBIC_MILLIMETER.</p>	0..1
CurrentDimension	<p>The dimension of material currently in <i>raw material</i>.</p> <p>The value of CurrentDimension MUST be reported in MILLIMETER_3D.</p>	0..1
CurrentQuantity	<p>The quantity of material currently in <i>raw material</i>.</p> <p>The value MUST be an integer.</p>	0..1
Material	<p>Material used as the <i>raw material</i>.</p> <p>See <i>Section 3.2 - Material</i> for details.</p>	0..1

193 3.2 Material

194 Material used as the *raw material*.

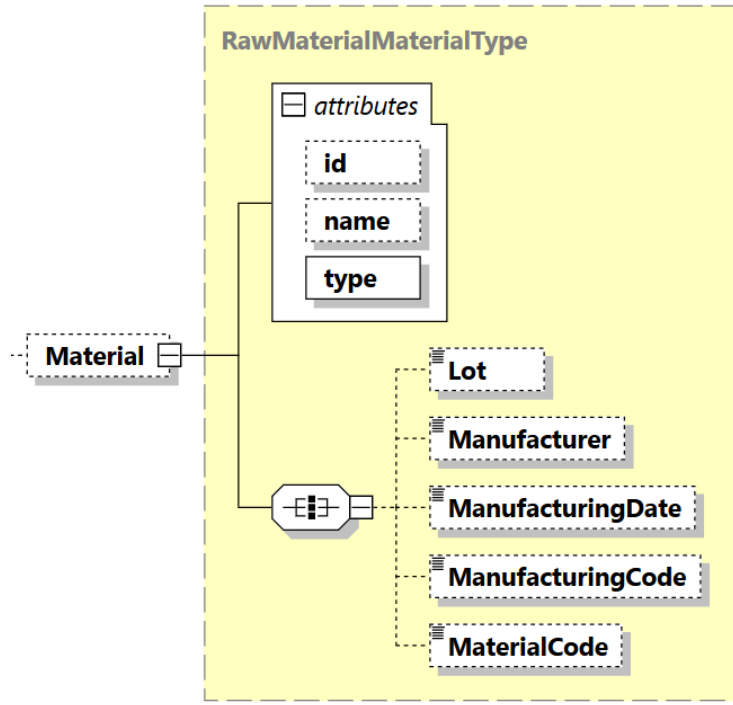


Figure 2: Material Diagram

195 3.2.1 Attributes for Material

196 *Table 3* lists the attributes for a **Material** element.

Table 3: Attributes for Material

Attribute	Description	Occurrence
id	The unique identifier for the material. The value for id MUST be a string.	0..1
name	The name of the material. Examples: ULTM9085, ABS, 4140. The value for name MUST be a string.	0..1

Continuation of Table 3		
Attribute	Description	Occurrence
type	The type of material. Examples: Metal, Polymer, Wood, 4140, Recycled, Prestine and Used. The value for type MUST be a string.	1

197 3.2.2 Elements for Material

198 *Table 4* lists the elements for a Material element.

Table 4: Elements for Material

Element	Description	Occurrence
Lot	The manufacturer's lot code of the material. The value for Lot MUST be a string.	0..1
Manufacturer	The name of the material manufacturer. The value for Manufacturer MUST be a string.	0..1
ManufacturingDate	The manufacturing date of the material from the material manufacturer. The value for ManufacturingDate MUST be reported in ISO 8601 format.	0..1
ManufacturingCode	The lot code of the raw feed stock for the material, from the feed stock manufacturer. The value for ManufacturingCode MUST be a string.	0..1
MaterialCode	The ASTM standard code that the material complies with. The value for MaterialCode MUST be a string.	0..1

199 Appendices

200 A Bibliography

- 201 Engineering Industries Association. *EIA Standard - EIA-274-D*, Interchangeable Variable,
 202 Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically
 203 Controlled Machines. Washington, D.C. 1979.
- 204 ISO TC 184/SC4/WG3 N1089. *ISO/DIS 10303-238*: Industrial automation systems and
 205 integration Product data representation and exchange Part 238: Application Protocols: Ap-
 206 plication interpreted model for computerized numerical controllers. Geneva, Switzerland,
 207 2004.
- 208 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
 209 tems and integration – Physical device control – Data model for computerized numerical
 210 controllers – Part 10: General process data. Geneva, Switzerland, 2004.
- 211 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
 212 tems and integration – Physical device control – Data model for computerized numerical
 213 controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.
- 214 International Organization for Standardization. *ISO 6983/1* – Numerical Control of ma-
 215 chines – Program format and definition of address words – Part 1: Data format for posi-
 216 tioning, line and contouring control systems. Geneva, Switzerland, 1982.
- 217 Electronic Industries Association. *ANSI/EIA-494-B-1992*, 32 Bit Binary CL (BCL) and
 218 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines.
 219 Washington, D.C. 1992.
- 220 National Aerospace Standard. *Uniform Cutting Tests* - NAS Series: Metal Cutting Equip-
 221 ment Specifications. Washington, D.C. 1969.
- 222 International Organization for Standardization. *ISO 10303-11*: 1994, Industrial automa-
 223 tion systems and integration Product data representation and exchange Part 11: Descrip-
 224 tion methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.
- 225 International Organization for Standardization. *ISO 10303-21*: 1996, Industrial automa-
 226 tion systems and integration – Product data representation and exchange – Part 21: Imple-
 227 mentation methods: Clear text encoding of the exchange structure. Geneva, Switzerland,
 228 1996.
- 229 H.L. Horton, F.D. Jones, and E. Oberg. *Machinery's Handbook*. Industrial Press, Inc.

- 230 New York, 1984.
- 231 International Organization for Standardization. *ISO 841-2001: Industrial automation sys-*
232 *tems and integration - Numerical control of machines - Coordinate systems and motion*
233 *nomenclature*. Geneva, Switzerland, 2001.
- 234 *ASME B5.59-2 Version 9c: Data Specification for Properties of Machine Tools for Milling*
235 *and Turning*. 2005.
- 236 *ASME/ANSI B5.54: Methods for Performance Evaluation of Computer Numerically Con-*
237 *trolled Machining Centers*. 2005.
- 238 OPC Foundation. *OPC Unified Architecture Specification, Part 1: Concepts Version 1.00*.
239 July 28, 2006.
- 240 International Organization for Standardization. *ISO 13399: Cutting tool data representa-*
241 *tion and exchange*. Geneva, Switzerland, 2000.



MTConnect[®] Standard

Part 4.4 – QIF Asset Information Model

Version 1.8.0

Prepared for: MTConnect Institute
Prepared on: September 6, 2021

MTConnect[®] is a registered trademark of AMT - The Association for Manufacturing Technology. Use of *MTConnect* is limited to use as specified on <http://www.mtconnect.org/>.

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	5
2.3	MTConnect References	5
3	QIF Asset Information Model	6
3.1	QIFDocumentWrapper	6
3.1.1	Attributes for QIFDocumentWrapper	7
3.1.2	Elements for QIFDocumentWrapper	8
	Appendices	9
A	Bibliography	9

Table of Figures

Figure 1: QIFDocumentWrapper Diagram	7
---	----------

List of Tables

Table 1: Attributes for QIFDocumentWrapper 8

Table 2: Elements for QIFDocumentWrapper 8

1 **1 Purpose of This Document**

2 This document, *MTConnect Standard: Part 4.4 - QIF Asset Information Model* of the
3 MTConnect Standard, establishes the rules and terminology to be used by designers to
4 parse a QIF Document as an *MTConnect Asset* that is provided by an *Agent* from a piece
5 of equipment.

6 The data associated with the QIF Document will be retrieved from multiple sources that
7 are responsible for providing their knowledge of an *MTConnect Asset*.

8 2 Terminology and Conventions

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

12 2.1 Glossary

13 ***Agent***

14 Refers to an MTConnect Agent.

15 Software that collects data published from one or more piece(s) of equipment, orga-
 16 nizes that data in a structured manner, and responds to requests for data from client
 17 software systems by providing a structured response in the form of a *Response Doc-*
 18 *ument* that is constructed using the *semantic data models* defined in the Standard.

19 Appears in the documents in the following form: *Agent*.

20 ***Asset***

21 item, thing or entity that has potential or actual value to an organization *Ref:ISO*
 22 *55000:2014(en)*

23 Note 1 to entry: Value can be tangible or intangible, financial or non-financial,
 24 and includes consideration of risks and liabilities. It can be positive or negative
 25 at different stages of the asset life.

26 Note 2 to entry: Physical assets usually refer to equipment, inventory and prop-
 27 erties owned by the organization. Physical assets are the opposite of intangible
 28 assets, which are non-physical assets such as leases, brands, digital assets, use
 29 rights, licences, intellectual property rights, reputation or agreements.

30 Note 3 to entry: A grouping of assets referred to as an asset system could also
 31 be considered as an asset.

32

33 ***Component***

34 General meaning:

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

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

Used in *Information Models*:

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

- When used as an XML container to organize *Lower Level* `Component` elements.

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

- When used as an abstract XML element. `Component` is replaced in a data model by a type of *Component* element. `Component` is also an XML container used to organize *Lower Level* `Component` elements, *Data Entities*, or both.

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

Current Request

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

Devices Information Model

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

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

Information Model

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

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

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

MTConnect Agent

See definition for *Agent*.

MTConnect Asset

An *MTConnect Asset* is an *Asset* used by the manufacturing process to perform tasks.

Note 1 to entry: An *MTConnect Asset* relies upon an *MTConnect Device* to provide *observations* and information about itself and the *MTConnect Device*

71 revises the information to reflect changes to the *MTConnect Asset* during their
72 interaction. Examples of *MTConnect Assets* are Cutting Tools, Part Information,
73 Manufacturing Processes, Fixtures, and Files.

74 Note 2 to entry: A singular `assetId` uniquely identifies an *MTConnect Asset*
75 throughout its lifecycle and is used to track and relate the *MTConnect Asset* to
76 other *MTConnect Devices* and entities.

77 Note 3 to entry: *MTConnect Assets* are temporally associated with a device and
78 can be removed from the device without damage or alteration to its primary
79 functions.

80

81 ***MTConnect Device***

82 An *MTConnect Device* is a piece of equipment or a manufacturing system that pro-
83 duces *observations* about itself and/or publishes data using the *MTConnect Infor-*
84 *mation Model*.

85 ***MTConnect Information Model***

86 See *Information Model*

87 ***MTConnectDevices Response Document***

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

90 ***MTConnectStreams Response Document***

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

93 ***observation***

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

95 ***Observations Information Model***

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

98 ***Probe Request***

99 A *Probe Request* is a *Request* to an *Agent* to produce an *MTConnectDevices Re-*
100 *sponse Document* containing the *Devices Information Model*.

101 ***Request***

102 A communications method where a client software application transmits a message
103 to an *Agent*. That message instructs the *Agent* to respond with specific information.
104 Appears in the documents in the following form: *Request*.

105 ***Response Document***

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

108 ***Sample Request***

109 A *Sample Request* is a *Request* to an *Agent* to produce an *MTConnectStreams Re-*
110 *sponse Document* containing the *Observations Information Model* for a set of time-
111 stamped *observations* made by *Components*.

112 ***semantic data model***

113 A methodology for defining the structure and meaning for data in a specific logical
114 way.
115 It provides the rules for encoding electronic information such that it can be inter-
116 preted by a software system.
117 Appears in the documents in the following form: *semantic data model*.

118 ***sequence number***

119 The primary key identifier used to manage and locate a specific piece of *Streaming*
120 *Data* in an *Agent*.
121 *sequence number* is a monotonically increasing number within an instance of an
122 *Agent*.
123 Appears in the documents in the following form: *sequence number*.

124 **2.2 Acronyms**

125 ***AMT***

126 The Association for Manufacturing Technology

127 **2.3 MTConnect References**

128 [MTConnect Part 1.0] *MTConnect Standard Part 1.0 - Overview and Fundamentals*. Ver-
129 sion 1.8.0.

130 [MTConnect Part 4.4] *MTConnect Standard: Part 4.4 - QIF Asset Information Model.*
131 Version 1.8.0.

132 3 QIF Asset Information Model

133 The Quality Information Framework (QIF) is an American National Standards Institute
134 (ANSI) accredited standard developed by the Digital Metrology Standards Consortium
135 (DMCS) standards development organization and an A-liaison to the International Stan-
136 dards Organization (ISO) Technical Committee (TC) 184. QIF addresses the needs of the
137 metrology community to have a semantic information model for the exchange of metrol-
138 ogy data throughout the verification lifecycle from product design to execution, analysis,
139 and reporting.

140 The MTConnect QIF Asset Model provides a wrapper around the QIF Information model
141 in its native XML representation utilizing the QIF XML Schema Definition Language
142 (XSDL) references in the wrapper to validate the document. The MTConnect standard
143 does not alter or extend the QIF standard and regards the QIF standard as a pass-through.

144 Information about the QIF standards is at the following location: <https://qifstandards.org>

145 3.1 QIFDocumentWrapper

146 `QIFDocumentWrapper` is an `Asset` that carries the Quality Information Framework
147 (QIF) Document.

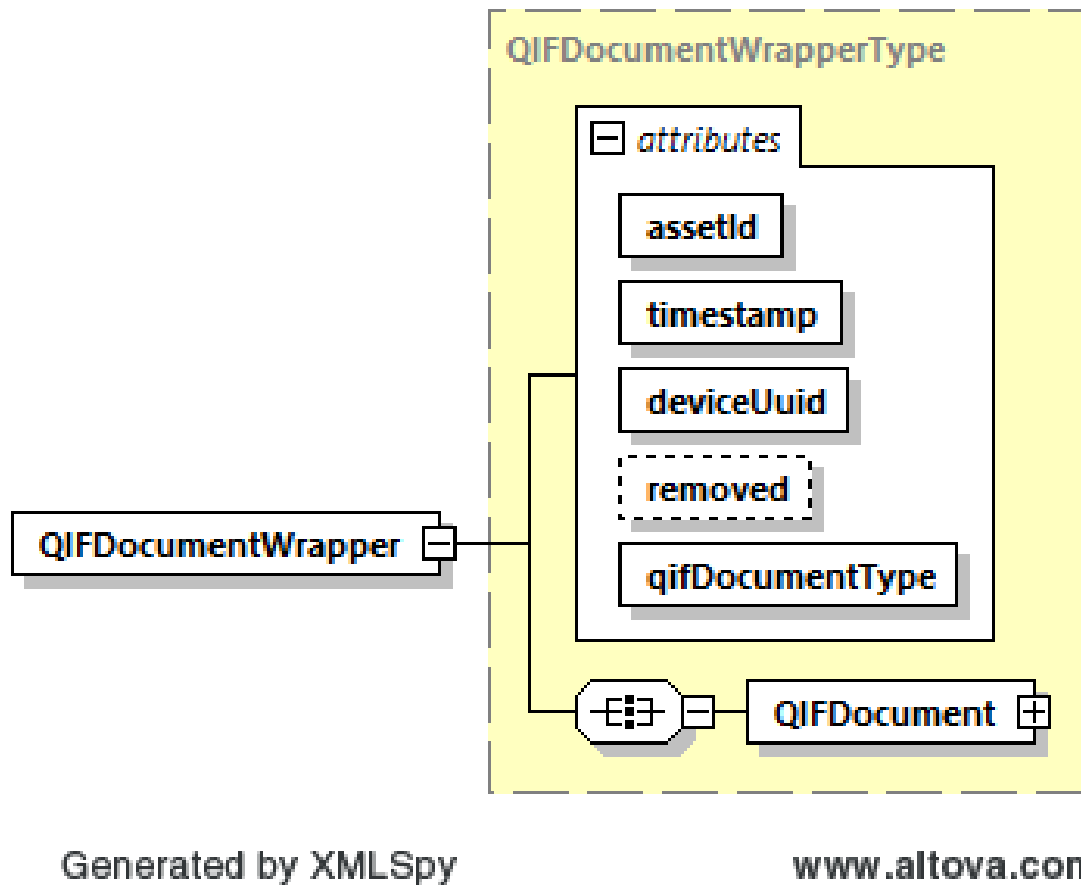


Figure 1: QIFDocumentWrapper Diagram

148 3.1.1 Attributes for QIFDocumentWrapper

149 *Table 1* lists the attributes for an QIFDocumentWrapper element in addition to at-
 150 tributes inherited from Asset element.

Table 1: Attributes for QIFDocumentWrapper

Attribute	Description	Occurrence
qifDocumentType	<p>The contained QIF Document type as defined in the QIF Standard.</p> <p>The value of qifDocumentType MUST be one of the current documents types as per QIF: MEASUREMENT_RESOURCE, PLAN, PRODUCT, RESULTS, RULES or STATISTICS.</p>	0..1

151 3.1.2 Elements for QIFDocumentWrapper

152 *Table 2* lists the elements for an QIFDocumentWrapper element.

Table 2: Elements for QIFDocumentWrapper

Element	Description	Occurrence
QIFDocument	The QIF Document as defined by the QIF standard.	1

153 Appendices

154 A Bibliography

- 155 Engineering Industries Association. *EIA Standard - EIA-274-D*, Interchangeable Variable,
 156 Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically
 157 Controlled Machines. Washington, D.C. 1979.
- 158 ISO TC 184/SC4/WG3 N1089. *ISO/DIS 10303-238*: Industrial automation systems and
 159 integration Product data representation and exchange Part 238: Application Protocols: Ap-
 160 plication interpreted model for computerized numerical controllers. Geneva, Switzerland,
 161 2004.
- 162 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
 163 tems and integration – Physical device control – Data model for computerized numerical
 164 controllers – Part 10: General process data. Geneva, Switzerland, 2004.
- 165 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
 166 tems and integration – Physical device control – Data model for computerized numerical
 167 controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.
- 168 International Organization for Standardization. *ISO 6983/1* – Numerical Control of ma-
 169 chines – Program format and definition of address words – Part 1: Data format for posi-
 170 tioning, line and contouring control systems. Geneva, Switzerland, 1982.
- 171 Electronic Industries Association. *ANSI/EIA-494-B-1992*, 32 Bit Binary CL (BCL) and
 172 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines.
 173 Washington, D.C. 1992.
- 174 National Aerospace Standard. *Uniform Cutting Tests* - NAS Series: Metal Cutting Equip-
 175 ment Specifications. Washington, D.C. 1969.
- 176 International Organization for Standardization. *ISO 10303-11*: 1994, Industrial automa-
 177 tion systems and integration Product data representation and exchange Part 11: Descrip-
 178 tion methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.
- 179 International Organization for Standardization. *ISO 10303-21*: 1996, Industrial automa-
 180 tion systems and integration – Product data representation and exchange – Part 21: Imple-
 181 mentation methods: Clear text encoding of the exchange structure. Geneva, Switzerland,
 182 1996.
- 183 H.L. Horton, F.D. Jones, and E. Oberg. *Machinery's Handbook*. Industrial Press, Inc.

- 184 New York, 1984.
- 185 International Organization for Standardization. *ISO 841-2001: Industrial automation sys-*
186 *tems and integration - Numerical control of machines - Coordinate systems and motion*
187 *nomenclature*. Geneva, Switzerland, 2001.
- 188 *ASME B5.59-2 Version 9c: Data Specification for Properties of Machine Tools for Milling*
189 *and Turning*. 2005.
- 190 *ASME/ANSI B5.54: Methods for Performance Evaluation of Computer Numerically Con-*
191 *trolled Machining Centers*. 2005.
- 192 OPC Foundation. *OPC Unified Architecture Specification, Part 1: Concepts Version 1.00*.
193 July 28, 2006.
- 194 International Organization for Standardization. *ISO 13399: Cutting tool data representa-*
195 *tion and exchange*. Geneva, Switzerland, 2000.



MTConnect[®] Standard

Part 5 – Interfaces

Version 1.8.0

Prepared for: MTConnect Institute
Prepared on: September 6, 2021

MTConnect[®] is a registered trademark of AMT - The Association for Manufacturing Technology. Use of *MTConnect* is limited to use as specified on <http://www.mtconnect.org/>.

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	7
3	Interfaces Overview	8
3.1	Interfaces Architecture	8
3.2	Request and Response Information Exchange	10
4	Interfaces for Devices and Streams Information Models	13
4.1	Interfaces	14
4.2	Interface	14
4.2.1	XML Schema Structure for Interface	14
4.2.2	Interface Types	16
4.2.3	Data for Interface	18
4.2.3.1	References for Interface	18
4.2.4	Data Items for Interface	19
4.2.4.1	INTERFACE_STATE for Interface	19
4.2.4.2	Specific Data Items for the Interaction Model for Interface	20
4.2.4.3	Event States for Interfaces	22
5	Operation and Error Recovery	27
5.1	Request/Response Failure Handling and Recovery	27
	Appendices	35
A	Bibliography	35

Table of Figures

Figure 1: Data Flow Architecture for Interfaces	9
Figure 2: Request and Response Overview	11
Figure 3: Interfaces as a Structural Element	13
Figure 4: Interface Schema	15
Figure 5: Request State Diagram	23
Figure 6: Response State Diagram	26
Figure 7: Success Scenario	27
Figure 8: Responder - Immediate Failure	28
Figure 9: Responder Fails While Providing a Service	29
Figure 10:Requester Fails During a Service Request	30
Figure 11:Requester Makes Unexpected State Change	31
Figure 12:Responder Makes Unexpected State Change	32
Figure 13:Requester/Responder Communication Failures	33

List of Tables

Table 1: Sequence of interaction between pieces of equipment 11

Table 2: Interface types 16

Table 3: InterfaceState Event 20

Table 4: Event Data Item types for Interface 21

Table 5: Request States 22

Table 6: Response States 24

1 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 nectStreams 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 docu-
21 ments 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 XML

37 Stands for eXtensible Markup Language.

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

40 XML is the language used for all code examples in the MTConnect Standard.

41 Refer to <http://www.w3.org/XML> for more information about XML.

42 *Agent*

43 Refers to an MTConnect Agent.

44 Software that collects data published from one or more piece(s) of equipment, orga-
 45 nizes that data in a structured manner, and responds to requests for data from client
 46 software systems by providing a structured response in the form of a *Response Doc-*
 47 *ument* that is constructed using the *semantic data models* defined in the Standard.

48 Appears in the documents in the following form: *Agent*.

49 *Child Element*

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

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

53 **Component**

54 General meaning:

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

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

58 Used in *Information Models*:

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

- 61 • When used as an XML container to organize *Lower Level* `Component` ele-
62 ments.

63 Appears in the documents in the following form: `Components`.

- 64 • When used as an abstract XML element. `Component` is replaced in a data
65 model by a type of *Component* element. `Component` is also an XML con-
66 tainer used to organize *Lower Level* `Component` elements, *Data Entities*, or
67 both.

68 Appears in the documents in the following form: `Component`.

69 **Controlled Vocabulary**

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

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

73 **Current Request**

74 A *Current Request* is a *Request* to an *Agent* to produce an *MTConnectStreams Re-*
75 *sponse Document* containing the *Observations Information Model* for a snapshot of
76 the latest *observations* at the moment of the *Request* or at a given *sequence number*.

77 **Data Entity**

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

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

82 **Devices Information Model**

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

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

86 ***Element Name***

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

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

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

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

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

94 ***Equipment Metadata***

95 See *Metadata*

96 ***Information Model***

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

99 For example, an information model is used to define the structure for each *MTCon-*
100 *nect Response Document*; the definition of each piece of information within those
101 documents and the relationship between pieces of information.

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

103 ***Interaction Model***

104 Defines how information is exchanged across an *Interface* between independent sys-
105 tems.

106 ***Interface***

107 The means by which communication is achieved between independent systems.

108 ***Lower Level***

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

110 ***Metadata***

111 Data that provides information about other data.

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

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

117 ***MTConnect Agent***

118 See definition for *Agent*.

119 ***MTConnectDevices Response Document***

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

122 ***MTConnectStreams Response Document***

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

125 ***observation***

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

127 ***Observations Information Model***

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

130 ***Parent Element***

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

133 Appears in the documents in the following form: *Parent Element*.

134 ***Probe Request***

135 A *Probe Request* is a *Request* to an *Agent* to produce an *MTConnectDevices Re-*
136 *sponse Document* containing the *Devices Information Model*.

137 ***Publish/Subscribe***

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

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

144 ***Request***

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

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

148 ***Requester***

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

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

151 ***Responder***

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

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

154 ***Response Document***

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

157 ***Sample Request***

158 A *Sample Request* is a *Request* to an *Agent* to produce an *MTConnectStreams Re-*
159 *sponse Document* containing the *Observations Information Model* for a set of time-
160 stamped *observations* made by *Components*.

161 ***semantic data model***

162 A methodology for defining the structure and meaning for data in a specific logical
163 way.

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

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

167 ***sequence number***

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

170 *sequence number* is a monotonically increasing number within an instance of an
171 *Agent*.

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

173 ***Streaming Data***

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

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

177 ***Structural Element***

178 General meaning:

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

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

Used to indicate hierarchy of Components:

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

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

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

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

Top Level

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

Valid Data Value

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

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

2.2 Acronyms

AMT

The Association for Manufacturing Technology

2.3 MTConnect References

[MTConnect Part 1.0] *MTConnect Standard Part 1.0 - Overview and Fundamentals*. Version 1.8.0.

[MTConnect Part 2.0] *MTConnect Standard: Part 2.0 - Devices Information Model*. Version 1.8.0.

[MTConnect Part 3.0] *MTConnect Standard: Part 3.0 - Streams Information Model*. Version 1.8.0.

[MTConnect Part 5.0] *MTConnect Standard: Part 5.0 - Interfaces*. Version 1.8.0.

207 3 Interfaces Overview

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

213 In the MTConnect Standard, *Interfaces* provides a means to replace this traditional method
 214 for interconnecting pieces of equipment with a structured *Interaction Model* that provides
 215 a rich set of information used to coordinate the actions between pieces of equipment. Im-
 216 plementers may utilize the information provided by this data model to (1) realize the inter-
 217 action between pieces of equipment and (2) to extend the functionality of the equipment
 218 to improve the overall performance of the manufacturing process.

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

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

- 228 • An *Agent* which provides:
 - 229 - The data required to implement the *Interaction Model*.
 - 230 - Any other data from a piece of equipment needed to implement the *Interface*
 - 231 – operating states of the equipment, position information, execution modes, process
 - 232 information, etc.
- 233 • A client software application that enables the piece of equipment to acquire and
- 234 interpret information from another piece of equipment.

235 3.1 Interfaces Architecture

236 MTConnect Standard is based on a communications method that provides no direct way
 237 for one piece of equipment to change the state of or cause an action to occur in another

piece of equipment. The *Interaction Model* used to implement *Interfaces* is based on a *Publish/Subscribe* type of communications as described in *MTConnect Standard Part 1.0 - Overview and Fundamentals* and utilizes a *Request* and *Response* information exchange mechanism. For *Interfaces*, pieces of equipment must perform both the publish (*Agent*) and subscribe (client) functions.

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

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

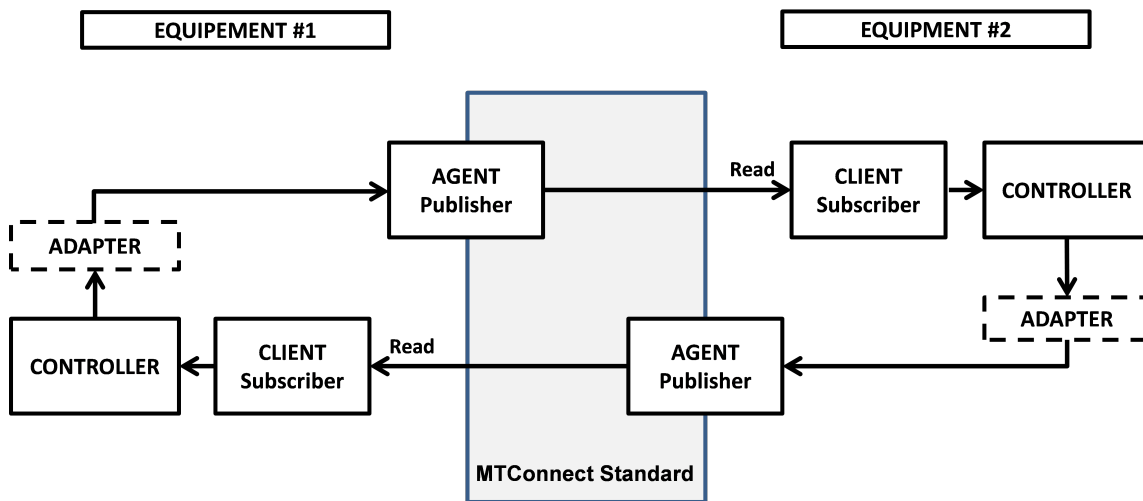


Figure 1: Data Flow Architecture for Interfaces

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

In the implementation of the *Interaction Model* for *Interfaces*, two pieces of equipment can exchange information in the following manner. One piece of equipment indicates a *Request* for service by publishing a type of *Request* using a data item provided through an *Agent* as defined in *Section 4 - Interfaces for Devices and Streams Information Models*. The client associated with the second piece of equipment, which is subscribing to data from the first machine, detects and interprets that *Request*. If the second machine chooses to take any action to fulfill this *Request*, it can indicate its acceptance by publishing a *Response* using a data item provided through its *Agent*. The client on the first piece of equipment continues to monitor information from the second piece of equipment until it detects an indication that the *Response* to the *Request* has been completed or has failed.

An example of this type of interaction between pieces of equipment can be represented

261 by a machine tool that wants the material to be loaded by a robot. In this example, the
 262 machine tool is the *Requester*, and the robot is the *Responder*. On the other hand, if the
 263 robot wants the machine tool to open a door, the robot becomes the *Requester* and the
 264 machine tool the *Responder*.

265 3.2 Request and Response Information Exchange

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

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

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

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

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

294 Note: The implementation details of how the *Responder* piece of equipment reacts to
 295 the *Request* and then completes the requested task are up to the implementer.

296 *Figure 2* provides an example of the *Request* and *Response* state machine:

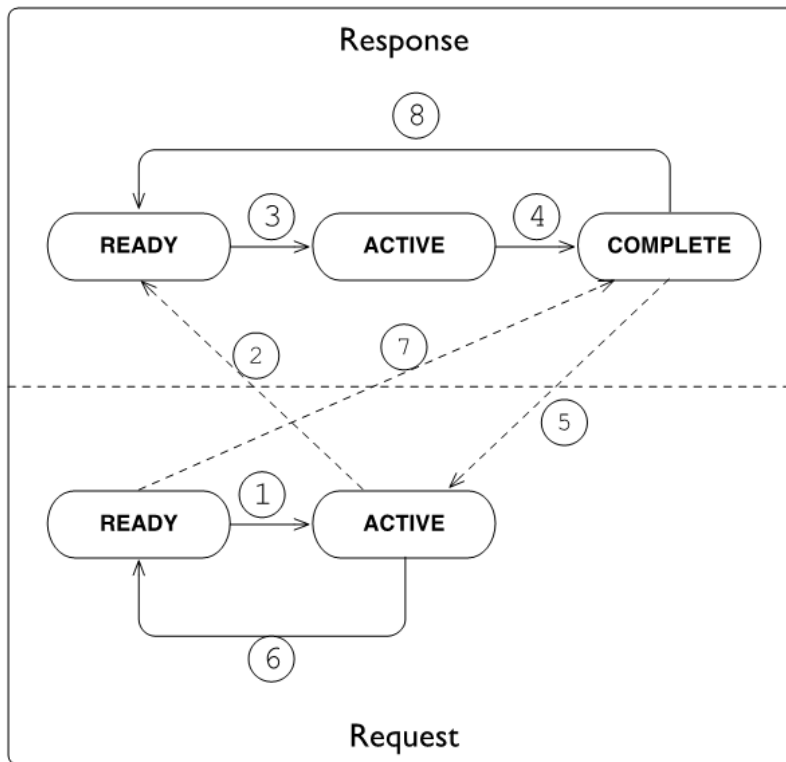


Figure 2: Request and Response Overview

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

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

Table 1: Sequence of interaction between pieces of equipment

Step	Description
1	The <i>Request</i> transitions from <i>READY</i> to <i>ACTIVE</i> 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 <i>READY</i> to <i>ACTIVE</i> indicating that it is performing the action.
4	Once the action has been performed, the <i>Response</i> transitions to <i>COMPLETE</i> .

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.

302 After the final action has been completed, both pieces of equipment are back in the READY
303 state indicating that they are able to perform another action.

304 4 Interfaces for Devices and Streams Information Models

305 The *Interaction Model* for implementing *Interfaces* is defined in the MTConnect Standard
 306 as an extension to the MTConnectDevices and MTConnectStreams XML docu-
 307 ments.

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

315 The XML tree in *Figure 3* illustrates where *Interfaces* is modeled in the *Devices Informa-*
 316 *tion Model* for a piece of equipment.

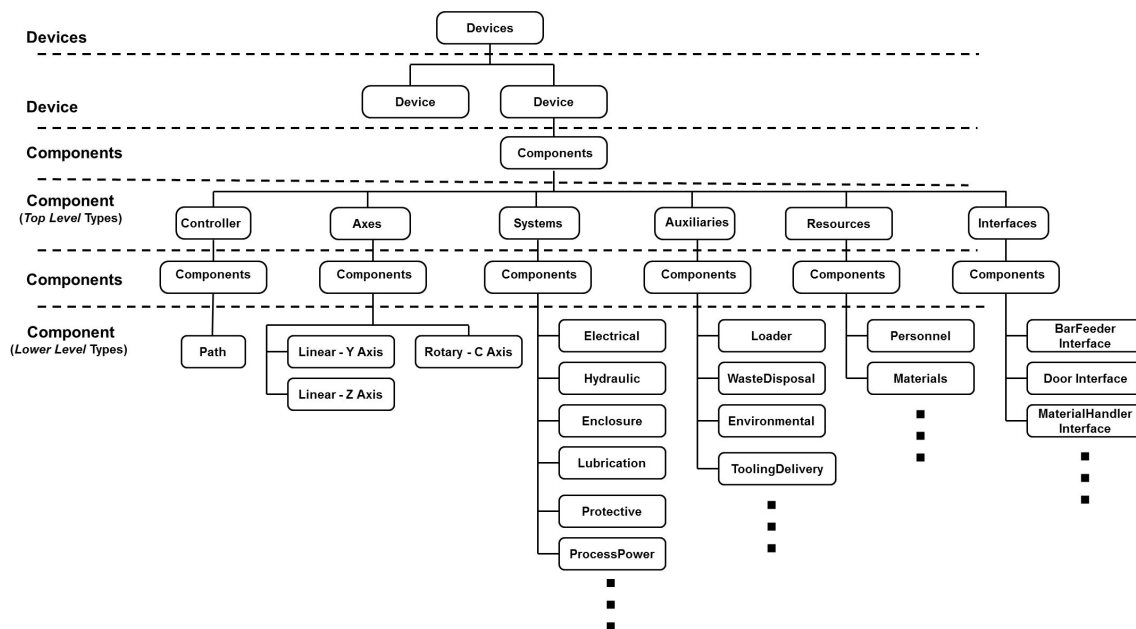


Figure 3: Interfaces as a Structural Element

317 4.1 Interfaces

318 *Interfaces* is an XML *Structural Element* in the MTConnectDevices XML document.
 319 *Interfaces* is a container type XML element. *Interfaces* is used to group information de-
 320 scribing *Lower Level* Interface XML elements, which each provide information for
 321 an individual *Interface*.

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

324 4.2 Interface

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

328 Each Interface is also a container type element. As a container, the Interface
 329 XML element is used to organize information required to implement the *Interaction Model*
 330 for an *Interface*. It also provides structure for describing the *Lower Level Structural Ele-*
 331 *ments* associated with the Interface. Each Interface contains *Data Entities* avail-
 332 able from the piece of equipment that may be needed to coordinate activities with associ-
 333 ated pieces of equipment.

334 The information provided by a piece of equipment for each *Interface* is returned in a Com-
 335 ponentStream container of an MTConnectStreams document in the same manner
 336 as all other types of components.

337 4.2.1 XML Schema Structure for Interface

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

339 The schema for an Interface element is the same as defined for Component elements
 340 described in Section 4.4 in *MTConnect Standard: Part 2.0 - Devices Information Model*
 341 of the MTConnect Standard. The *Figure 4* shows the attributes defined for Interface
 342 and the elements that may be associated with Interface.

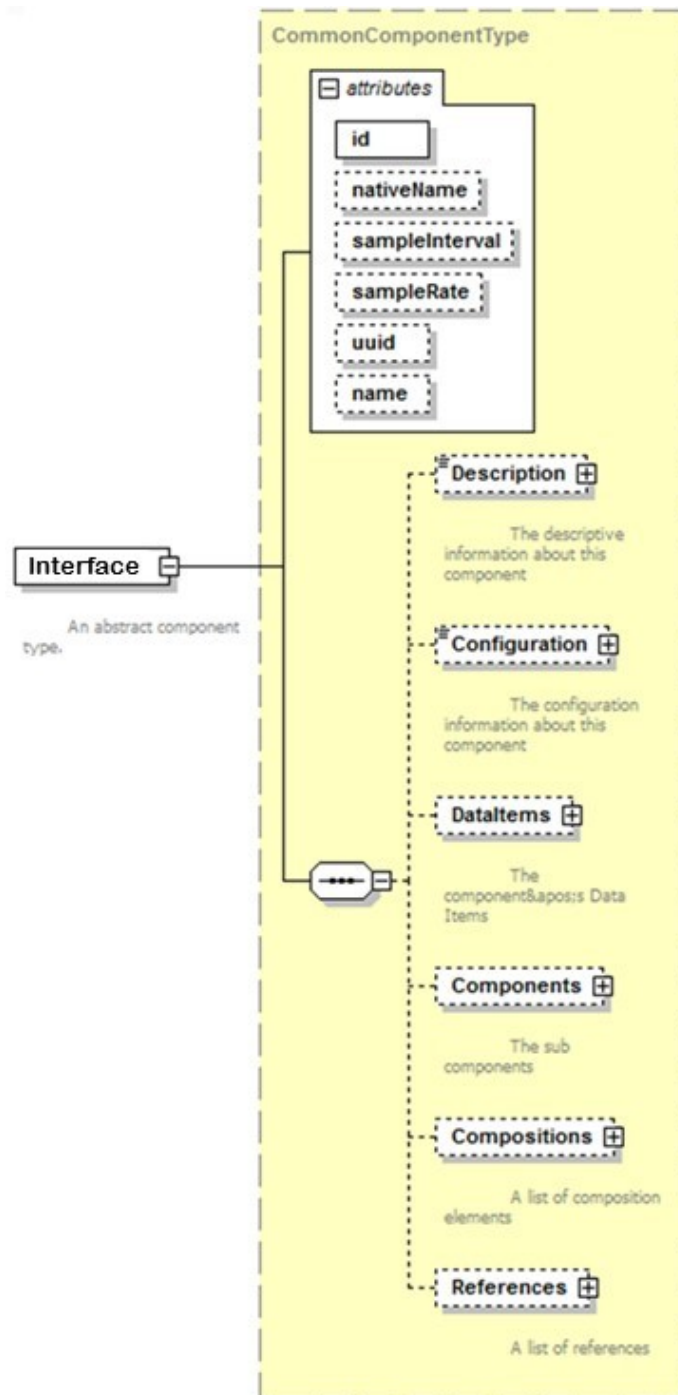


Figure 4: Interface Schema

343 Refer to *MTConnect Standard: Part 2.0 - Devices Information Model*, Section 4.4 for
 344 complete descriptions of the attributes and elements that are illustrated in the *Figure 4* for
 345 *Interface*.

346 4.2.2 Interface Types

347 As an abstract type XML element, *Interface* is replaced in the *MTConnectDevices*
 348 document with a XML element representing a specific type of *Interface*. An initial list of
 349 *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>

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

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

355 4.2.3 Data for Interface

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

360 Details on data items specific to the *Interaction Model* for each type of *Interface* are pro-
361 vided in *Section 4.2.4 - Data Items for Interface*.

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

364 4.2.3.1 References for Interface

365 Some of the data items needed to support a specific *Interface* may already be defined else-
366 where in the XML document for a piece of equipment. However, the implementer may
367 not be able to directly associate this data with the *Interface* since the MTConnect Standard
368 does not permit multiple occurrences of a piece of data to be configured in a XML docu-
369 ment. *References* provides a mechanism for associating information defined elsewhere

370 in the *Information Model* for a piece of equipment with a specific *Interface*.

371 *References* is an XML container that organizes pointers to information defined else-
 372 where in the XML document for a piece of equipment. *References* **MAY** contain one
 373 or more *Reference* XML elements.

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

377 *References* is an economical syntax for providing interface specific information with-
 378 out directly duplicating the occurrence of the data. It provides a mechanism to include all
 379 necessary information required for interaction and deterministic information flow between
 380 pieces of equipment.

381 For more information on the definition for *References* and *Reference*, see Section
 382 4.7 and 4.8 of *MTConnect Standard: Part 2.0 - Devices Information Model*.

383 4.2.4 Data Items for Interface

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

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

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

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

394 4.2.4.1 INTERFACE_STATE for Interface

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

398 An *INTERFACE_STATE* data item **MUST** be defined for every *Interface* XML ele-

399 ment.

400 INTERFACE_STATE is reported in the MTConnectStreams XML document as In-
401 terfaceState. InterfaceState reports one of two states – ENABLED or DIS-
402 ABLED, which are provided in the CDATA for InterfaceState.

403 The *Table 3* shows both the INTERFACE_STATE data item as defined in the MTCon-
404 nectDevices document and the corresponding *Element Name* that **MUST** be reported
405 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 Interface type element indicating whether the <i>Interface</i> is active or not currently functioning.</p> <p><i>Valid Data Values:</i></p> <p>ENABLED: The <i>Interface</i> is currently operational and performing as expected.</p> <p>DISABLED: The <i>Interface</i> is currently not operational.</p> <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>

406 4.2.4.2 Specific Data Items for the Interaction Model for Interface

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

the associated piece of equipment to indicate its progress in performing its *Response* to the *Request* for service.

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

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

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

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

The *Table 4* also provides the corresponding transformed *Element Name* for each data item that **MAY** be returned by an *Agent* as an *Event* type XML *Data Entity* in the `MTConnectStreams` XML document. The *Controlled Vocabulary* for each of these data items are defined in *Section 4.2.4.3 - Event States for Interfaces*.

Table 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.

4.2.4.3 Event States for Interfaces

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

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

Table 5: Request States

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 <code>FAIL</code>.</p> <p>CONDITION 2:</p> <p>If the <i>Responder</i> changes its state to <code>FAIL</code>, the <i>Requester</i> MUST change its state to <code>FAIL</code>.</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 <code>FAIL</code> state by changing its state to <code>FAIL</code>.</p> <p>Once the <code>FAIL</code> state has been acknowledged by the <i>Responder</i>, the <i>Requester</i> may attempt to clear its <code>FAIL</code> state.</p> <p>As part of the attempt to clear the <code>FAIL</code> 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 <i>Request</i> state from <code>FAIL</code> to <code>READY</code>. If for some reason the <i>Requester</i> is not again prepared to perform a service, it transitions its state from <code>FAIL</code> to <code>NOT_READY</code>.</p>

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

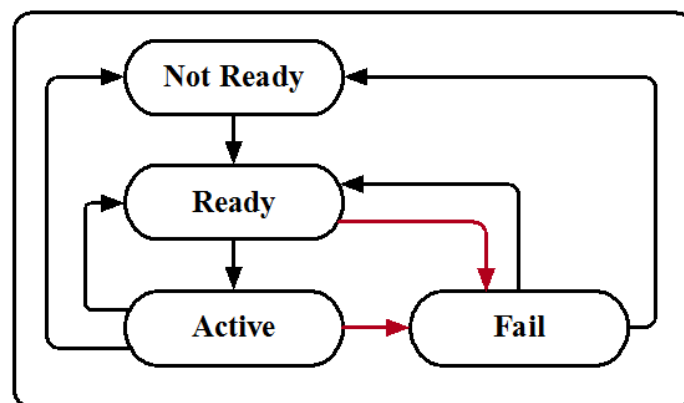


Figure 5: Request State Diagram

445 The *Response* portion of the *Interaction Model* for *Interfaces* has five states as defined in
 446 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>

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

449 *Figure 6* shows a graphical representation of the possible state transitions for a *Response*:

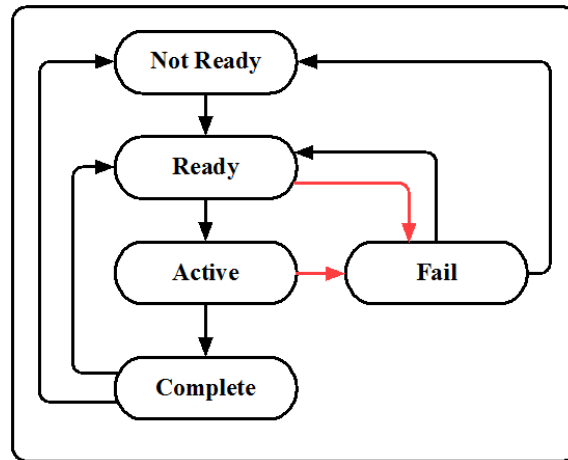


Figure 6: Response State Diagram

450 5 Operation and Error Recovery

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

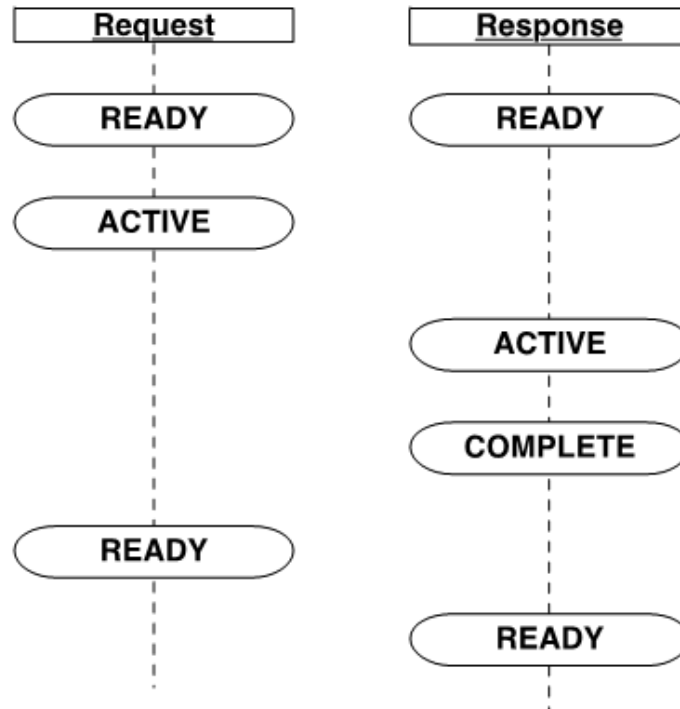


Figure 7: Success Scenario

455 5.1 Request/Response Failure Handling and Recovery

456 A significant feature of the *Request/Response Interaction Model* is the ability for either
 457 piece of equipment to detect a failure associated with either the *Request* or *Response* ac-
 458 tions. When either a failure or unexpected action occurs, the *Request* and the *Response*
 459 portion of the *Interaction Model* can announce a FAIL state upon detecting a problem. The
 460 following are graphical models describing multiple scenarios where either the *Requester*
 461 or *Responder* detects and reacts to a failure. In these examples, either the *Requester* or *Re-*
 462 sponder announces the detection of a failure by setting either the *Request* or the *Response*
 463 state to FAIL.

464 Once a failure is detected, the *Interaction Model* provides information from each piece of

465 equipment as they attempt to recover from a failure, reset all of their functions associated
 466 with the *Interface* to their original state, and return to normal operation.

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

470 Scenario #1 – Responder Fails Immediately

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

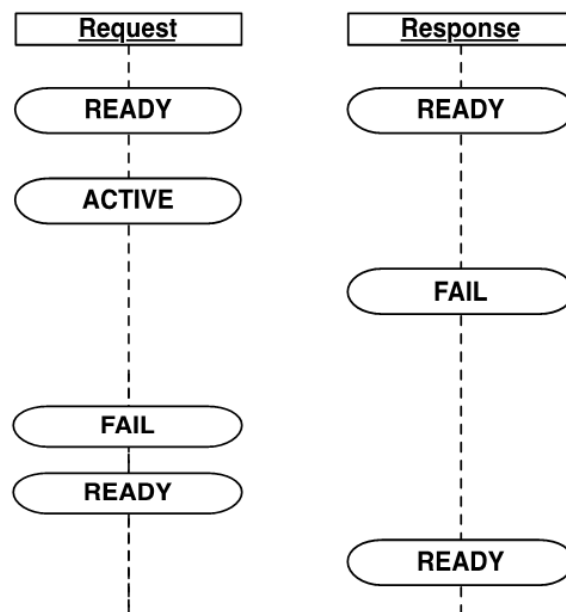


Figure 8: Responder - Immediate Failure

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

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

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

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

488 Scenario #2 – Responder Fails While Providing a Service

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

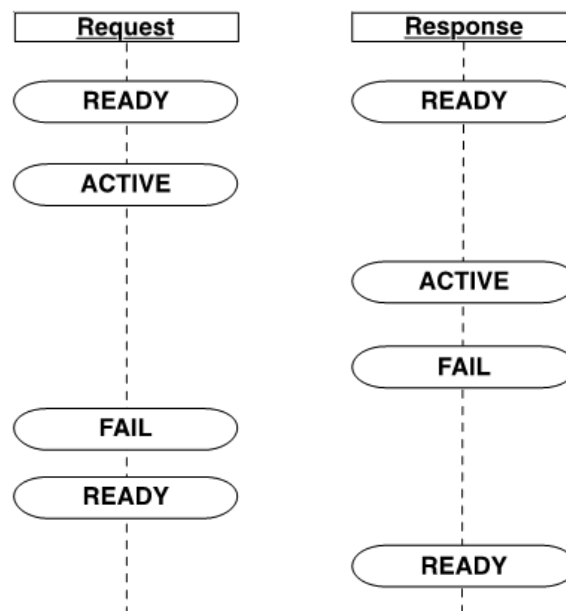


Figure 9: Responder Fails While Providing a Service

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

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

500 The *Responder*, as part of clearing a failure, resets any partial actions that were initiated
 501 and attempts to return to a condition where it is again ready to perform a service. If the
 502 recovery is successful, the *Responder* changes its *Response* state from FAIL to READY if

503 it is again prepared to perform a service. If for some reason the *Responder* is not again
 504 prepared to perform a service, it transitions its state from `FAIL` to `NOT_READY`.

505 Scenario #3 – Requester Failure During a Service Request

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

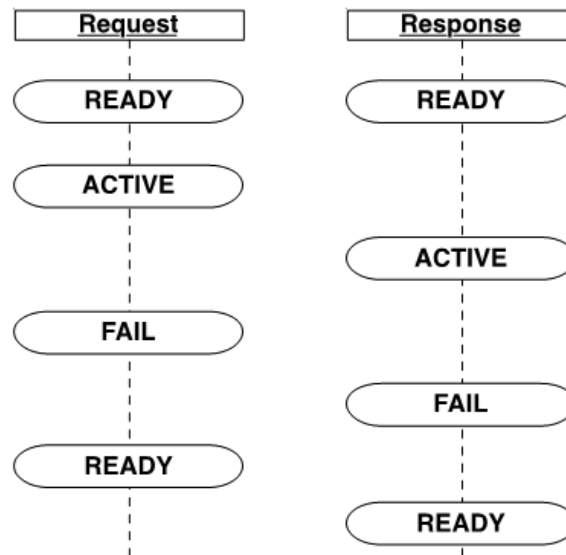


Figure 10: Requester Fails During a Service Request

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

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

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

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

522 In some cases, a *Requester* may transition to an unexpected state after it has initiated a

523 *Request* for service.

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

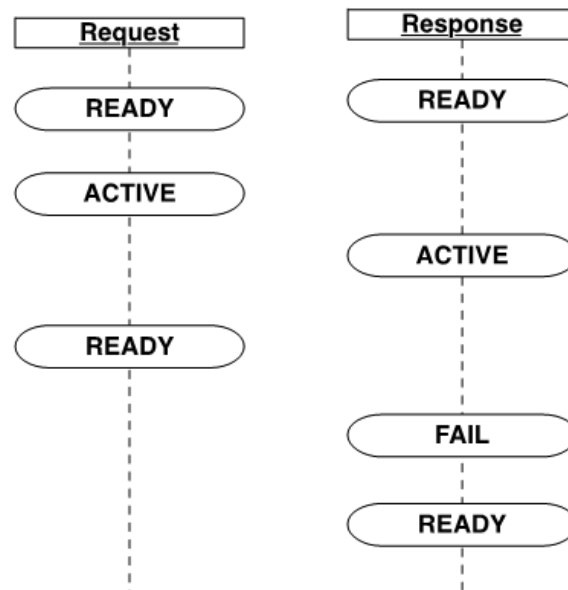


Figure 11: Requester Makes Unexpected State Change

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

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

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

538 Note: The same scenario exists if the *Requester* transitions its *Request* state to **NOT_**–
 539 **READY**. However, in this case, the *Requester* then transitions its *Request* state
 540 to **READY** after it resets all of its functions back to a condition where it is again
 541 prepared to make a *Request* for service.

542 Scenario #5 – Responder Changes to an Unexpected State While Providing a Service

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

545 As demonstrated in Figure 12, the *Responder* is performing the actions to provide a ser-
 546 vice and the *Response* state is ACTIVE. During these actions, the *Responder* transitions its
 547 state to NOT_READY before completing its actions. This should be regarded as a failure
 548 of the *Responder*.

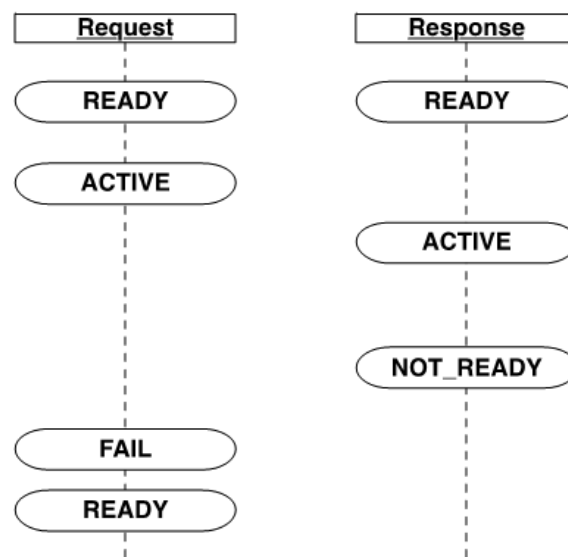


Figure 12: Responder Makes Unexpected State Change

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

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

556 Since the *Responder* has failed to an invalid state, the condition of the *Responder* is un-
 557 known. Where possible, the *Responder* should try to reset to an initial state.

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

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

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

565 In this scenario, a failure occurs in the communications connection between the *Responder*
 566 and *Requester*. This failure may result from the *InterfaceState* from either piece of
 567 equipment returning a value of UNAVAILABLE or one of the pieces of equipment does
 568 not provide a heartbeat within the desired amount of time (See *MTConnect Standard Part*
 569 *1.0 - Overview and Fundamentals* for details on heartbeat).

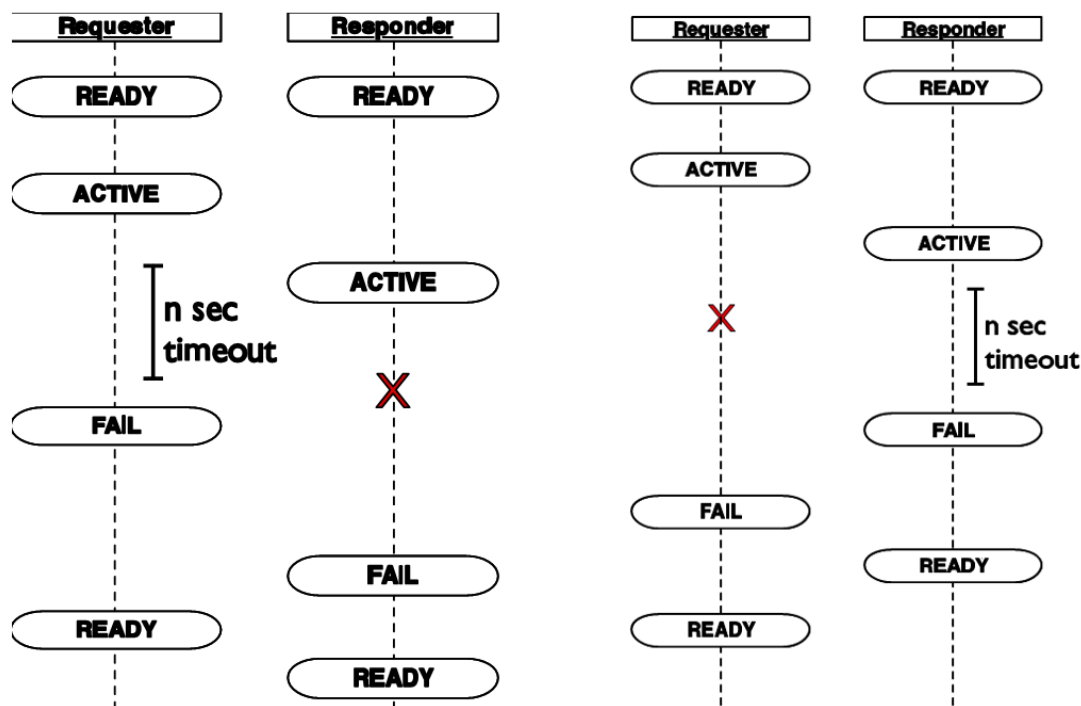


Figure 13: Requester/Responder Communication Failures

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

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

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

579 a service, it transitions its state from `FAIL` to `NOT_READY`.

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

585 Appendices

586 A Bibliography

- 587 Engineering Industries Association. *EIA Standard - EIA-274-D*, Interchangeable Variable,
 588 Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically
 589 Controlled Machines. Washington, D.C. 1979.
- 590 ISO TC 184/SC4/WG3 N1089. *ISO/DIS 10303-238*: Industrial automation systems and
 591 integration Product data representation and exchange Part 238: Application Protocols: Ap-
 592 plication interpreted model for computerized numerical controllers. Geneva, Switzerland,
 593 2004.
- 594 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
 595 tems and integration – Physical device control – Data model for computerized numerical
 596 controllers – Part 10: General process data. Geneva, Switzerland, 2004.
- 597 International Organization for Standardization. *ISO 14649*: Industrial automation sys-
 598 tems and integration – Physical device control – Data model for computerized numerical
 599 controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.
- 600 International Organization for Standardization. *ISO 6983/1* – Numerical Control of ma-
 601 chines – Program format and definition of address words – Part 1: Data format for posi-
 602 tioning, line and contouring control systems. Geneva, Switzerland, 1982.
- 603 Electronic Industries Association. *ANSI/EIA-494-B-1992*, 32 Bit Binary CL (BCL) and
 604 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines.
 605 Washington, D.C. 1992.
- 606 National Aerospace Standard. *Uniform Cutting Tests* - NAS Series: Metal Cutting Equip-
 607 ment Specifications. Washington, D.C. 1969.
- 608 International Organization for Standardization. *ISO 10303-11*: 1994, Industrial automa-
 609 tion systems and integration Product data representation and exchange Part 11: Descrip-
 610 tion methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.
- 611 International Organization for Standardization. *ISO 10303-21*: 1996, Industrial automa-
 612 tion systems and integration – Product data representation and exchange – Part 21: Imple-
 613 mentation methods: Clear text encoding of the exchange structure. Geneva, Switzerland,
 614 1996.
- 615 H.L. Horton, F.D. Jones, and E. Oberg. *Machinery's Handbook*. Industrial Press, Inc.

- 616 New York, 1984.
- 617 International Organization for Standardization. *ISO 841-2001: Industrial automation sys-*
618 *tems and integration - Numerical control of machines - Coordinate systems and motion*
619 *nomenclature*. Geneva, Switzerland, 2001.
- 620 ASME B5.57: *Methods for Performance Evaluation of Computer Numerically Controlled*
621 *Lathes and Turning Centers*, 1998.
- 622 ASME/ANSI B5.54: *Methods for Performance Evaluation of Computer Numerically Con-*
623 *trolled Machining Centers*. 2005.
- 624 OPC Foundation. *OPC Unified Architecture Specification, Part 1: Concepts Version 1.00*.
625 July 28, 2006.
- 626 IEEE STD 1451.0-2007, *Standard for a Smart Transducer Interface for Sensors and Ac-*
627 *tuators – Common Functions, Communication Protocols, and Transducer Electronic Data*
628 *Sheet (TEDS) Formats*, IEEE Instrumentation and Measurement Society, TC-9, The In-
629 *stitute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH99684,*
630 *October 5, 2007.*
- 631 IEEE STD 1451.4-1994, *Standard for a Smart Transducer Interface for Sensors and Ac-*
632 *tuators – Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet*
633 *(TEDS) Formats*, IEEE Instrumentation and Measurement Society, TC-9, The Institute of
634 *Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH95225, December*
635 *15, 2004.*