



MTConnect[®] Standard

Part 1.0 – Fundamentals

Version 2.0.0

Prepared for: MTConnect Institute

Prepared from: MTConnectSysMLModel.xml

Prepared on: May 24, 2022

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.

The normative XMI is located at the following URL: MTConnectSysMLModel.xml

Table of Contents

1	Overview of MTConnect	2
2	Purpose of This Document	7
3	Terminology and Conventions	8
3.1	General Terms	8
3.2	Information Model Terms	14
3.3	Protocol Terms	15
3.4	HTTP Terms	17
3.5	XML Terms	19
3.6	MTConnect Terms	20
3.7	Acronyms	21
3.8	MTConnect References	33
4	Fundamentals	34
4.1	Agent	34
4.1.1	Agent Instance ID	34
4.1.2	Storage of Equipment Metadata	35
4.1.3	Storage of Streaming Data	35
4.1.4	Storage of MTConnect Assets	40
4.2	Response Documents	41
4.3	Request/Response Information Exchange	42
5	MTConnect Protocol	43
5.1	REST Protocol	43
5.1.1	HTTP Request	43
5.1.2	MTConnect REST API	44
5.1.3	HTTP Errors	45
5.1.4	Agent	48
5.2	MTConnectDevices Response Document	57
5.2.1	MTConnectDevices	58
5.2.2	Header	59
5.2.3	<<deprecated>>AssetCount	61
5.3	MTConnectStreams Response Document	62
5.3.1	MTConnectStreams	62
5.3.2	Header	64
5.4	MTConnectAssets Response Document	66
5.4.1	MTConnectAssets	66
5.4.2	Header	67
6	Error Information Model	70

6.1	MTConnectErrors Response Document	70
6.1.1	MTConnectError	70
6.1.2	Header	71
6.1.3	Error	74
7	Profile	76
7.1	DataTypes	76
7.1.1	boolean	76
7.1.2	ID	76
7.1.3	string	76
7.1.4	float	76
7.1.5	dateTime	77
7.1.6	integer	77
7.1.7	xlinktype	77
7.1.8	xslang	77
7.1.9	SECOND	77
7.1.10	IDREF	77
7.1.11	xlinkhref	77
7.1.12	x509	78
7.1.13	int32	78
7.1.14	int64	78
7.1.15	version	78
7.1.16	uInt32	78
7.1.17	uInt64	78
7.2	Stereotypes	78
7.2.1	organizer	78
7.2.2	deprecated	79
7.2.3	extensible	79
7.2.4	informative	79
7.2.5	valueType	79
7.2.6	normative	79
7.2.7	observes	79
	Appendices	81
A	Bibliography	81
B	Fundamentals of Using XML to Encode Response Documents	83
C	Schema and Namespace Declaration Information	86
D	Extensibility	88

Table of Figures

Figure 1: Basic MTConnect Implementation Structure	4
Figure 2: Data Storage in Buffer	35
Figure 3: First In First Out Buffer Management	35
Figure 4: Identifying the range of data with firstSequence and lastSequence .	37
Figure 5: Identifying the range of data with from and count	37
Figure 6: Identifying the range of data with nextSequence and lastSequence .	37
Figure 7: First In First Out Asset Buffer Management	40
Figure 8: Relationship between assetId and stored Asset documents	40
Figure 9: MTConnectDevices	58
Figure 10:MTConnectStreams	63
Figure 11:MTConnectAssets	66
Figure 12:MTConnectError	70
Figure 13:DataTypes	76
Figure 14:Stereotypes	80

List of Tables

Table 1: instanceId and sequence	36
Table 2: Data Storage Concept	38
Table 3: Value Properties of Agent	48
Table 4: Part Properties of Agent	49
Table 5: Part Properties of MTConnectDevices	58
Table 6: Value Properties of Header	59
Table 7: Value Properties of AssetCount	62
Table 8: Part Properties of MTConnectStreams	62
Table 9: Value Properties of Header	64
Table 10: Part Properties of MTConnectAssets	67
Table 11: Value Properties of Header	68
Table 12: Part Properties of MTConnectError	71
Table 13: Value Properties of Header	72
Table 14: Value Properties of Error	74

1 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 model* 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 and*
12 *response* and *publish and subscribe* type of communications. The *request and response*
13 communications structure is used throughout this document to describe the functionality
14 provided by MTConnect. See *Section 5.1.3.1 - Streaming Data* for details describing the
15 functionality of the *publish and subscribe* communications structure available from an
16 *agent*.

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

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

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

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

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

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

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

53 Note: The term “document” is used with different meanings in the MTCon-
54 nect Standard:

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

62 The following will be used throughout the MTConnect Standard to distinguish between
63 these different meanings for the term “document”:

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

- 69 • All references to electronic documents generated by an *agent* and sent to a client
 70 software application shall be referred to as a *response document*.

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

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

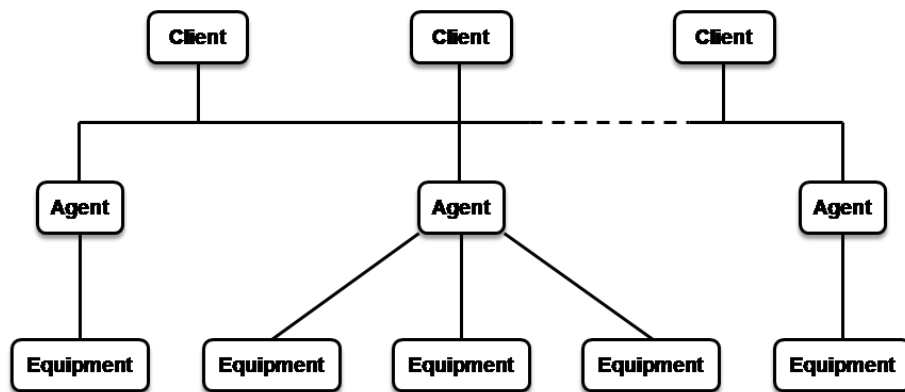


Figure 1: Basic MTConnect Implementation Structure

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

- 77 • **Equipment:** Any data source. In the MTConnect Standard, equipment is defined as
 78 any tangible property that is used to equip the operations of a manufacturing facil-
 79 ity. Examples of equipment are machine tools, ovens, sensor units, workstations,
 80 software applications, and bar feeders.
- 81 • **Agent:** Software that collects data published from one or more piece(s) of equip-
 82 ment, organizes that data in a structured manner, and responds to requests for data
 83 from client software systems by providing a structured response in the form of a
 84 *response document* that is constructed using the *semantic data models* defined in the
 85 Standard.

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

- 90 • **Client Software Application:** Software that requests data from *agents* and processes
 91 that data in support of manufacturing operations.

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

94 • the method used by a client software application to request information from an
95 *agent*.

96 • the response that an *agent* provides to a client software application.

97 • a *data dictionary* used to provide consistency in understanding the meaning of data
98 reported by a data source.

99 • the description of the *semantic data models* used to structure *response documents*
100 provided by an *agent* to a client software application.

101 These functions are the primary building blocks that define the base functional structure
102 of the MTConnect Standard.

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

110 • **Data Collection:** The most common use of data in manufacturing is the collection
111 of data associated with the production of products and the operation of equipment
112 that produces those products. The MTConnect Standard provides comprehensive
113 *semantic data models* that represent data collected from manufacturing operations.
114 These *semantic data models* are detailed in *MTConnect Standard: Part 2.0 - Device*
115 *Information Model* and *MTConnect Standard: Part 3.0 - Observation Information*
116 *Model* of the MTConnect Standard.

117 • **Inter-operations Between Pieces of Equipment:** The MTConnect Standard provides
118 an *interaction model* that structures the information required to allow multiple pieces
119 of equipment to coordinate actions required to implement manufacturing activities.
120 This *interaction model* is an implementation of a *request and response* messaging
121 structure. This *interaction model* is called `Interfaces` which is detailed in *MT-*
122 *Connect Standard: Part 5.0 - Interface Interaction Model* of the MTConnect Stan-
123 dard.

- 124 • Shared Data: Certain information used in a manufacturing operation is commonly
125 shared amongst multiple pieces of equipment and/or software applications. This
126 information is not typically “owned” by any one manufacturing resource. The MT-
127 Connect Standard represents this information through a series of *semantic data mod-*
128 *els* – each describing different types of information used in the manufacturing en-
129 vironment. Each type of information is called an *Asset*. *Assets* are detailed in *MT-*
130 *Connect Standard: Part 4.0 - Asset Information Model*, and its sub-Parts, of the
131 MTCConnect Standard.

132 **2 Purpose of This Document**

133 This document, *MTConnect Standard Part 1.0 - Fundamentals* of the MTConnect Stan-
134 dard, addresses two major topics relating to the MTConnect Standard. The first sections of
135 the document define the organization of the documents used to describe the MTConnect
136 Standard; including the terms and terminology used throughout the Standard. The balance
137 of the document defines the following:

- 138 • Operational concepts describing how an *agent* should organize and structure data
139 that has been collected from a data source.
- 140 • Definition and structure of the *response documents* supplied by an *agent*.
- 141 • The protocol used by a client software application to communicate with an *agent*.

142 3 Terminology and Conventions

143 This section provides a dictionary of terms, reserved language, and document conventions
144 used in the MTConnect Standard.

145 3.1 General Terms

146 *adapter*

147 optional piece of hardware or software that transforms information provided by a
148 piece of equipment into a form that can be received by an *agent*.

149 *agent*

150 software that collects data published from one or more piece(s) of equipment, or-
151 ganizes that data in a structured manner, and responds to requests for data from
152 client software systems by providing a structured response in the form of a *response*
153 *document* that is constructed using the *semantic data model* of a Standard.

154 *alarm limit*

155 limit used to trigger warning or alarm indicators.

156 *application*

157 software or a program that is specific to the solution of an application problem.
158 *Ref ISO/IEC 20944-1:2013*

159 *archetype*

160 *archetype* provides the requirements, constraints, and common properties for a type
161 of *Asset*.

162 *asset buffer*

163 *buffer* for *Assets*.

164 *attachment*

165 connection by which one thing is associated with another.

166 *buffer*

167 section of an *agent* that provides storage for information published from pieces of
168 equipment.

169 ***client***

170 *application* that sends *request* for information to an *agent*.

171 Note: Examples include software applications or a function that imple-
172 ments the *request* portion of an *interface interaction model*.

173 ***controlled vocabulary***

174 restricted set of values that may be published for an observation.

175 ***data dictionary***

176 listing of standardized terms and definitions used in *MTCConnect Information Model*.

177 ***data model***

178 organizes elements of data and standardizes how they relate to one another and to
179 the properties of real-world entities.

180 ***data set***

181 *key-value pairs* where each entry is uniquely identified by the *key*.

182 ***data source***

183 piece of equipment that can produce data that is published to an *agent*.

184 ***deprecated***

185 indication that specific content in an *MTCConnect Document* is currently usable but
186 is regarded as being obsolete or superseded.

187 ***deprecation warning***

188 indication that specific content in an *MTCConnect Document* may be changed to *dep-*
189 *recated* in a future release of the standard.

190 ***document***

191 piece of written, printed, or electronic matter that provides information or evidence
192 that serves as an official record.

193 ***electric current***

194 rate of flow of electric charge.

195 ***element***

196 constituent part or a basic unit of identifiable and definable data.

197 ***extensible***

198 ability for an implementer to extend *MTConnect Information Model* by adding con-
199 tent not currently addressed in the MTConnect Standard.

200 ***force***

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

202 ***heartbeat***

203 function that indicates to a *client* that the communications connection to an *agent* is
204 still viable during times when there is no new data available to report often referred
205 to as a “keep alive” message.

206 ***higher level***

207 nested element that is above a lower level element.

208 ***implementation***

209 specific instantiation of the MTConnect Standard.

210 ***information model***

211 rules, relationships, and terminology that are used to define how information is struc-
212 tured.

213 ***instance***

214 describes a set of *streaming data* in an *agent*. Each time an *agent* is restarted with
215 an empty *buffer*, data placed in the *buffer* represents a new *instance* of the *agent*.

216 ***interaction model***

217 model that defines how information is exchanged across an *interface* to enable in-
218 teractions between independent systems.

219 ***interface***

220 means by which communication is achieved between independent systems.

221 ***key***

222 unique identifier in a *key-value pair* association.

223 ***key-value pair***

224 association between an identifier referred to as the *key* and a value which taken
225 together create a *key-value pair*.

226 ***lower camel case***

227 first word is lowercase and the remaining words are capitalized and all spaces be-
228 tween words are removed.

229 ***lower level***

230 nested element that is below a higher level element.

231 ***lower limit***

232 lower conformance boundary for a variable.

233 ***lower warning***

234 lower boundary indicating increased concern and supervision may be required.

235 ***major***

236 identifier representing a consistent set of functionalities defined by the MTConnect
237 Standard.

238 ***maximum***

239 numeric upper constraint.

240 ***message***

241 communication in writing, in speech, or by signals.

242 ***metadata***

243 data that provides information about other data.

244 ***minimum***

245 numeric lower constraint.

246 ***minor***

247 identifier representing a specific set of functionalities defined by the MTConnect
248 Standard.

249 ***nominal***

250 ideal or desired value for a variable.

251 ***organize***

252 act of containing and owning one or more elements.

253 ***organizer***

254 entity that *organizes* one or more elements.

255 ***parameter***

256 variable that must be given a value during the execution of a program or a commu-
257 nications command.

258 ***part***

259 discrete item that has both defined and measurable physical characteristics including
260 mass, material, and features, and is created by applying one or more manufacturing
261 process steps to a workpiece

262 ***pascal case***

263 first letter of each word is capitalized and the remaining letters are in lowercase. All
264 space is removed between letters

265 ***persistence***

266 method for retaining or restoring information.

267 ***probe***

268 instrument commonly used for measuring the physical geometrical characteristics
269 of an object.

270 ***profile***

271 extends a reference metamodel (such as Unified Modeling Language (UML)) by
272 allowing to adapt or customize the metamodel with constructs that are specific to a
273 particular domain, platform, or a software development method.

274 ***requester***

275 entity that initiates a *request* for information in a communications exchange.

276 ***reset***

277 act of reverting back the accumulated value or statistic to their initial value.

278 Note: An *Observation* with a *data set* representation removes all *key-*
279 *value pairs*, setting the *data set* to an empty set.

280 ***responder***

281 entity that responds to a *request* for information in a communications exchange.

282 ***response document***

283 electronic *document* published by an *MTCConnect Agent* in response to a *probe re-*
284 *quest*, *current request*, *sample request* or *asset request*.

285 ***revision***

286 supplemental identifier representing only organizational or editorial changes to a
287 *minor* version document with no changes in the functionality described in that doc-
288 ument.

289 ***schema***

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

292 ***semantic data model***

293 methodology for defining the structure and meaning for data in a specific logical
294 way that can be interpreted by a software system.

295 ***sensing element***

296 mechanism that provides a signal or measured value.

297 ***sequence number***

298 primary key identifier used to manage and locate a specific piece of *streaming data*
299 in an *agent*.

300 ***specification limit***

301 limit defining a range of values designating acceptable performance for a variable.

302 ***spindle***

303 mechanism that provides rotational capabilities to a piece of equipment.

304 Note: Typically used for either work holding, materials or cutting tools.

305 ***standard***

306 *document* established by consensus that provides rules, guidelines, or characteristics
307 for activities or their results.. *Ref ISO/IEC Guide 2:2004*

308 ***stereotype***

309 defines how an existing UML metaclass may be extended as part of a *profile*.

310 ***subtype***

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

312 ***table***

313 two dimensional set of values given by a set of *key-value pairs table entries*.

- 314 ***table cell***
 315 subdivision of a *table entry* representing a singular value.
- 316 ***table entry***
 317 subdivision of a *table* containing a set of *key-value pairs* representing *table cells*.
- 318 ***top level***
 319 element that represents the most significant physical or logical functions of a piece
 320 of equipment.
- 321 ***type***
 322 classification or categorization of information.
- 323 ***upper limit***
 324 upper conformance boundary for a variable.
- 325 ***upper warning***
 326 upper boundary indicating increased concern and supervision may be required.
- 327 ***version***
 328 unique identifier of the administered item. *Ref ISO/IEC 11179-:2015*

329 **3.2 Information Model Terms**

- 330 ***Asset Information Model***
 331 *information model* that provides semantic models for *Assets*.
- 332 ***Device Information Model***
 333 *information model* that describes the physical and logical configuration for a piece
 334 of equipment and the data that may be reported by that equipment.
- 335 ***Error Information Model***
 336 *information model* that describes the *response document* returned by an *agent* when
 337 it encounters an error while interpreting a *request* for information from a *client* or
 338 when an *agent* experiences an error while publishing the *response* to a *request* for
 339 information.
- 340 ***MTCConnect Information Model***
 341 *information model* that defines the semantics of the MTCConnect Standard.

342 ***Observation Information Model***

343 *information model* that describes the *streaming data* reported by a piece of equip-
344 ment.

345 **3.3 Protocol Terms**

346 ***asset request***

347 *HTTP Request* to the *agent* regarding *Assets*.

348 ***current request***

349 *request* to an *agent* to produce an *MTCConnectStreams Response Document* contain-
350 ing the *Observation Information Model* for a snapshot of the latest observations at
351 the moment of the *request* or at a given *sequence number*.

352 ***data streaming***

353 method for an *agent* to provide a continuous stream of information in response to a
354 single *request* from a *client*.

355 ***MTCConnect Request***

356 *request* for information issued from a *client* to an *MTCConnect Agent*.

357 ***MTCConnect Response Document***

358 *response document* published by an *MTCConnect Agent*.

359 ***MTCConnectAssets Response Document***

360 *response document* published by an *MTCConnect Agent* in response to an *asset re-*
361 *quest*.

362 ***MTCConnectDevices Response Document***

363 *response document* published by an *MTCConnect Agent* in response to a *probe re-*
364 *quest*.

365 ***MTCConnectErrors Response Document***

366 *response document* published by an *MTCConnect Agent* whenever it encounters an
367 error while interpreting an *MTCConnect Request*.

368 ***MTCConnectStreams Response Document***

369 *response document* published by an *MTCConnect Agent* in response to a *current re-*
370 *quest* or a *sample request*.

371 ***probe request***

372 *request* to an *agent* to produce an *MTConnectDevices Response Document* contain-
373 ing the *Device Information Model*.

374 ***protocol***

375 set of rules that allow two or more entities to transmit information from one to the
376 other.

377 ***publish***

378 sending of messages in a *publish and subscribe* pattern.

379 ***publish and subscribe***

380 asynchronous communication method in which messages are exchanged between
381 applications without knowing the identity of the sender or recipient.

382 Note: In the MTConnect Standard, a communications messaging pattern
383 that may be used to publish *streaming data* from an *agent*.

384 ***request***

385 communications method where a *client* transmits a message to an *agent*. That mes-
386 sage instructs the *agent* to respond with specific information.

387 ***request and response***

388 communications pattern that supports the transfer of information between an *agent*
389 and a *client*.

390 ***response***

391 response *interface* which responds to a *request*.

392 ***sample request***

393 *request* to an *agent* to produce an *MTConnectStreams Response Document* contain-
394 ing the *Observation Information Model* for a set of timestamped observations made
395 by *Components*.

396 ***streaming data***

397 observations published by a piece of equipment defined by the equipment metadata.

398 ***subscribe***

399 receiving messages in a *publish and subscribe* pattern.

400 ***transport protocol***

401 set of capabilities that provide the rules and procedures used to transport information
402 between an *agent* and a client software application through a physical connection.

403 3.4 HTTP Terms

404 **HTTP Body**

405 data bytes transmitted in an HTTP transaction message immediately following the
406 headers. *Ref IETF:RFC-2616*

407 **HTTP Error Message**

408 response provided by an *agent* indicating that an *HTTP Request* is incorrectly for-
409 matted or identifies that the requested data is not available from the *agent*. *Ref IETF:RFC-*
410 *2616*

411 **HTTP Header**

412 header of either an *HTTP Request* from a *client* or an *HTTP Response* from an *agent*.
413 *Ref IETF:RFC-2616*

414 **HTTP Header Field**

415 components of the header section of request and response messages in an HTTP
416 transaction. *Ref IETF:RFC-2616*

417 **HTTP Message**

418 consist of requests from client to server and responses from server to client. *Ref IETF:RFC-*
419 *2616*

420 Note: In MTConnect Standard, it describes the information that is ex-
421 changed between an *agent* and a *client*.

422 **HTTP Messaging**

423 *interface* for information exchange functionality. *Ref IETF:RFC-2616*

424 **HTTP Method**

425 portion of a command in an *HTTP Request* that indicates the desired action to be
426 performed on the identified resource; often referred to as verbs. *Ref IETF:RFC-*
427 *2616*

428 **HTTP Query**

429 portion of a request for information that more precisely defines the specific informa-
430 tion to be published in response to the request. *Ref IETF:RFC-2616*

431 **HTTP Request**

432 request message from a client to a server includes, within the first line of that mes-
433 sage, the method to be applied to the resource, the identifier of the resource, and the
434 protocol version in use. *Ref IETF:RFC-2616*

435 Note: In MTConnect Standard, a request issued by a *client* to an *agent*
436 requesting information defined in the *HTTP Request Line*.

437 ***HTTP Request Line***

438 begins with a method token, followed by the Request-URI and the protocol version,
439 and ending with CRLF. A CRLF is allowed in the definition of TEXT only as part
440 of a header field continuation. *Ref IETF:RFC-2616*

441 Note: the first line of an *HTTP Request* describing a specific *response*
442 *document* to be published by an *agent*.

443 ***HTTP Request Method***

444 indicates the method to be performed on the resource identified by the Request-URI.
445 *Ref IETF:RFC-2616*

446 ***HTTP Request URI***

447 Uniform Resource Identifier that identifies the resource upon which to apply the
448 request. *Ref IETF:RFC-2616*

449 ***HTTP Response***

450 after receiving and interpreting a request message, a server responds with an HTTP
451 response message. *Ref IETF:RFC-2616*

452 Note: In MTConnect Standard, the information published from an *agent*
453 in reply to an *HTTP Request*.

454 ***HTTP Server***

455 server that accepts *HTTP Request* from *client* and publishes *HTTP Response* as a
456 reply to those *HTTP Request*. *Ref IETF:RFC-2616*

457 ***HTTP Status Code***

458 3-digit integer result code of the attempt to understand and satisfy the request.
459 *Ref IETF:RFC-2616*

460 ***HTTP Version***

461 version of the HTTP protocol. *Ref IETF:RFC-2616*

462 3.5 XML Terms

463 ***abstract element***

464 element that defines a set of common characteristics that are shared by a group of
465 elements. An abstract entity cannot appear in a document. In a specific implemen-
466 tation, an abstract entity is replaced by a derived element that is itself not an abstract
467 entity. The characteristics for the derived element are inherited from the abstract
468 entity.

469 ***attribute***

470 additional information or property for an *element*.

471 ***child element***

472 *element* of a data modeling structure that illustrates the relationship between itself
473 and the higher-level *parent element* within which it is contained.

474 ***document body***

475 portion of the content of an *MTCConnect Response Document* that is defined by the
476 relative *MTCConnect Information Model*. The *document body* contains the *structural*
477 *elements* and *Observations* or *DataItems* reported in a *response document*.

478 ***document header***

479 portion of the content of an *MTCConnect Response Document* that provides infor-
480 mation from an *agent* defining version information, storage capacity, protocol, and
481 other information associated with the management of the data stored in or retrieved
482 from the *agent*.

483 ***element name***

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

486 ***namespace***

487 organizes information into logical groups.

488 ***parent element***

489 *element* of a data modeling structure that illustrates the relationship between itself
490 and the lower-level *child element*.

491 ***root element***

492 first *structural element* provided in a *response document* encoded using XML.

493 ***structural element***

494 *element* that organizes information that represents the physical and logical parts and
 495 sub-parts of a piece of equipment.

496 ***XML Document***

497 structured text file encoded using Extensible Markup Language (XML).

498 ***XML Schema***

499 *schema* defining a specific document encoded in XML.

500 **3.6 MTConnect Terms**501 ***Asset***

502 asset that is used by the manufacturing process to perform tasks.

503 Note 1 to entry: An *Asset* relies upon an *Device* to provide observations
 504 and information about itself and the *Device* revises the information to
 505 reflect changes to the *Asset* during their interaction. Examples of *Assets*
 506 are cutting tools, Part Information, Manufacturing Processes, Fixtures,
 507 and Files.

508 Note 2 to entry: A singular `assetId`, *Asset* uniquely identifies an
 509 *Asset* throughout its lifecycle and is used to track and relate the *Asset* to
 510 other *Devices* and entities.

511 Note 3 to entry: *Assets* are temporally associated with a device and can
 512 be removed from the device without damage or alteration to its primary
 513 functions.

514 ***Component***

515 engineered system part of a *Device* composed of zero or more *Components*

516 ***Composition***

517 *Component* belonging to a *Component* and not composed of any *Components*.

518 ***Configuration***

519 configuration for a *Component*

520 ***DataItem***

521 observable observed by a *Component* that may make *Observations*

522 ***Device***

523 *Component* not belonging to any *Component* that may have assets

524 ***MTCConnect Agent***

525 *agent* for the *MTCConnect Information Model*.

526 ***MTCConnect Document***

527 *document* that represents a Part(s) of the MTCConnect Standard.

528 ***MTCConnect Event***

529 observation of either a state or discrete value of the *Component*.

530 ***MTCConnect Interface***

531 *interaction model* for interoperability between pieces of equipment.

532 ***Observation***

533 observation that provides telemetry data for a *DataItem*.

534 **3.7 Acronyms**

535 ***2D***

536 two-dimensional

537 ***3D***

538 three-dimensional

539 ***AI***

540 artificial intelligence

541 ***ALM***

542 application lifecycle management

543 ***AMT***

544 The Association for Manufacturing Technology

545 ***ANSI***

546 American National Standards Institute

- 547 **AP**
- 548 Application Protocol
- 549 **API**
- 550 application programming interface
- 551 **ASME**
- 552 American Society of Mechanical Engineers
- 553 **ASTM**
- 554 American Society for Testing and Materials
- 555 **AWS**
- 556 American Welding Society
- 557 **BDD**
- 558 block definition diagram
- 559 **BOM**
- 560 bill of materials
- 561 **BST**
- 562 Board on Standardization and Testing
- 563 **C&R**
- 564 cause and remedy
- 565 **CA**
- 566 certificate authority
- 567 **CAD**
- 568 computer-aided design
- 569 **CAE**
- 570 computer-aided engineering
- 571 **CAI**
- 572 computer-aided inspection
- 573 **CAM**
- 574 computer-aided manufacturing

575	CAx
576	computer-aided technologies
577	CDATA
578	Character Data
579	CFD
580	computational fluid dynamics
581	CM
582	configuration management
583	CMS
584	coordinate-measurement system
585	CNC
586	Computer Numerical Controller
587	CNRI
588	Corporation for National Research Initiatives
589	CPM
590	Core Product Model
591	CPM2
592	Revised Core Product Model
593	CPSC
594	Consumer Product Safety Commission
595	cUAV
596	configurable unmanned aerial vehicle
597	DARPA
598	Defense Advanced Research Projects Agency
599	DER
600	designated-engineering representative
601	DFM
602	design for manufacturing

603	<i>DLA</i>
604	Defense Logistics Agency
605	<i>DMC</i>
606	digital manufacturing certificate
607	<i>DMSC</i>
608	Dimensional Metrology Standards Consortium
609	<i>DNS</i>
610	Domain Name System
611	<i>DoD</i>
612	U.S. Department of Defense
613	<i>DOI</i>
614	Distributed Object Identifier
615	<i>DRM</i>
616	digital rights management
617	<i>ECR</i>
618	engineering change request
619	<i>ERP</i>
620	enterprise resource planning
621	<i>FAA</i>
622	Federal Aviation Administration
623	<i>FAIR</i>
624	first article inspection reporting
625	<i>FDA</i>
626	Food and Drug Administration
627	<i>FEA</i>
628	finite-element analysis
629	<i>GD&T</i>
630	geometric dimensions and tolerances

631	<i>GID</i>
632	global identifier
633	<i>HMI</i>
634	Human Machine Interface
635	<i>HTML</i>
636	Hypertext Markup Language
637	<i>HTTP</i>
638	Hypertext Transfer Protocol
639	<i>HTTPS</i>
640	Hypertext Transfer Protocol over Secure Sockets Layer
641	<i>I/O</i>
642	in-out
643	<i>ID</i>
644	identifier
645	<i>IEEE</i>
646	Institute of Electrical and Electronics Engineers
647	<i>IIoT</i>
648	industrial internet of things
649	<i>INCOSE</i>
650	International Council on Systems Engineering
651	<i>IP</i>
652	intellectual property
653	<i>ISO</i>
654	International Standards Organization
655	<i>ISS</i>
656	International Space Station
657	<i>ISV</i>
658	Independent Software Vendor

- 659 ***IT***
- 660 information technology
- 661 ***ITU-T***
- 662 Telecommunication Standardization Sector of the International Telecommunication
- 663 Union
- 664 ***JSON***
- 665 JavaScript Object Notation
- 666 ***JT***
- 667 Jupiter Tessellation
- 668 ***LHS***
- 669 Lifecycle Handler System
- 670 ***LIFT***
- 671 Lifecycle Information Framework and Technology
- 672 ***LOI***
- 673 Lifecycle Object Identifier
- 674 ***MAC***
- 675 media access control
- 676 ***MADE***
- 677 Manufacturing Automation and Design Engineering
- 678 ***MBD***
- 679 model-based definition
- 680 ***MBE***
- 681 Model-Based Enterprise
- 682 ***MBI***
- 683 model-based inspection
- 684 ***MBM***
- 685 model-based manufacturing

- 686 ***MBSD***
- 687 model-based standards development
- 688 ***MBSE***
- 689 model-based systems engineering
- 690 ***MEDALS***
- 691 Military Engineering Data Asset Locator System
- 692 ***MES***
- 693 manufacturing execution system
- 694 ***MOI***
- 695 manufacturing object identifier
- 696 ***MOM***
- 697 Message Orienged Middleware
- 698 ***MQTT***
- 699 Message Queuing Telemetry Transport
- 700 ***MTC***
- 701 Manufacturing Technology Centre
- 702 ***NASA***
- 703 National Aeronautics and Space Administration
- 704 ***NC***
- 705 numerical control
- 706 ***NIST***
- 707 National Institute of Standards and Technology
- 708 ***NMTOKEN***
- 709 Name Token
- 710 ***NNMI***
- 711 National Network of Manufacturing Innovation
- 712 ***NSF***
- 713 National Science Foundation

- 714 ***NTSC***
- 715 National Transportation Safety Board
- 716 ***OASIS***
- 717 Organization for the Advancement of Structured Information Standards
- 718 ***ODI***
- 719 Open Data Institute
- 720 ***OEM***
- 721 original equipment manufacturer
- 722 ***OOI***
- 723 Ocean Observatories Initiative
- 724 ***OPC***
- 725 OLE for Process Control
- 726 ***OSLC***
- 727 Open Services for Lifecycle Collaboration
- 728 ***OSTP***
- 729 Office of Science and Technology Policy
- 730 ***OT***
- 731 operational technology
- 732 ***OWL***
- 733 Ontology Web Language
- 734 ***PDF***
- 735 Portable Document Format
- 736 ***PDM***
- 737 product-data management
- 738 ***PDQ***
- 739 product-data quality
- 740 ***PHM***
- 741 prognosis and health monitoring

742	<i>PI</i>	
743		principal investigator
744	<i>PLC</i>	
745		Programmable Logic Controller
746	<i>PLCS</i>	
747		Product Life Cycle Support
748	<i>PLM</i>	
749		product lifecycle management
750	<i>PLOT</i>	
751		product lifecycle of trust
752	<i>PMI</i>	
753		product and manufacturing information
754	<i>PMS</i>	
755		Production Management System
756	<i>PRC</i>	
757		Product Representation Compact
758	<i>PSI</i>	
759		Physical Science Informatics
760	<i>PTAB</i>	
761		Primary Trustworthy Digital Repository Authorization Body Ltd.
762	<i>QIF</i>	
763		Quality Information Framework
764	<i>QMS</i>	
765		quality management system
766	<i>QName</i>	
767		Qualified Name
768	<i>RDF</i>	
769		Resource Description Framework

770	REST
771	Representational State Transfer
772	RII
773	receiving and incoming inspection
774	S/MIME
775	Secure/Multipurpose Internet Mail Extensions
776	SaaS
777	software-as-a-service
778	SAML
779	Security Assertion Markup Language
780	SC
781	Standards Committee
782	SCADA
783	Supervisory Control And Data Acquisition
784	SDO
785	Standards Development Organization
786	SFTP
787	Secure File Transfer Protocol
788	SKOS
789	Simple Knowledge Organization System
790	SLH
791	system lifecycle handler
792	SLR
793	systematic literature review
794	SME
795	small-to-medium enterprise
796	SMOPAC
797	Smart Manufacturing Operations Planning and Control

798	<i>SMS Test Bed</i>
799	Smart Manufacturing Systems Test Bed
800	<i>SOA</i>
801	service-oriented architecture
802	<i>SPMM</i>
803	semantic-based product metamodel
804	<i>SSL</i>
805	Secure Sockets Layer
806	<i>STEP</i>
807	Standard for the Exchange of Product Model Data
808	<i>STEP AP242</i>
809	Standard for the Exchange of Product Model Data Application Protocol 242
810	<i>STL</i>
811	Stereolithography
812	<i>SysML</i>
813	Systems Modeling Language
814	<i>TCP/IP</i>
815	Transmission Control Protocol/Internet Protocol
816	<i>TDP</i>
817	technical data package
818	<i>TLS</i>
819	Transport Layer Security
820	<i>TSM</i>
821	Total System Model
822	<i>UA</i>
823	Unified Architecture
824	<i>UAL</i>
825	Unified Architecture Language

826	<i>UML</i>
827	Unified Modeling Language
828	<i>URI</i>
829	Uniform Resource Identifier
830	<i>URL</i>
831	Uniform Resource Locator
832	<i>URN</i>
833	Uniform Resource Name
834	<i>UTC</i>
835	Coordinated Universal Time
836	<i>UUID</i>
837	Universally Unique Identifier
838	<i>V&V</i>
839	verification and validation
840	<i>W3C</i>
841	World Wide Web Consortium
842	<i>WSN</i>
843	Wirth Syntax Notation
844	<i>WWW</i>
845	World Wide Web
846	<i>X.509-PKI</i>
847	Public Key Infrastructure
848	<i>X.509-PMI</i>
849	Privilege Management Infrastructure
850	<i>XML</i>
851	Extensible Markup Language
852	<i>XPath</i>
853	XML Path Language
854	<i>XSD</i>
855	XML Schema Definitions

856 **3.8 MTConnect References**

857 [MTConnect Part 1.0] *MTConnect Standard Part 1.0 - Fundamentals*. Version 2.0.

858 [MTConnect Part 2.0] *MTConnect Standard: Part 2.0 - Device Information Model*. Ver-
859 sion 2.0.

860 [MTConnect Part 3.0] *MTConnect Standard: Part 3.0 - Observation Information Model*.
861 Version 2.0.

862 [MTConnect Part 4.0] *MTConnect Standard: Part 4.0 - Asset Information Model*. Ver-
863 sion 2.0.

864 [MTConnect Part 5.0] *MTConnect Standard: Part 5.0 - Interface Interaction Model*. Ver-
865 sion 2.0.

866

867 4 Fundamentals

868 The MTConnect Standard defines the normative information model and protocol for re-
869 trieving information from manufacturing equipment. This document specifies the *agent*
870 behavior and protocol.

871 4.1 Agent

872 The MTConnect Standard specifies the minimum functionality of the *agent*. The function-
873 ality is as follows:

- 874 • Provides store and forward messaging middleware service.
- 875 • Provides key-value information storage and asset retrieval service.
- 876 • Implements the REST API for the MTConnect Standard (See *Section 5.1 - REST*
877 *Protocol*).
 - 878 – *Device* metadata.
 - 879 – observations collected by the agent.
 - 880 – assets collected by the agent.

881 There are three types of information stored by an *agent* that **MAY** be published in a *re-*
882 *sponse document*. These are as follows:

- 883 • equipment metadata specified in *MTConnect Standard: Part 2.0 - Device Informa-*
884 *tion Model*.
- 885 • *streaming data* provides the observations specified in *MTConnect Standard: Part*
886 *3.0 - Observation Information Model*.
- 887 • *Assets* specified in *MTConnect Standard: Part 4.0 - Asset Information Model*.

888 4.1.1 Agent Instance ID

889 The *agent* **MUST** set the `instanceId` to a unique value whenever the *sequence number*
890 in the agent is initialized to 1. (see *Section 4.1.3.1 - Sequence Numbers* and *Section 4.1.3.7*
891 *- Persistence and Recovery* below).

892 4.1.2 Storage of Equipment Metadata

893 An *agent* **MUST** be capable of publishing equipment metadata for the *agent* as specified
894 in *MTConnect Standard: Part 2.0 - Device Information Model*.

895 4.1.3 Storage of Streaming Data

896 The *agent* **MAY** implement a *buffer* with a fixed number of observations. If the `buffer-`
897 `Size` is fixed, the *agent* **MUST** store observations using a first-in-first-out pattern. The
898 *agent* will remove the oldest observation when the *buffer* is full and a new observation
899 arrives.

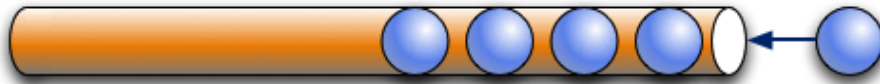


Figure 2: Data Storage in Buffer

900 In Figure 3, the maximum number of observations that can be stored in the *buffer* of the
901 *agent* is 8. The `bufferSize` in the header reports the maximum number of observations.
902 This example illustrates that when the *buffer* fills up, the oldest piece of data falls out the
903 other end.

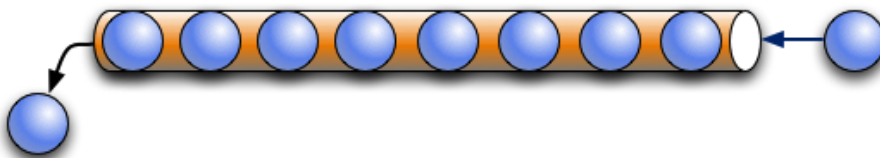


Figure 3: First In First Out Buffer Management

904 Note: As an implementation suggestion, the *buffer* should be sized large
905 enough to provide a continuous stream of observations. The implementer
906 should also consider the impact of a temporary loss of communications when
907 determining the size for the *buffer*. A larger *buffer* will allow more time to
908 reconnect to an *agent* without losing data.

909 4.1.3.1 Sequence Numbers

910 In an *agent*, each occurrence of an observation in the *buffer* will be assigned a mono-
 911 tonically increasing unsigned 64-bit integer (*sequence number*) when it arrives. The first
 912 *sequence number* **MUST** be 1.

913 The *sequence number* for each observation **MUST** be unique for an instance of an *agent*
 914 identified by an `instanceId`.

915 Table 1 illustrates the changing of the `instanceId` when an *agent* resets the *sequence*
 916 *number* to 1.

<code>instanceId</code>	<code>sequence</code>
234556	234
	235
	236
	237
	238
Agent Stops and Restarts	
234557	1
	2
	3
	4
	5

Table 1: `instanceId` and `sequence`

917 Figure 4 shows two additional pieces of information defined for an *agent*:

918 • `firstSequence` – the oldest observation in the *buffer*. The *agent* removes this
 919 observation when it receives the next observation

920 • `lastSequence` – the newest observation in the *buffer*

921 `firstSequence` and `lastSequence` provide the range of values for the REST API
 922 requests.

923 The *agent* **MUST** begin evaluating observations with *sample request*'s `from` parameter.
 924 Also, the *agent* **MUST** include a maximum number of observations given by the `count`
 925 parameter in the *response document*.

926 In Figure 5, the request specifies the observations start at *sequence number* 15 (`from`)
 927 and includes a total of three items (`count`).

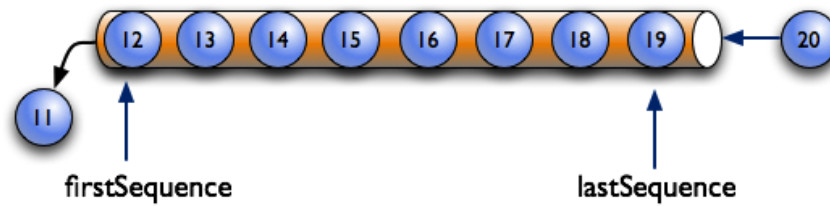


Figure 4: Identifying the range of data with firstSequence and lastSequence

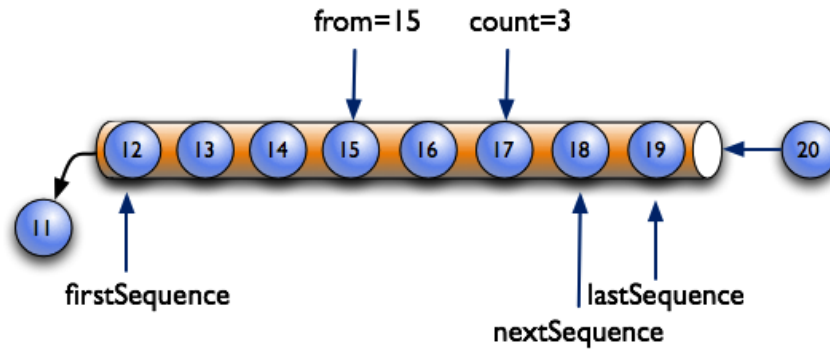


Figure 5: Identifying the range of data with from and count

928 nextSequence header property has the *sequence number* of the next observation in the
 929 *buffer* for subsequent *sample requests* providing a contiguous set of observations. In the
 930 example in Figure 5, the next *sequence number* (nextSequence) will be 18.

931 As shown in Figure 6, the combination of from and count defined by the *request* indi-
 932 cates a *sequence number* for data that is beyond that which is currently in the *buffer*. In
 933 this case, nextSequence is set to a value of lastSequence + 1.

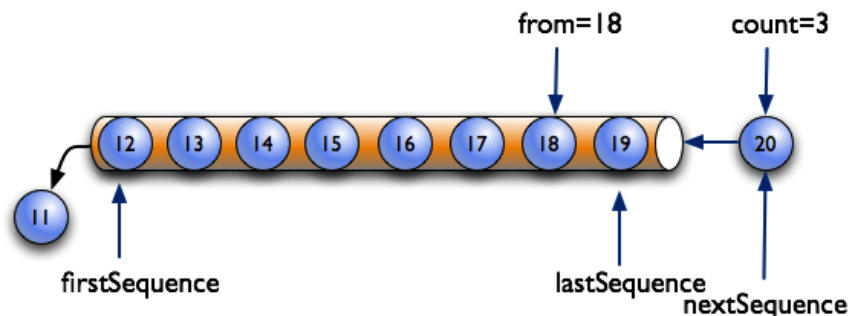


Figure 6: Identifying the range of data with nextSequence and lastSequence

934 **4.1.3.2 Observation Buffer**

935 An observation has four pieces of information as follows:

- 936 1. *sequence number* associated with each observation - *sequence*.
- 937 2. The *timestamp* the observation was made. .
- 938 3. A reference to the *dataItemid* from the *MTConnect Standard: Part 2.0 - Device*
939 *Information Model*.
- 940 4. The value of the observation.

941 Table 2 is an example demonstrating the concept of how data may be stored in an *agent*:

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

Table 2: Data Storage Concept

942 4.1.3.3 Timestamp

943 observations **MUST** have a *timestamp* giving the most accurate time that the observa-
944 tion occurred.

945 The timezone of the *timestamp* **MUST** be UTC (Coordinated Universal Time) and
946 represented using ISO 8601 format: e.g., “2010-04-01T21:22:43Z”.

947 Applications **SHOULD** use the observation’s *timestamp* for ordering as opposed to
948 *sequence number*.

949 All observations occurring at the same time **MUST** have the same *timestamp*.

950 4.1.3.4 Recording Occurrences of Streaming Data

951 The *agent* **MUST** only place observations in the *buffer* if the data has changed from the
952 previous observation for the same `DataItem`.

953 The *agent* **MUST** place every observation in the *buffer*, without checking for changes, in
954 the following cases:

- 955 • The `discrete` attribute is `true` for the `DataItem`.
- 956 • The `representation` is `DISCRETE`.
- 957 • The `representation` is `TIME_SERIES`.

958 4.1.3.5 Maintaining Last Value for Data Entities

959 An *agent* **MUST** retain the most recent observation associated with each `DataItem`, even
960 if the observation is no longer in the *buffer*. This function supports the *current request*
961 functionality.

962 4.1.3.6 Unavailability of Data

963 An observation with the value of `UNAVAILABLE` indicates the value is indeterminate.

964 The *agent* **MUST** initialize every `DataItem`, unless it has a constant value (see below),
965 with an observation with the value of `UNAVAILABLE`. Additionally, whenever the data
966 source is unreachable, every `DataItem` associated with the data source must have an
967 observation with the value of `UNAVAILABLE` and `timestamp` when the connection was
968 lost.

969 An `DataItem` that is constrained to a constant value, as defined in *MTConnect Standard:*
970 *Part 2.0 - Device Information Model*, **MUST** only have an observation with the constant
971 value and **MUST NOT** be set to `UNAVAILABLE`.

972 4.1.3.7 Persistence and Recovery

973 The *agent* **MAY** have a fixed size *buffer* and the *buffer* **MAY** be ephemeral.

974 If the *buffer* is recoverable, the *agent* **MUST NOT** change the `instanceId` and **MUST**
975 **NOT** set the *sequence number* to 1. The *sequence number* **MUST** be one greater than the
976 maximum value of the recovered observations. $max(sequence) + 1$

977 **4.1.4 Storage of MTConnect Assets**

978 An agent **MAY** only retain a limited number of Assets in the *asset buffer*. The Assets
 979 are stored in first-in-first-out method where the oldest Asset is removed when the *asset*
 980 *buffer* is full and a new Asset arrives.

981 Figure 7 illustrates the oldest Asset being removed from the *asset buffer* when a new
 982 Asset is added and the *asset buffer* is full:

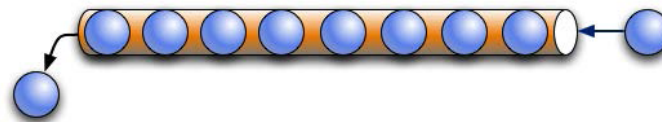


Figure 7: First In First Out Asset Buffer Management

983 Assets are indexed by *assetId*. In the case of Assets, Figure 8 demonstrates the
 984 relationship between the key (*assetId*) and the stored Asset:

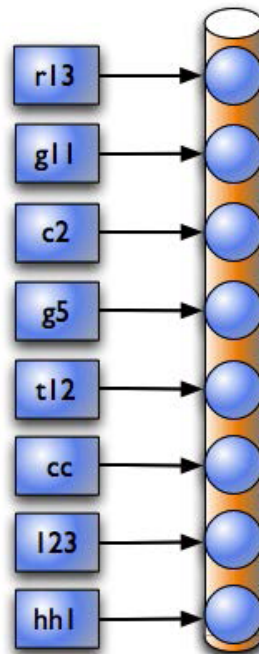


Figure 8: Relationship between *assetId* and stored Asset documents

985 Note: The key (*assetId*) is independent of the order of the Asset stored
 986 in the *asset buffer*.

987 When the *agent* receives a new *Asset*, one of the following rules **MUST** apply:

- 988 • If the *Asset* is not in the *asset buffer*, the *agent* **MUST** add the new *Asset* to the
989 front of the *asset buffer*. If the *asset buffer* is full, the oldest *Asset* will be removed
990 from the *asset buffer*.
- 991 • If the *Asset* is already in the *asset buffer*, the *agent* **MUST** replace the existing
992 *Asset* and move the *Asset* to the front of the *asset buffer*.

993 The number of *Asset* that may be stored in an *agent* is defined by the value for *as-*
994 *setBufferSize*. An *assetBufferSize* of 4,294,967,296 or 2^{32} **MUST** indicate
995 unlimited storage.

996 The *asset buffer* **MAY** be ephemeral and the *Asset* entities will be lost if the *agent* clears
997 the *asset buffer*. They must be recovered from the data source.

998 *MTConnect Standard: Part 4.0 - Asset Information Model* provides additional information
999 on asset management.

1000 4.2 Response Documents

1001 *response documents* are electronic documents generated by an *agent* in response to a *re-*
1002 *quest* for data.

1003 The *response documents* defined in the *MTConnect Standard* are:

- 1004 • *MTConnectDevices Response Document*: Describes the composition and config-
1005 uration of the *Device* and the data that can be observed. See *Section 5.2 - MT-*
1006 *ConnectDevices Response Document* and *MTConnect Standard: Part 2.0 - Device*
1007 *Information Model* for details on this information model.
- 1008 • *MTConnectStreams Response Document*: *Observations* made at a point in time
1009 about related *DataItems*. See *Section 5.3 - MTConnectStreams Response Document*
1010 and *MTConnect Standard: Part 3.0 - Observation Information Model* for details on
1011 this information model.
- 1012 • *MTConnectAssets Response Document*: *Assets* related to *Devices*. See *Section 5.4 -*
1013 *MTConnectAssets Response Document* and *MTConnect Standard: Part 4.0 - Asset*
1014 *Information Model* for details on this information model.

- 1015 • *MTConnectErrors Response Document*: Information in response to a failed request.
1016 See *Section 6.1 - MTConnectErrors Response Document* for details on this informa-
1017 tion model.

1018 4.3 Request/Response Information Exchange

1019 The transfer of information between an *agent* and a client software application is based on
1020 a *request and response* REST protocol. A client application requests specific information
1021 from an *agent* and an *agent* responds with a *response document*.

1022 There are four types of *MTConnect Requests*. These *requests* are as follows:

- 1023 • *probe request*: Requests information about one more more *Devices* as an MTCon-
1024 nectDevices block.
- 1025 • *current request*: Requests the most recent, or snapshot at a *sequence number*, obser-
1026 vations as an MTConnectStreams block.
- 1027 • *sample request*: Requests a series of observations as an MTConnectStreams
1028 block.
- 1029 • *asset request*: Requests a set of assets as an MTConnectAssets block.

1030 If an *agent* is unable to respond to the request for information or the request includes
1031 invalid information, the *agent* will publish an *MTConnectErrors Response Document*. See
1032 *MTConnectErrors*.

1033 See *Section 5.1 - REST Protocol* for the details on the normative requirements of the agent.

1034 5 MTConnect Protocol

1035 The *agent* **MUST** support the *Section 5.1 - REST Protocol* and produce XML representa-
1036 tions of the information models.

1037 All other protocols and representations are optional.

1038 5.1 REST Protocol

1039 An *agent* **MUST** provide a REST API application programming interface (API) support-
1040 ing HTTP version 1.0 or greater. This interface **MUST** support HTTP (RFC7230) and use
1041 URIs (RFC3986) to identify specific information requested from an *agent*.

1042 The REST API adheres to the architectural principles of a stateless service to retrieve infor-
1043 mation associated with pieces of equipment. Additionally, the API is read-only and does
1044 not produce any side effects on the *agent* or the equipment. In REST state management,
1045 the client is responsible for recovery in case of an error or loss of connection.

1046 5.1.1 HTTP Request

1047 An *agent* **MUST** support the HTTP GET verb, all other verbs are optional. See IETF RFC
1048 7230 for a complete description of the HTTP request structure.

1049 The HTTP uses Uniform Resource Identifiers (URI) as outlined in IETF RFC 3986 as the
1050 *request-target*. IETF RFC 7230 specifies the http URI scheme for the *request-target* as
1051 follows:

- 1052 1. *protocol*: The protocol used for the request. Must be `http` or `https`.
- 1053 2. *authority*: The network domain or address of the agent with an optional port.
- 1054 3. *path*: A Hierarchical Identifier following a slash (/) and before the optional question-
1055 mark (?). The *path* separates segments by a slash (/).
- 1056 4. *query*: The portion of the HTTP request following the question-mark (?). The
1057 query portion of the HTTP request is composed of key-value pairs, = separated by
1058 an ampersand (&).

1059 **5.1.1.1 path Portion of an HTTP Request**

1060 The path portion of the *request-target* has the following segments:

- 1061 • `device-name` or `uuid`: optional name or uuid of the Device
- 1062 • `request`: request, must be one of the following: (also see *Section 5.1.4.3 - Operations for Agent*)
1063
 - 1064 – `probe`
 - 1065 – `current`
 - 1066 – `sample`
 - 1067 – `asset` or `assets`
 - 1068 * `asset` request has additional optional segment `<asset ids>`

1069 If name or uuid segment are not specified in the *HTTP Request*, an *agent* **MUST** return
1070 information for all pieces of equipment. The following sections will

1071 Examples:

- 1072 • `http://localhost:5000/my_device/probe`
1073 The request only provides information about `my_device`.
- 1074 • `http://localhost:5000/probe`
1075 The request provides information for all devices.

1076 The following section specifies the details for each request.

1077 **5.1.2 MTConnect REST API**

1078 An *agent* **MUST** support *probe requests*, *current requests*, *sample requests*, and *asset*
1079 *requests*.

1080 See the operations of the *Agent* for details regarding the *requests*.

1081 5.1.3 HTTP Errors

1082 When an *agent* receives an *HTTP Request* that is incorrectly formatted or is not supported
 1083 by the *agent*, the *agent* **MUST** publish an *HTTP Error Message* which includes a specific
 1084 status code from the tables above indicating that the *request* could not be handled by the
 1085 *agent*.

1086 Also, if the *agent* experiences an internal error and is unable to provide the requested
 1087 *response document*, it **MUST** publish an *HTTP Error Message* that includes a specific
 1088 status code from the table above.

1089 When an *agent* encounters an error in interpreting or responding to an *HTTP Request*,
 1090 the *agent* **MUST** also publish an *MConnectErrors Response Document* that provides
 1091 additional details about the error. See *Section 6 - Error Information Model* for details on
 1092 the *MConnectErrors Response Document*.

1093 5.1.3.1 Streaming Data

1094 *HTTP data streaming* is a method for an *agent* to provide a continuous stream of observa-
 1095 tions in response to a single *request* using a *publish and subscribe* communication pattern.

1096 When an *HTTP Request* includes an `interval` parameter, an *agent* **MUST** provide data
 1097 with a minimum delay in milliseconds between the end of one data transmission and the
 1098 beginning of the next. A value of zero (0) for the `interval` parameter indicates that
 1099 the *agent* should deliver data at the highest rate possible and is only relevant for *sample*
 1100 *requests* .

1101 The format of the response **MUST** use an `x-multipart-replace` encoded message
 1102 with each section separated by MIME boundaries. Each section **MUST** contain an entire
 1103 *MConnectStreams Response Document*.

1104 When streaming for a *current request*, the *agent* produces an *MConnectStreams Response*
 1105 *Document* with the most current observations every `interval` milliseconds.

1106 When streaming for a *sample request*, if there are no available observations after the `in-`
 1107 `terval` time elapsed, the *agent* **MUST** wait for either the `heartbeat` time to elapse or
 1108 an observation arrives. If the `heartbeat` time elapses and no observations arrive, then
 1109 an empty *MConnectStreams Response Document* **MUST** be sent.

1110 Note: For more information on MIME, see IETF RFC 1521 and RFC 822.

1111 An example of the format for an *HTTP Request* that includes an `interval` parameter is:

Example 1: Example for HTTP Request with interval parameter

1112 1 http://localhost:5000/sample?interval=1000

1113 HTTP Response Header:

Example 2: HTTP Response header

1114 1 HTTP/1.1 200 OK
 1115 2 Connection: close
 1116 3 Date: Sat, 13 Mar 2010 08:33:37 UTC
 1117 4 Status: 200 OK
 1118 5 Content-Disposition: inline
 1119 6 X-Runtime: 144ms
 1120 7 Content-Type: multipart/x-mixed-replace;boundary=
 1121 8 a8e12eced4fb871ac096a99bf9728425
 1122 9 Transfer-Encoding: chunked

1123 Lines 1-9 in *Example 2* represent a standard header for a MIME `multipart/x-mixed-`
 1124 `replace` message. The boundary is a separator for each section of the stream. Lines 7-8
 1125 indicate this is a multipart MIME message and the boundary between sections.

1126 With streaming protocols, the `Content-length` **MUST** be omitted and `Transfer-`
 1127 `Encoding` **MUST** be set to `chunked` (line 9). See IETF RFC 7230 for a full description
 1128 of the HTTP protocol and chunked encoding.

Example 3: HTTP Response header 2

1129 10 --a8e12eced4fb871ac096a99bf9728425
 1130 11 Content-type: text/xml
 1131 12 Content-length: 887
 1132 13
 1133 14 <?xml version="1.0" encoding="UTF-8"?>
 1134 15 <MTConnectStreams ...>...

1135 Each section of the document begins with a boundary preceded by two hyphens (-). The
 1136 `Content-type` and `Content-length` header fields **MUST** be provided for each
 1137 section and **MUST** be followed by `<CR><LF><CR><LF>` (ASCII code for `<CR>` is 13
 1138 and `<LF>` 10) before the XML document. The header and the `<CR><LF><CR><LF>`
 1139 **MUST NOT** be included in the computation of the content length.

1140 An *agent* **MUST** continue to stream results until the client closes the connection. The
 1141 *agent* **MUST NOT** stop streaming for any reason other than the following:

- 1142 • *agent* process stops
- 1143 • The client application stops receiving data

1144 5.1.3.1.1 Heartbeat

1145 When *streaming data* is requested from a *sample request*, an *agent* **MUST** support a *heart-*
 1146 *beat* to indicate to a client application that the HTTP connection is still viable during
 1147 times when there is no new data available to be published. The *heartbeat* is indicated by
 1148 an *agent* by sending an *MTConnect response document* with an empty *Streams* entity
 1149 (See *MTConnect Standard: Part 3.0 - Observation Information Model* for more details on
 1150 *Streams*) to the client software application.

1151 The *heartbeat* **MUST** occur on a periodic basis given by the optional *heartbeat* query
 1152 parameter and **MUST** default to 10 seconds. An *agent* **MUST** maintain a separate *heart-*
 1153 *beat* for each client application for which the *agent* is responding to a *data streaming*
 1154 *request*.

1155 An *agent* **MUST** begin calculating the interval for the time-period of the *heartbeat* for
 1156 each client application immediately after a *response document* is published to that specific
 1157 client application.

1158 The *heartbeat* remains in effect for each client software application until the *data stream-*
 1159 *ing request* is terminated by either the *agent* or the client application.

1160 5.1.3.2 References

1161 A *Component* **MAY** include a set of *Reference* entities of the following types that
 1162 **MAY** alter the content of the *MTConnectStreams Response Documents* published in re-
 1163 sponse to a *current request* or a *sample request* as specified:

- 1164 • A *Component* reference (*ComponentRef*) modifies the set of *Observations*, lim-
 1165 ited by a path query parameter of a *current request* or *sample request*, to include
 1166 the *Observations* associated with the entity whose value for its *id* attribute matches
 1167 the value provided for the *idRef* attribute of the *ComponentRef* element. Ad-
 1168 ditionally, *Observations* defined for any *lower level* entity(s) associated with the
 1169 identified entities **MUST** also be returned. The result is equivalent to appending
 1170 `//[@id=<"idRef">]` to the path query parameters of the *current request* or *sam-*
 1171 *ple request*. See *Section 4.1 - Agent* for more details on path queries.

- 1172 • A *DataItem* reference (*DataItemRef*) modifies the set of resulting *Observations*,
 1173 limited by a path query parameter of a *current request* or *sample request*, to include
 1174 the *Observations* whose value for its *id* attribute matches the value provided for the
 1175 *idRef* attribute of the *DataItemRef* element. The result is equivalent to append-
 1176 ing `//[@id=<"idRef">]` to the path query parameters of the *current request* or
 1177 *sample request*. See *Section 4.1 - Agent* for more details on path queries.

1178 5.1.4 Agent

1179 *agent*.

1180 An *agent* **MUST** perform the following tasks:

- 1181 • Collect data from manufacturing equipment.
- 1182 • Generate *response documents*.
- 1183 • Provide a REST interface using Hypertext Transfer Protocol (HTTP).

1184 In addition to XML and HTTP, An *agent* **MAY** provide additional protocols and represen-
 1185 tations. Some representations **MAY** have companion specifications.

1186 5.1.4.1 Value Properties of Agent

1187 *Table 3* lists the Value Properties of Agent.

Value Property name	Value Property type	Multiplicity
instanceId	uInt32	1
sequenceNumber	uInt64	1
bufferSize	uInt32	1
maxAssets	uInt32	1
assetCount	uInt32	1

Table 3: Value Properties of Agent

1188 Descriptions for Value Properties of Agent:

- 1189 • instanceId
 1190 identifier for an *instance* of the *agent*.
 1191 instanceId **MUST** be changed to a different unique number each time the *buffer*
 1192 is cleared and a new set of data begins to be collected.
- 1193 • sequenceNumber
 1194 *sequence number*.

- 1195 • `bufferSize`
 1196 maximum number of *Observations* that **MAY** be retained in the *agent* that published
 1197 the *response document* at any point in time.
- 1198 • `maxAssets`
 1199 maximum number of *Assets* that **MAY** be retained in the *agent* that published the
 1200 *response document* at any point in time.
- 1201 • `assetCount`
 1202 current number of *Assets* that are currently stored in the *agent* as of the creation-
 1203 Time that the *agent* published the *response document*.

1204 **5.1.4.2 Part Properties of Agent**

1205 *Table 4* lists the Part Properties of Agent.

Part Property name	Multiplicity
Observation (organized by buffer)	0..*
Asset (organized by assetBuffer)	0..*

Table 4: Part Properties of Agent

1206 Descriptions for Part Properties of Agent:

- 1207 • `Observation`
 1208 abstract entity that provides telemetry data for a `DataItem` at a point in time.
 1209 *buffer* is a *buffer* for `Observation` types.
- 1210 • `Asset`
 1211 abstract *Asset*.
 1212 *assetBuffer* is an *asset buffer* for `Asset` types.

1213 **5.1.4.3 Operations for Agent**

- 1214 • `probe`
 1215 *agent* **MUST** respond to a successful *probe request* with an `MTConnectDevices`
 1216 entity containing either one, when a `Device` name or `uuid` is given, or all known
 1217 `Device` entries.

1218 When successful, an `MTConnectDevices` entity is returned and status code of
 1219 200. Otherwise an `MTConnectError` and an associated status code.

1220 The parameters for `Agent` are:

1221 – `device`
 1222 if present, specifies that only the `Device` for the given name or uuid will be
 1223 returned.

1224 If not present, all associated `Device` for the `Agent` will be returned.

1225 – `status`

1226 *HTTP Status Code.*

1227 The following *HTTP Status Codes* **MUST** be supported as possible responses
 1228 to a *probe request*:

1229 * Status Code: 200, Code Name: OK:

1230 The *request* succeeded.

1231 * Status Code: 400, Code Name: Bad Request:

1232 The *request* was invalid. The *response* **MUST** have an *MTConnectErrors*
 1233 *Response Document*.

1234 * Status Code: 404, Code Name: Not Found:

1235 The device name or uuid could not be located. The *response* **MUST** have
 1236 an *MTConnectErrors Response Document*.

1237 * Status Code: 405, Code Name: Method Not Allowed:

1238 The *request* specified a method other than GET

1239 * Status Code: 406, Code Name: Not Acceptable:

1240 The HTTP Accept Header in the *request* was not one of the supported
 1241 representations.

1242 * Status Code: 431, Code Name: Request Header Fields Too
 1243 Large:

1244 The fields in the *HTTP Request* exceed the limit of the implementation of
 1245 the *agent*.

1246 * Status Code: 500, Code Name: Internal Server Error:

1247 There was an unexpected error in the *agent* while responding to a *request*.

1248 – `return`

1249 *agent* **MUST** respond to a successful *probe request* with an *HTTP Status Code*
 1250 200 (OK) and an *MTConnectDevices Response Document*. If the *request* fails,
 1251 the *agent* **MUST** respond with an *MTConnectErrors Response Document* an
 1252 *HTTP Status Code* other than 200.

1253 `MTConnectDevices` if successful, `MTConnectError` otherwise.

1254 • `current`

1255 *agent* **MUST** respond to a successful *current request* with an `MTConnectStreams`
 1256 block containing the latest values for the selected observations. If the `at` parameter
 1257 is given, the values for the observations are a snapshot taken when the `lastSe-`
 1258 `quence` number was equal to the value of the `at` parameter.

1259 When successful, an `MTConnectStreams` entity is returned and status code of
 1260 200. Otherwise an `MTConnectError` and an associated status code.

1261 The parameters for `Agent` are:

1262 - `device`

1263 optional `Device` name or `uuid`. If not given, all devices are returned.

1264 - `path`

1265 XPath evaluated against the *Device Information Model* that references the *Com-*
 1266 *ponents* and *DataItems* to include in the *MTConnectStreams Response Docu-*
 1267 *ment*.

1268 When a `Component` element is referenced by the XPath, all observations for
 1269 its *DataItems* and related *Components* **MUST** be included in the *MTConnect-*
 1270 *Streams Response Document*.

1271 - `frequency`

1272 *agent* **MUST** stream samples and events to the client application pausing for
 1273 frequency milliseconds between each part. Each part will contain a maximum
 1274 of `count` events or samples and `from` will be used to indicate the beginning
 1275 of the stream.

1276 **DEPRECATED** Version 1.2, replace by `interval`

1277 - `at`

1278 *response documents* **MUST** include observations consistent with a specific *se-*
 1279 *quence number* given by the value of the `at` parameter.

1280 If the value is either less than the `firstSequence` or greater than the `last-`
 1281 `Sequence`, the *request* **MUST** return a 404 *HTTP Status Code* and the *agent*
 1282 **MUST** return an *MTConnectErrors Response Document* with an `OUT_OF_RANGE`
 1283 `errorCode`.

1284 The `at` parameter **MUST NOT** be used in conjunction with the `interval`
 1285 parameter.

1286 - `interval`

1287 *agent* **MUST** continuously publish *response documents* pausing for the num-
 1288 ber of milliseconds given as the value.

1289 The `interval` value **MUST** be in milliseconds, and **MUST** be a positive
 1290 integer greater than zero (0).

1291 The `interval` parameter **MUST NOT** be used in conjunction with the `at`
 1292 parameter.

- 1293 – status
- 1294 *HTTP Status Code.*
- 1295 The following *HTTP Status Codes* **MUST** be supported as possible responses
- 1296 to a *current request*:
- 1297 * Status Code: 200, Code Name: OK:
- 1298 The *request* succeeded.
- 1299 * Status Code: 400, Code Name: Bad Request:
- 1300 The *request* was invalid. The *response* **MUST** have an *MTCConnectErrors*
- 1301 *Response Document*.
- 1302 * Status Code: 404, Code Name: Not Found:
- 1303 One of the following conditions apply:
- 1304 · The device name or uuid could not be located.
- 1305 · The at was OUT_OF_RANGE range.
- 1306 The *response* **MUST** have an *MTCConnectErrors Response Document*.
- 1307 * Status Code: 405, Code Name: Method Not Allowed:
- 1308 The *request* specified a method other than GET
- 1309 * Status Code: 406, Code Name: Not Acceptable:
- 1310 The HTTP Accept Header in the *request* was not one of the supported
- 1311 representations.
- 1312 * Status Code: 431, Code Name: Request Header Fields Too
- 1313 Large:
- 1314 The fields in the *HTTP Request* exceed the limit of the implementation of
- 1315 the *agent*.
- 1316 * Status Code: 500, Code Name: Internal Server Error:
- 1317 There was an unexpected error in the *agent* while responding to a *request*.
- 1318 – return
- 1319 *agent* responds to a *current request* with an *MTCConnectStreams Response Doc-*
- 1320 *ument* that contains the current value of *Observations* associated with each
- 1321 piece of *streaming data* available from the *agent*, subject to any filtering de-
- 1322 fined in the *request*.
- 1323 • sample
- 1324 *agent* **MUST** respond to a successful *sample request* with an *MTCConnectStreams*
- 1325 entity containing the values for the selected observations according to the parameters
- 1326 provided.
- 1327 When successful, an *MTCConnectStreams* entity is returned and status code of
- 1328 200. Otherwise an *MTCConnectError* and an associated status code.
- 1329 The parameters for Agent are:

- 1330 – device
- 1331 optional Device name or uuid. If not given, all devices are returned.
- 1332 – path
- 1333 XPath evaluated against the *Device Information Model* that references the *Components* and *DataItems* to include in the *MTCConnectStreams Response Document*.
- 1334 When a Component element is referenced by the XPath, all observations for its *DataItems* and related *Components* **MUST** be included in the *MTCConnectStreams Response Document*.
- 1335
- 1336 When a Component element is referenced by the XPath, all observations for its *DataItems* and related *Components* **MUST** be included in the *MTCConnectStreams Response Document*.
- 1337
- 1338
- 1339 – from
- 1340 designates the *sequence number* of the first observation in the *buffer* the *agent* **MUST** consider publishing in the *response document*.
- 1341
- 1342 If from is zero (0), it **MUST** be set to the `firstSequence`, the oldest observation in the *buffer*.
- 1343
- 1344 If from and count parameters are not given, from **MUST** default to the `firstSequence`.
- 1345
- 1346 If the from parameter is less than the `firstSequence` or greater than `lastSequence`, the *agent* **MUST** return a 404 *HTTP Status Code* and **MUST** publish an *MTCConnectErrors Response Document* with an `OUT_OF_RANGE` `errorCode`.
- 1347
- 1348
- 1349
- 1350 – count
- 1351 designates the maximum number of observations the *agent* **MUST** publish in the *response document*.
- 1352
- 1353 The count **MUST NOT** be zero (0).
- 1354 When the count is greater than zero (0), the from parameter **MUST** default to the `firstSequence`. The evaluation of observations starts at from and moves forward accumulating newer observations until the number of observations equals the count or the observation at `lastSequence` is considered.
- 1355
- 1356 When the count is less than zero (0), the from parameter **MUST** default to the `lastSequence`. The evaluation of observations starts at from and moves backward accumulating older observations until the number of observations equals the absolute value of count or the observation at `firstSequence` is considered.
- 1357
- 1358 When the count is less than zero (0), the from parameter **MUST** default to the `lastSequence`. The evaluation of observations starts at from and moves backward accumulating older observations until the number of observations equals the absolute value of count or the observation at `firstSequence` is considered.
- 1359
- 1360
- 1361
- 1362
- 1363 count **MUST NOT** be less than zero (0) when an interval parameter is given.
- 1364
- 1365 If count is not provided, it **MUST** default to 100.
- 1366 If the absolute value of count is greater than the size of the *buffer* or equal to zero (0), the *agent* **MUST** return a 404 *HTTP Status Code* and **MUST** publish an *MTCConnectErrors Response Document* with an `OUT_OF_RANGE` `errorCode`.
- 1367
- 1368
- 1369

- 1370 If the `count` parameter is not a numeric value, the *agent* **MUST** return a
 1371 400 *HTTP Status Code* and **MUST** publish an *MTConnectErrors Response*
 1372 *Document* with an `INVALID_REQUEST` `errorCode`.
- 1373 – `frequency`
 1374 *agent* **MUST** stream samples and events to the client application pausing for
 1375 `frequency` milliseconds between each part. Each part will contain a maximum
 1376 of `count` events or samples and from will be used to indicate the beginning
 1377 of the stream.
 1378 **DEPRECATED** Version 1.2, replace by `interval`
 - 1379 – `heartbeat`
 1380 sets the time period for the *heartbeat* function in an *agent*.
 1381 The value for `heartbeat` represents the amount of time after a *response doc-*
 1382 *ument* has been published until a new *response document* **MUST** be published,
 1383 even when no new data is available.
 1384 The value for `heartbeat` is defined in milliseconds.
 1385 If no value is defined for `heartbeat`, the value **MUST** default to 10 seconds.
 1386 `heartbeat` **MUST** only be specified if `interval` is also specified.
 - 1387 – `interval`
 1388 *agent* **MUST** continuously publish *response documents* when the query pa-
 1389 rameters include `interval` using the value as the minimum period between
 1390 adjacent publications.
 1391 The `interval` value **MUST** be in milliseconds, and **MUST** be a positive
 1392 integer greater than or equal to zero (0).
 1393 If the value for the `interval` parameter is zero (0), the *agent* **MUST** publish
 1394 *response documents* when any observations become available.
 1395 If the period between the publication of a *response document* and reception of
 1396 observations exceeds the `interval`, the *agent* **MUST** wait for a maximum
 1397 of `heartbeat` milliseconds for observations. Upon the arrival of observa-
 1398 tions, the *agent* **MUST** immediately publish a *response document*. When the
 1399 period equals or exceeds the `heartbeat`, the *agent* **MUST** publish an empty
 1400 *response document*.
 - 1401 – `to`
 1402 specifies the *sequence number* of the observation in the *buffer* that will be the
 1403 upper bound of the observations in the *response document*.
 1404 Rules for `to` are as follows:
 - 1405 * The value of `to` **MUST** be an unsigned 64-bit integer.
 - 1406 * The value of `to` **MUST** be greater than the `firstSequence`.
 - 1407 * The value of `to` **MUST** be less than or equal to the `lastSequence`.

- 1408 * The value of `to` **MUST** be greater than `from`.
- 1409 * If `to` and `count` are given, the `count` parameter **MUST** be greater than
- 1410 zero.
- 1411 * If `to` and `count` are given, the maximum number of observations pub-
- 1412 lished in the *response document* **MUST NOT** be greater than the value of
- 1413 `count`.
- 1414 * If `to` is not given, see the `from` parameter for default behavior.
- 1415 * If the `to` parameter is less than the `firstSequence` or greater than
- 1416 `lastSequence`, the *agent* **MUST** return a 404 *HTTP Status Code*
- 1417 and **MUST** publish an *MTCConnectErrors Response Document* with an
- 1418 `OUT_OF_RANGE` `errorCode`.
- 1419 * If the `to` parameter is not a positive numeric value, the *agent* **MUST**
- 1420 return a 400 *HTTP Status Code* and **MUST** publish an *MTCConnectErrors*
- 1421 *Response Document* with an `INVALID_REQUEST` `errorCode`.
- 1422 * If the `to` parameter is less than the `from` parameter, the *agent* **MUST**
- 1423 return a 400 *HTTP Status Code* and **MUST** publish an *MTCConnectErrors*
- 1424 *Response Document* with an `INVALID_REQUEST` `errorCode`.
- 1425 * If the `to` parameter is given and the `count` parameter is less than zero,
- 1426 the *agent* **MUST** return a 400 *HTTP Status Code* and **MUST** publish
- 1427 an *MTCConnectErrors Response Document* with an `INVALID_REQUEST`
- 1428 `errorCode`.
- 1429 – `status`
- 1430 *HTTP Status Code*.
- 1431 The following *HTTP Status Codes* **MUST** be supported as possible responses
- 1432 to a *current request*:
- 1433 * Status Code: 200, Code Name: OK:
- 1434 The *request* succeeded.
- 1435 * Status Code: 400, Code Name: Bad Request:
- 1436 The *request* was invalid. The *response* **MUST** have an *MTCConnectErrors*
- 1437 *Response Document*.
- 1438 * Status Code: 404, Code Name: Not Found:
- 1439 One of the following conditions apply:
- 1440 · The device name or UUID could not be located.
- 1441 · One of the `asset_ids` could not be found.
- 1442 The *response* **MUST** have an *MTCConnectErrors Response Document*.
- 1443 * Status Code: 405, Code Name: Method Not Allowed:
- 1444 The *request* specified a method other than GET
- 1445 * Status Code: 406, Code Name: Not Acceptable:
- 1446 The HTTP Accept Header in the *request* was not one of the supported
- 1447 representations.

1448 * Status Code: 431, Code Name: Request Header Fields Too
 1449 Large:
 1450 The fields in the *HTTP Request* exceed the limit of the implementation of
 1451 the *agent*.

1452 * Status Code: 500, Code Name: Internal Server Error:
 1453 There was an unexpected error in the *agent* while responding to a *request*.

1454 – return
 1455 *agent* **MUST** respond to a successful *sample request* with an *HTTP Status*
 1456 *Code* 200 (OK) and an *MTCConnectStreams Response Document*. If the *request*
 1457 fails, the *agent* **MUST** respond with an *MTCConnectErrors Response Document*
 1458 an *HTTP Status Code* other than 200.

1459 • asset
 1460 *agent* **MUST** respond to a successful *asset request* with an *MTCConnectAssets*
 1461 entity with the selected asset entities according to the parameters provided.

1462 When successful, an *MTCConnectAssets* entity is returned and status code of 200.
 1463 Otherwise an *MTCConnectError* and an associated status code.

1464 The parameters for Agent are:

1465 – device
 1466 optional Device name or uuid. If not given, all devices are returned.

1467 – assetIds
 1468 path portion is a list of (asset_id) for specific *MTCConnectAssets Response*
 1469 *Documents*.

1470 In response, the *agent* returns an *MTCConnectAssets Response Document* that
 1471 contains information for the specific assets for each of the *asset_id* values
 1472 provided in the *request*. Each *asset_id* is separated by a “;”.

1473 – count
 1474 specifies the maximum number of *MTCConnectAssets Response Documents* re-
 1475 turned in an *MTCConnectAssets Response Document*.
 1476 If *count* is not given, the default value **MUST** be 100.

1477 – type
 1478 type of *Asset*. See *MTCConnect Standard: Part 4.0 - Asset Information Model*.

1479 – removed
 1480 value for *removed* **MUST** be *true* or *false* and interpreted as follows:

1481 * *true*: *MTCConnectAssets Response Documents* for assets marked as re-
 1482 moved **MUST** be included in the *response document*.

1483 * false: *MTConnectAssets Response Documents* for assets marked as re-
1484 removed **MUST NOT** be included in the *response document*.

1485 If `removed` is not given, the default value **MUST** be `false`.

1486 – `status`
1487 *HTTP Status Code*.

1488 The following *HTTP Status Codes* **MUST** be supported as possible responses
1489 to a *asset request*:

1490 * Status Code: 200, Code Name: OK:
1491 The *request* succeeded.

1492 * Status Code: 400, Code Name: Bad Request:
1493 The *request* was invalid. The *response* **MUST** have an *MTConnectErrors*
1494 *Response Document*.

1495 * Status Code: 404, Code Name: Not Found:
1496 One of the following conditions apply:

1497 · The device name or uuid could not be located.

1498 · The `from` or `to` was `OUT_OF_RANGE`.

1499 The *response* **MUST** have an *MTConnectErrors Response Document*.

1500 * Status Code: 405, Code Name: Method Not Allowed:
1501 The *request* specified a method other than `GET`

1502 * Status Code: 406, Code Name: Not Acceptable:
1503 The `HTTP Accept` Header in the *request* was not one of the supported
1504 representations.

1505 * Status Code: 431, Code Name: Request Header Fields Too
1506 Large:
1507 The fields in the *HTTP Request* exceed the limit of the implementation of
1508 the *agent*.

1509 * Status Code: 500, Code Name: Internal Server Error:
1510 There was an unexpected error in the *agent* while responding to a *request*.

1511 – `return`
1512 *MTConnectAssets Response Documents* provided in the *MTConnectAssets Re-*
1513 *sponse Document* will be limited to those specified in the combination of the
1514 path segment of the *asset request* and the parameters provided in the query
1515 segment of that *request*.

1516 5.2 MTConnectDevices Response Document

1517 This section provides semantic information for the `MTConnectDevices` entity.

1518 **5.2.1 MTConnectDevices**

1519 root entity of an *MTConnectDevices Response Document* that contains the *Device Information Model* of one or more Device entities.
 1520

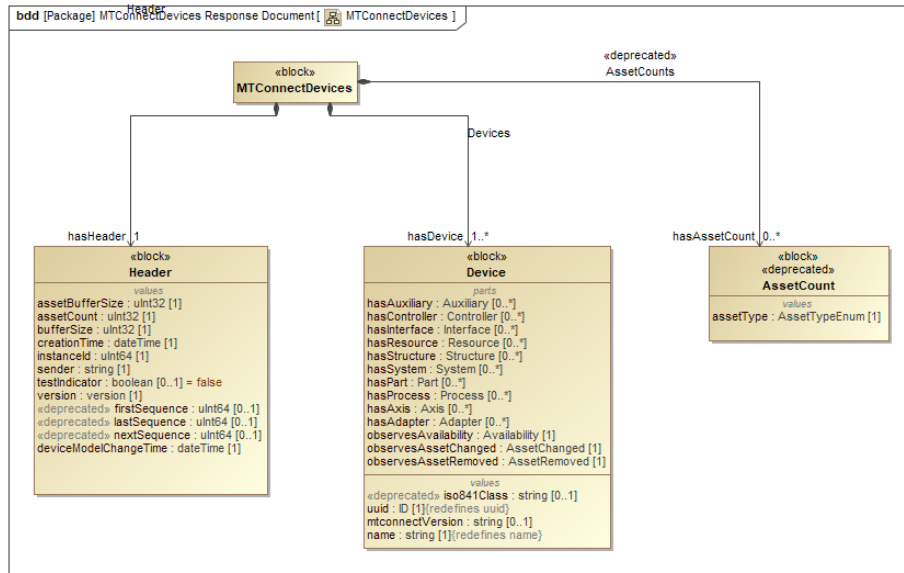


Figure 9: MTConnectDevices

1521 Note: Additional properties of MTConnectDevices **MAY** be defined for
 1522 schema and namespace declaration. See *Section C - Schema and Namespace*
 1523 *Declaration Information* for an XML example.

1524 **5.2.1.1 Part Properties of MTConnectDevices**

1525 *Table 5* lists the Part Properties of MTConnectDevices.

Part Property name	Multiplicity
Header	1
Device (organized by Devices)	1..*
<<deprecated>> AssetCount (organized by <<deprecated>> AssetCounts)	0..*

Table 5: Part Properties of MTConnectDevices

1526 Descriptions for Part Properties of MTConnectDevices:

- 1527 • Header

1528 provides information from an *agent* defining version information, storage capacity,
1529 and parameters associated with the data management within the *agent*.

1530 • Device

1531 Component composed of a piece of equipment that produces observations about
1532 itself.

1533 Devices groups one or more Device entities. See *MTCConnect Standard: Part*
1534 *2.0 - Device Information Model* for more detail.

1535 • <<deprecated>> AssetCount

1536 count of each asset type currently in the *agent*.

1537 AssetCounts groups AssetCount entities.

1538 5.2.2 Header

1539 provides information from an *agent* defining version information, storage capacity, and
1540 parameters associated with the data management within the *agent*.

1541 5.2.2.1 Value Properties of Header

1542 *Table 6* lists the Value Properties of Header.

Value Property name	Value Property type	Multiplicity
assetBufferSize	uInt32	1
assetCount	uInt32	1
bufferSize	uInt32	1
creationTime	dateTime	1
instanceId	uInt64	1
sender	string	1
testIndicator	boolean	0..1
version	version	1
<<deprecated>> firstSequence	uInt64	0..1
<<deprecated>> lastSequence	uInt64	0..1
<<deprecated>> nextSequence	uInt64	0..1
deviceModelChangeTime	dateTime	1

Table 6: Value Properties of Header

1543 Descriptions for Value Properties of Header:

1544 • `assetBufferSize`
 1545 maximum number of `Asset` types that can be stored in the *agent* that published the
 1546 *response document*.

1547 Note: The implementer is responsible for allocating the appropriate amount
 1548 of storage capacity required to accommodate the `assetBufferSize`.

1549 • `assetCount`
 1550 current number of `Asset` that are currently stored in the *agent* as of the `cre-`
 1551 `ationTime` that the *agent* published the *response document*.
 1552 `assetCount` **MUST NOT** be larger than the value reported for `assetBuffer-`
 1553 `Size`.

1554 • `bufferSize`
 1555 maximum number of *DataItems* that **MAY** be retained in the *agent* that published
 1556 the *response document* at any point in time.

1557 Note 1 to entry: `bufferSize` represents the maximum number of se-
 1558 quence numbers that **MAY** be stored in the *agent*.

1559 Note 2 to entry: The implementer is responsible for allocating the appro-
 1560 priate amount of storage capacity required to accommodate the `buffer-`
 1561 `Size`.

1562 • `creationTime`
 1563 timestamp that an *agent* published the *response document*.

1564 • `instanceId`
 1565 identifier for a specific instantiation of the *buffer* associated with the *agent* that pub-
 1566 lished the *response document*.
 1567 `instanceId` **MUST** be changed to a different unique number each time the *buffer*
 1568 is cleared and a new set of data begins to be collected.

1569 • `sender`
 1570 identification defining where the *agent* that published the *response document* is in-
 1571 stalled or hosted.
 1572 `sender` **MUST** be either an IP Address or Hostname describing where the *agent*
 1573 is installed or the URL of the *agent*; e.g., `http://<address>[:port]/`.

1574 Note: The port number need not be specified if it is the default HTTP
 1575 port 80.

- 1576 • `testIndicator`
 1577 indicates whether the *agent* that published the *response document* is operating in a
 1578 test mode.
 1579 If `testIndicator` is not specified, the value for `testIndicator` **MUST** be
 1580 interpreted to be `false`.
- 1581 • `version`
 1582 *major*, *minor*, and *revision* number of the MTConnect Standard that defines the
 1583 *semantic data model* that represents the content of the *response document*. It also
 1584 includes the revision number of the *schema* associated with that specific *semantic*
 1585 *data model*.
 1586 As an example, the value reported for `version` for a *response document* that was
 1587 structured based on *schema* revision 10 associated with Version 1.4.0 of the MT-
 1588 Connect Standard would be: 1.4.0.10
- 1589 • `<<deprecated>> firstSequence`
 1590 *sequence number* assigned to the oldest piece of *streaming data* stored in the *buffer*
 1591 of the *agent* immediately prior to the time that the *agent* published the *response*
 1592 *document*.
- 1593 • `<<deprecated>> lastSequence`
 1594 *sequence number* assigned to the last piece of *streaming data* that was added to
 1595 the *buffer* of the *agent* immediately prior to the time that the *agent* published the
 1596 *response document*.
- 1597 • `<<deprecated>> nextSequence`
 1598 *sequence number* of the piece of *streaming data* that is the next piece of data to be
 1599 retrieved from the *buffer* of the *agent* that was not included in the *response document*
 1600 published by the *agent*.
 1601 If the *streaming data* included in the *response document* includes the last piece of
 1602 data stored in the *buffer* of the *agent* at the time that the document was published,
 1603 then the value reported for `nextSequence` **MUST** be equal to `lastSequence`
 1604 + 1.
- 1605 • `deviceModelChangeTime`
 1606 timestamp of the last update of the `Device` information for any device.

1607 5.2.3 `<<deprecated>>AssetCount`

1608 count of each asset type currently in the *agent*.

1609 5.2.3.1 Value Properties of AssetCount

1610 *Table 7* lists the Value Properties of AssetCount.

Value Property name	Value Property type	Multiplicity
assetType	AssetTypeEnum	1

Table 7: Value Properties of AssetCount

1611 Descriptions for Value Properties of AssetCount:

- 1612 • assetType
- 1613 type of *Asset*.

1614 5.3 MTConnectStreams Response Document

1615 This section provides semantic information for the MTConnectStreams entity.

1616 5.3.1 MTConnectStreams

1617 root entity of an *MTConnectStreams Response Document* that contains the *Observation*
 1618 *Information Model* of one or more *Device* entities.

1619 Note: Additional properties of MTConnectStreams **MAY** be defined for
 1620 schema and namespace declaration. See *Section C - Schema and Namespace*
 1621 *Declaration Information* for an XML example.

1622 5.3.1.1 Part Properties of MTConnectStreams

1623 *Table 8* lists the Part Properties of MTConnectStreams.

Part Property name	Multiplicity
Header	1
DeviceStream (organized by Streams)	0..*

Table 8: Part Properties of MTConnectStreams

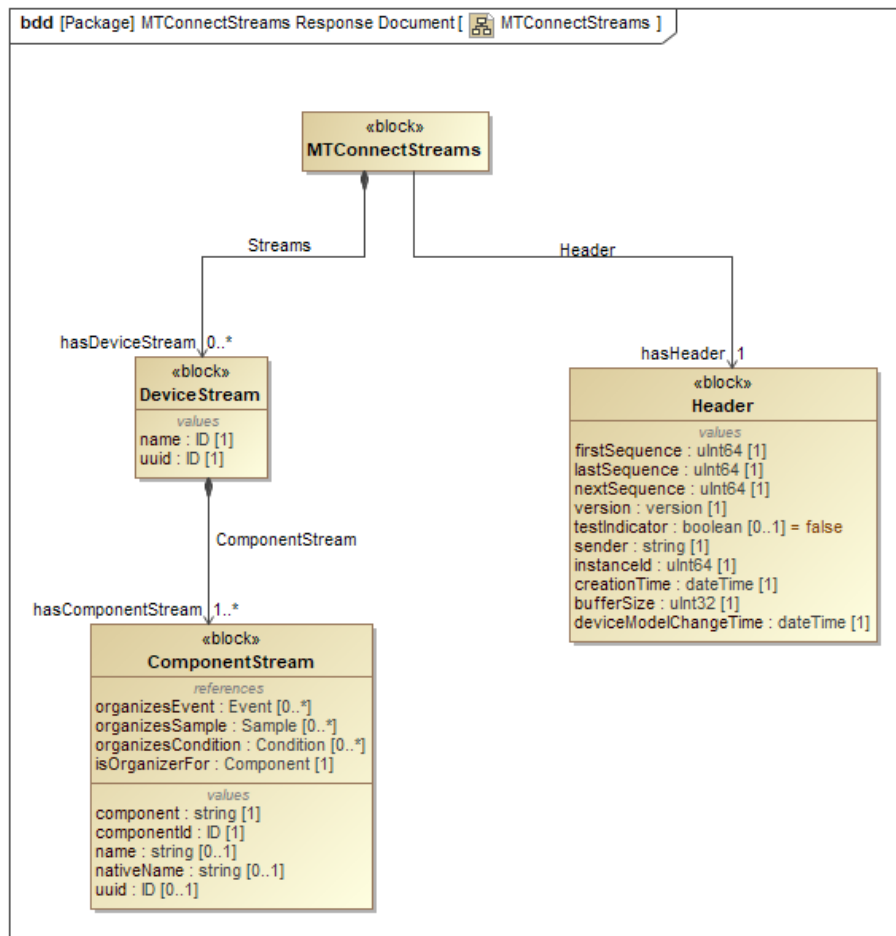


Figure 10: MTConnectStreams

1624 Descriptions for Part Properties of MTConnectStreams:

1625 • Header

1626 provides information from an *agent* defining version information, storage capacity,
 1627 and parameters associated with the data management within the *agent*.

1628 • DeviceStream

1629 *organizes* data reported from a Device.

1630 Streams groups one or more DeviceStream entities. See *MTConnect Stan-*
 1631 *dard: Part 3.0 - Observation Information Model* for more detail.

1632 5.3.2 Header

1633 provides information from an *agent* defining version information, storage capacity, and
1634 parameters associated with the data management within the *agent*.

1635 5.3.2.1 Value Properties of Header

1636 *Table 9* lists the Value Properties of Header.

Value Property name	Value Property type	Multiplicity
firstSequence	uInt64	1
lastSequence	uInt64	1
nextSequence	uInt64	1
version	version	1
testIndicator	boolean	0..1
sender	string	1
instanceId	uInt64	1
creationTime	dateTime	1
bufferSize	uInt32	1
deviceModelChangeTime	dateTime	1

Table 9: Value Properties of Header

1637 Descriptions for Value Properties of Header:

1638 • firstSequence

1639 *sequence number* assigned to the oldest piece of *streaming data* stored in the *buffer*
1640 of the *agent* immediately prior to the time that the *agent* published the *response*
1641 *document*.

1642 • lastSequence

1643 *sequence number* assigned to the last piece of *streaming data* that was added to
1644 the *buffer* of the *agent* immediately prior to the time that the *agent* published the
1645 *response document*.

1646 • nextSequence

1647 *sequence number* of the piece of *streaming data* that is the next piece of data to be
1648 retrieved from the *buffer* of the *agent* that was not included in the *response document*
1649 published by the *agent*.

1650 If the *streaming data* included in the *response document* includes the last piece of
 1651 data stored in the *buffer* of the *agent* at the time that the document was published,
 1652 then the value reported for `nextSequence` **MUST** be equal to `lastSequence`
 1653 + 1.

1654 • `version`

1655 *major*, *minor*, and *revision* number of the MTConnect Standard that defines the
 1656 *semantic data model* that represents the content of the *response document*. It also
 1657 includes the revision number of the *schema* associated with that specific *semantic*
 1658 *data model*.

1659 As an example, the value reported for `version` for a *response document* that was
 1660 structured based on *schema* revision 10 associated with Version 1.4.0 of the MT-
 1661 Connect Standard would be: 1.4.0.10

1662 • `testIndicator`

1663 indicates whether the *agent* that published the *response document* is operating in a
 1664 test mode.

1665 If `testIndicator` is not specified, the value for `testIndicator` **MUST** be
 1666 interpreted to be `false`.

1667 • `sender`

1668 identification defining where the *agent* that published the *response document* is in-
 1669 stalled or hosted.

1670 `sender` **MUST** be either an IP Address or Hostname describing where the *agent*
 1671 is installed or the URL of the *agent*; e.g., `http://<address>[:port]/`.

1672 Note: The port number need not be specified if it is the default HTTP
 1673 port 80.

1674 • `instanceId`

1675 identifier for a specific instantiation of the *buffer* associated with the *agent* that pub-
 1676 lished the *response document*.

1677 `instanceId` **MUST** be changed to a different unique number each time the *buffer*
 1678 is cleared and a new set of data begins to be collected.

1679 • `creationTime`

1680 timestamp that an *agent* published the *response document*.

1681 • `bufferSize`

1682 maximum number of *DataItems* that **MAY** be retained in the *agent* that published
 1683 the *response document* at any point in time.

1684 Note 1 to entry: bufferSize represents the maximum number of se-
 1685 quence numbers that **MAY** be stored in the *agent*.

1686 Note 2 to entry: The implementer is responsible for allocating the appro-
 1687 priate amount of storage capacity required to accommodate the buffer-
 1688 Size.

- 1689 • deviceModelChangeTime
 1690 timestamp of the last update of the Device information for any device.

1691 5.4 MTConnectAssets Response Document

1692 This section provides semantic information for the MTConnectAssets entity.

1693 5.4.1 MTConnectAssets

1694 root entity of an *MTConnectAssets Response Document* that contains the *Asset Information*
 1695 *Model* of Asset types.

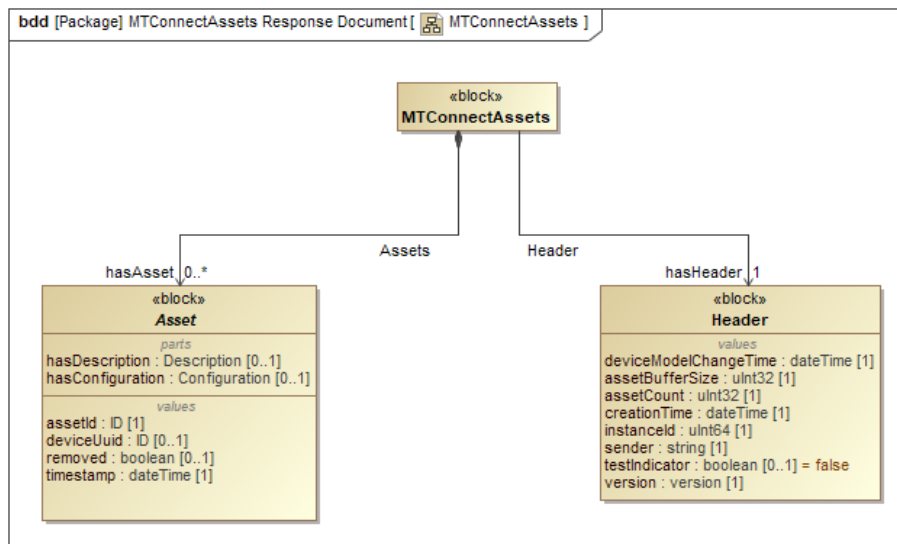


Figure 11: MTConnectAssets

1696 Note: Additional properties of MTConnectAssets **MAY** be defined for
 1697 schema and namespace declaration. See *Section C - Schema and Namespace*
 1698 *Declaration Information* for an XML example.

1699 5.4.1.1 Part Properties of MTConnectAssets

1700 *Table 10* lists the Part Properties of MTConnectAssets.

Part Property name	Multiplicity
Header	1
Asset (organized by Assets)	0..*

Table 10: Part Properties of MTConnectAssets

1701 Descriptions for Part Properties of MTConnectAssets:

- 1702 • Header
- 1703 provides information from an *agent* defining version information, storage capacity,
- 1704 and parameters associated with the data management within the *agent*.
- 1705 • Asset
- 1706 abstract *Asset*.
- 1707 *Assets* groups one or more *Asset* types. See *MTConnect Standard: Part 4.0 -*
- 1708 *Asset Information Model* for more details.

1709 5.4.2 Header

1710 provides information from an *agent* defining version information, storage capacity, and
 1711 parameters associated with the data management within the *agent*.

1712 5.4.2.1 Value Properties of Header

1713 *Table 11* lists the Value Properties of Header.

Value Property name	Value Property type	Multiplicity
deviceModelChangeTime	dateTime	1
assetBufferSize	uInt32	1
assetCount	uInt32	1
creationTime	dateTime	1
instanceId	uInt64	1
sender	string	1
testIndicator	boolean	0..1
version	version	1

Table 11: Value Properties of Header

1714 Descriptions for Value Properties of Header:

- 1715 • deviceModelChangeTime
1716 timestamp of the last update of the *Device* information for any device.
- 1717 • assetBufferSize
1718 maximum number of *Asset* types that can be stored in the *agent* that published the
1719 *response document*.
- 1720 Note: The implementer is responsible for allocating the appropriate amount
1721 of storage capacity required to accommodate the *assetBufferSize*.
- 1722 • assetCount
1723 current number of *Asset* that are currently stored in the *agent* as of the *cre-*
1724 *ationTime* that the *agent* published the *response document*.
- 1725 *assetCount* **MUST NOT** be larger than the value reported for *assetBuffer-*
1726 *Size*.
- 1727 • creationTime
1728 timestamp that an *agent* published the *response document*.
- 1729 • instanceId
1730 identifier for a specific instantiation of the *buffer* associated with the *agent* that pub-
1731 lished the *response document*.
- 1732 *instanceId* **MUST** be changed to a different unique number each time the *buffer*
1733 is cleared and a new set of data begins to be collected.

- 1734 • sender
1735 identification defining where the *agent* that published the *response document* is in-
1736 stalled or hosted.
1737 sender **MUST** be either an IP Address or Hostname describing where the *agent*
1738 is installed or the URL of the *agent*; e.g., `http://<address>[:port]/`.
- 1739 Note: The port number need not be specified if it is the default HTTP
1740 port 80.
- 1741 • testIndicator
1742 indicates whether the *agent* that published the *response document* is operating in a
1743 test mode.
1744 If testIndicator is not specified, the value for testIndicator **MUST** be
1745 interpreted to be false.
- 1746 • version
1747 *major*, *minor*, and *revision* number of the MTConnect Standard that defines the
1748 *semantic data model* that represents the content of the *response document*. It also
1749 includes the revision number of the *schema* associated with that specific *semantic*
1750 *data model*.
1751 As an example, the value reported for version for a *response document* that was
1752 structured based on *schema* revision 10 associated with Version 1.4.0 of the MT-
1753 Connect Standard would be: 1.4.0.10

1754 6 Error Information Model

1755 The *Error Information Model* establishes the rules and terminology that describes the *re-*
 1756 *response document* returned by an *agent* when it encounters an error while interpreting a
 1757 *request* for information from a client software application or when an *agent* experiences
 1758 an error while publishing the *response* to a *request* for information.

1759 An *agent* provides the information regarding errors encountered when processing a *request*
 1760 for information by publishing an *MTConnectErrors Response Document* to the client soft-
 1761 ware application that made the *request* for information.

1762 6.1 MTConnectErrors Response Document

1763 This section provides semantic information for the `MTConnectErrors` entity.

1764 6.1.1 MTConnectError

1765 root entity of an *MTConnectErrors Response Document* that contains the *Error Informa-*
 1766 *tion Model*.

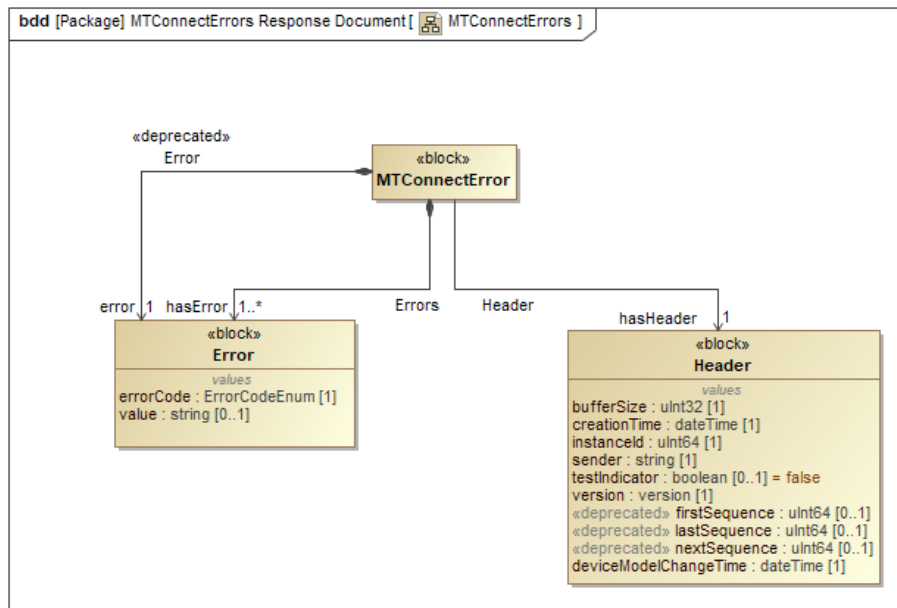


Figure 12: MTConnectError

1767 Note: Additional properties of `MTConnectError` **MAY** be defined for schema
 1768 and namespace declaration. See *Section C - Schema and Namespace Decla-*
 1769 *ration Information* for an XML example.

1770 6.1.1.1 Part Properties of `MTConnectError`

1771 *Table 12* lists the Part Properties of `MTConnectError`.

Part Property name	Multiplicity
Header	1
Error (organized by Errors)	1..*
<<deprecated>> Error	1

Table 12: Part Properties of `MTConnectError`

1772 Descriptions for Part Properties of `MTConnectError`:

- 1773 • Header

1774 provides information from an *agent* defining version information, storage capacity,
 1775 and parameters associated with the data management within the *agent*.

- 1776 • Error

1777 error encountered by an *agent* when responding to a *request*.

1778 `Errors` groups one or more `Error` entities. See *Section 6.1.3 - Error*.

1779 Note: When compatibility with Version 1.0.1 and earlier of the `MTCon-`
 1780 `nect Standard` is required for an implementation, the *MTConnectErrors*
 1781 *Response Document* contains only a single `Error` entity and the `Er-`
 1782 `rors` entity **MUST NOT** appear in the document.

- 1783 • Error

1784 error encountered by an *agent* when responding to a *request*.

1785 6.1.2 Header

1786 provides information from an *agent* defining version information, storage capacity, and
 1787 parameters associated with the data management within the *agent*.

1788 **6.1.2.1 Value Properties of Header**1789 *Table 13* lists the Value Properties of Header.

Value Property name	Value Property type	Multiplicity
bufferSize	uInt32	1
creationTime	dateTime	1
instanceId	uInt64	1
sender	string	1
testIndicator	boolean	0..1
version	version	1
<<deprecated>> firstSequence	uInt64	0..1
<<deprecated>> lastSequence	uInt64	0..1
<<deprecated>> nextSequence	uInt64	0..1
deviceModelChangeTime	dateTime	1

Table 13: Value Properties of Header

1790 Descriptions for Value Properties of Header:

1791 • bufferSize

1792 maximum number of *DataItems* that **MAY** be retained in the *agent* that published
1793 the *response document* at any point in time.

1794 Note 1 to entry: bufferSize represents the maximum number of se-
1795 quence numbers that **MAY** be stored in the *agent*.

1796 Note 2 to entry: The implementer is responsible for allocating the appro-
1797 priate amount of storage capacity required to accommodate the buffer-
1798 Size.

1799 • creationTime

1800 timestamp that an *agent* published the *response document*.

1801 • instanceId

1802 identifier for a specific instantiation of the *buffer* associated with the *agent* that pub-
1803 lished the *response document*.

1804 instanceId **MUST** be changed to a different unique number each time the *buffer*
1805 is cleared and a new set of data begins to be collected.

- 1806 • sender
 1807 identification defining where the *agent* that published the *response document* is in-
 1808 stalled or hosted.
 1809 sender **MUST** be either an IP Address or Hostname describing where the *agent*
 1810 is installed or the URL of the *agent*; e.g., `http://<address>[:port]/`.
- 1811 Note: The port number need not be specified if it is the default HTTP
 1812 port 80.
- 1813 • testIndicator
 1814 indicates whether the *agent* that published the *response document* is operating in a
 1815 test mode.
 1816 If testIndicator is not specified, the value for testIndicator **MUST** be
 1817 interpreted to be false.
- 1818 • version
 1819 *major*, *minor*, and *revision* number of the MTConnect Standard that defines the
 1820 *semantic data model* that represents the content of the *response document*. It also
 1821 includes the revision number of the *schema* associated with that specific *semantic*
 1822 *data model*.
 1823 As an example, the value reported for version for a *response document* that was
 1824 structured based on *schema* revision 10 associated with Version 1.4.0 of the MT-
 1825 Connect Standard would be: 1.4.0.10
- 1826 • <<deprecated>> firstSequence
 1827 *sequence number* assigned to the oldest piece of *streaming data* stored in the *buffer*
 1828 of the *agent* immediately prior to the time that the *agent* published the *response*
 1829 *document*.
- 1830 • <<deprecated>> lastSequence
 1831 *sequence number* assigned to the last piece of *streaming data* that was added to
 1832 the *buffer* of the *agent* immediately prior to the time that the *agent* published the
 1833 *response document*.
- 1834 • <<deprecated>> nextSequence
 1835 *sequence number* of the piece of *streaming data* that is the next piece of data to be
 1836 retrieved from the *buffer* of the *agent* that was not included in the *response document*
 1837 published by the *agent*.
 1838 If the *streaming data* included in the *response document* includes the last piece of
 1839 data stored in the *buffer* of the *agent* at the time that the document was published,

1840 then the value reported for `nextSequence` **MUST** be equal to `lastSequence`
 1841 + 1.

- 1842 • `deviceModelChangeTime`
- 1843 timestamp of the last update of the `Device` information for any device.

1844 6.1.3 Error

1845 error encountered by an *agent* when responding to a *request*.

1846 The value of `Error` **MUST** be `string`.

1847 6.1.3.1 Value Properties of Error

1848 *Table 14* lists the Value Properties of `Error`.

Value Property name	Value Property type	Multiplicity
<code>errorCode</code>	<code>ErrorCodeEnum</code>	1

Table 14: Value Properties of Error

1849 Descriptions for Value Properties of `Error`:

- 1850 • `errorCode`
- 1851 descriptive code that indicates the type of error that was encountered by an *agent*.
- 1852 `ErrorCodeEnum` Enumeration:
 - 1853 – `ASSET_NOT_FOUND`
 - 1854 *request* for information specifies an `Asset` that is not recognized by the *agent*.
 - 1855 – `INTERNAL_ERROR`
 - 1856 *agent* experienced an error while attempting to published the requested infor-
 1857 mation.
 - 1858 – `INVALID_REQUEST`
 - 1859 *request* contains information that was not recognized by the *agent*.
 - 1860 – `INVALID_URI`
 - 1861 Uniform Resource Identifier (URI) provided was incorrect.

- 1862 – INVALID_XPATH
 1863 XML Path Language (XPath) identified in the *request* for information could
 1864 not be parsed correctly by the *agent*.
 1865 This could be caused by an invalid syntax or the XPath did not match a valid
 1866 identify for any information stored in the *agent*.
- 1867 – NO_DEVICE
 1868 identity of the *Device* specified in the *request* for information is not associ-
 1869 ated with the *agent*.
- 1870 – OUT_OF_RANGE
 1871 *request* for information specifies *streaming data* that includes sequence num-
 1872 ber(s) for pieces of data that are beyond the end of the *buffer*.
- 1873 – QUERY_ERROR
 1874 *agent* was unable to interpret the query.
 1875 The query parameters do not contain valid values or include an invalid param-
 1876 eter.
- 1877 – TOO_MANY
 1878 *count* parameter provided in the *request* for information requires either of the
 1879 following:
 1880 * *streaming data* that includes more pieces of data than the *agent* is capable
 1881 of organizing in an *MTCConnectStreams Response Document*.
 1882 * *Assets* that include more *Asset* in an *MTCConnectAssets Response Doc-*
 1883 *ument* than the *agent* is capable of handling.
- 1884 – UNAUTHORIZED
 1885 *requester* does not have sufficient permissions to access the requested informa-
 1886 tion.
- 1887 – UNSUPPORTED
 1888 valid *request* was provided, but the *agent* does not support the feature or type
 1889 of *request*.

1890 7 Profile

1891 MTConnect Profile is a *profile* that extends the Systems Modeling Language (SysML)
 1892 metamodel for the MTConnect domain using additional data types and *stereotypes*.

1893 7.1 DataTypes

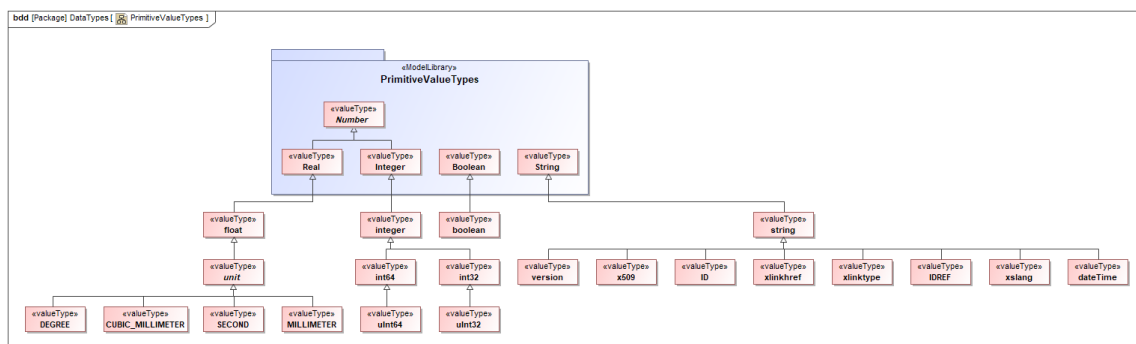


Figure 13: DataTypes

1894 7.1.1 boolean

1895 primitive type.

1896 7.1.2 ID

1897 string that represents an identifier (ID).

1898 7.1.3 string

1899 primitive type.

1900 7.1.4 float

1901 primitive type.

1902 7.1.5 dateTime

1903 string that represents timestamp in ISO 8601 format.

1904 7.1.6 integer

1905 primitive type.

1906 7.1.7 xlinktype

1907 string that represents the type of an XLink element. See <https://www.w3.org/TR/xlink11/>.

1909 7.1.8 xslang

1910 string that represents a language tag. See <http://www.ietf.org/rfc/rfc4646.txt>.

1912 7.1.9 SECOND

1913 float that represents time in seconds.

1914 7.1.10 IDREF

1915 string that represents a reference to an ID.

1916 7.1.11 xlinkhref

1917 string that represents the locator attribute of an XLink element. See <https://www.w3.org/TR/xlink11/>.

1919 7.1.12 x509

1920 string that represents an x509 data block. *Ref ISO/IEC 9594-8:2020.*

1921 7.1.13 int32

1922 32-bit integer.

1923 7.1.14 int64

1924 64-bit integer.

1925 7.1.15 version

1926 series of four numeric values, separated by a decimal point, representing a *major*, *minor*,
1927 and *revision* number of the MTConnect Standard and the revision number of a specific
1928 *schema*.

1929 7.1.16 uInt32

1930 32-bit unsigned integer.

1931 7.1.17 uInt64

1932 64-bit unsigned integer.

1933 7.2 Stereotypes

1934 7.2.1 organizer

1935 element that *organizes* other elements of a type.

1936 7.2.2 deprecated

1937 element that has been deprecated.

1938 7.2.3 extensible

1939 enumeration that can be extended.

1940 7.2.4 informative

1941 element that is descriptive and non-normative.

1942 7.2.5 valueType

1943 extends SysML <<ValueType>> to include `Class` as a value type.

1944 7.2.6 normative

1945 element that has been added to the standard.

1946 7.2.7 observes

1947 association in which a *Component* makes *Observations* about an observable *DataItem*.

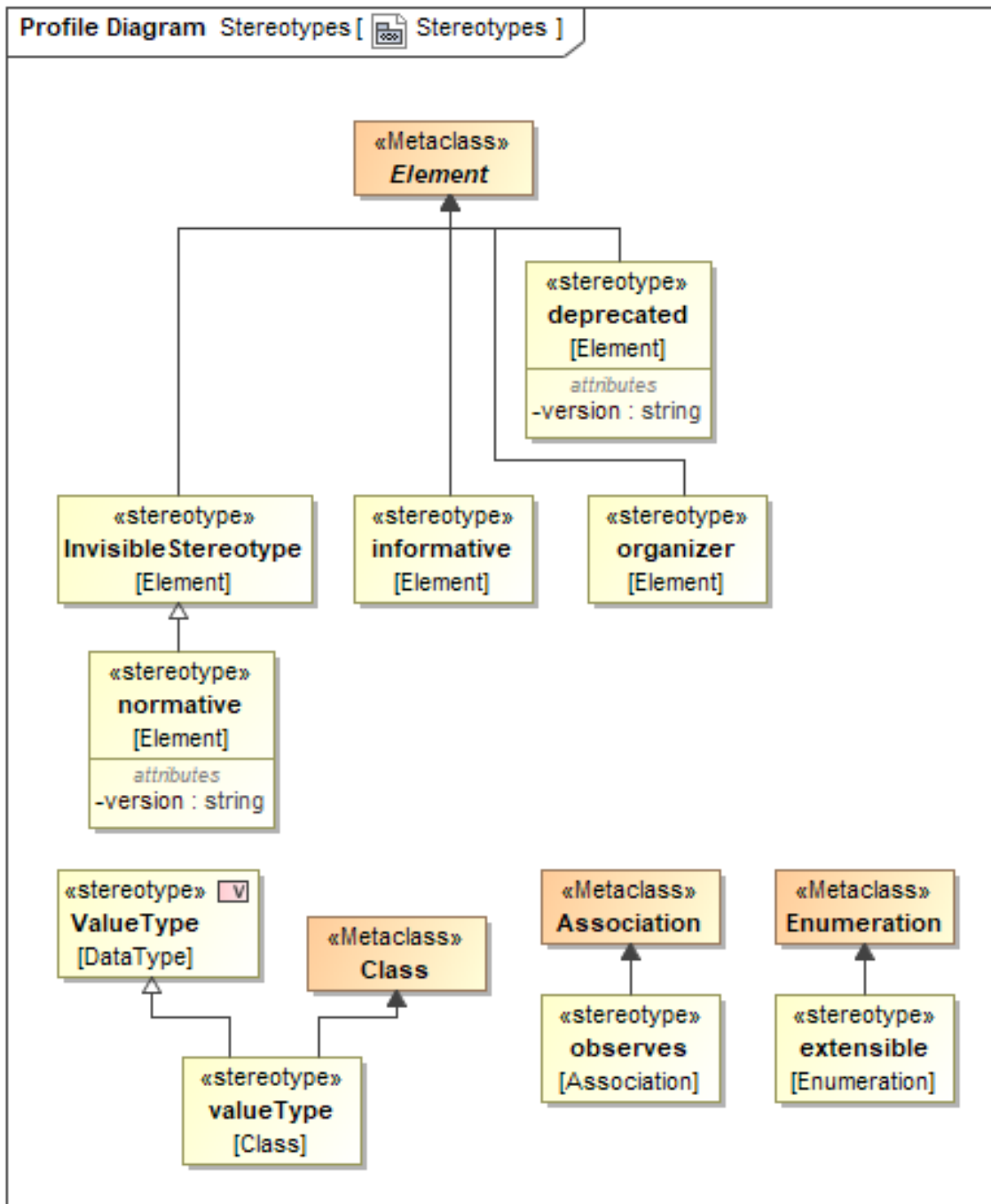


Figure 14: Stereotypes

1948 Appendices

1949 A Bibliography

- 1950 Engineering Industries Association. EIA Standard - EIA-274-D, Interchangeable Variable,
1951 Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically
1952 Controlled Machines. Washington, D.C. 1979.
- 1953 ISO TC 184/SC4/WG3 N1089. ISO/DIS 10303-238: Industrial automation systems and
1954 integration Product data representation and exchange Part 238: Application Protocols: Ap-
1955 plication interpreted model for computerized numerical controllers. Geneva, Switzerland,
1956 2004.
- 1957 International Organization for Standardization. ISO 14649: Industrial automation sys-
1958 tems and integration – Physical device control – Data model for computerized numerical
1959 controllers – Part 10: General process data. Geneva, Switzerland, 2004.
- 1960 International Organization for Standardization. ISO 14649: Industrial automation sys-
1961 tems and integration – Physical device control – Data model for computerized numerical
1962 controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.
- 1963 International Organization for Standardization. ISO 6983/1 – Numerical Control of ma-
1964 chines – Program format and definition of address words – Part 1: Data format for posi-
1965 tioning, line and contouring control systems. Geneva, Switzerland, 1982.
- 1966 Electronic Industries Association. ANSI/EIA-494-B-1992, 32 Bit Binary CL (BCL) and
1967 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines.
1968 Washington, D.C. 1992.
- 1969 National Aerospace Standard. Uniform Cutting Tests - NAS Series: Metal Cutting Equip-
1970 ment Specifications. Washington, D.C. 1969.
- 1971 International Organization for Standardization. ISO 10303-11: 1994, Industrial automa-
1972 tion systems and integration Product data representation and exchange Part 11: Descrip-
1973 tion methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.
- 1974 International Organization for Standardization. ISO 10303-21: 1996, Industrial automa-
1975 tion systems and integration – Product data representation and exchange – Part 21: Imple-
1976 mentation methods: Clear text encoding of the exchange structure. Geneva, Switzerland,
1977 1996.
- 1978 H.L. Horton, F.D. Jones, and E. Oberg. Machinery's Handbook. Industrial Press, Inc.

- 1979 New York, 1984.
- 1980 International Organization for Standardization. ISO 841-2001: Industrial automation systems and integration - Numerical control of machines - Coordinate systems and motion
1981 nomenclature. Geneva, Switzerland, 2001.
1982
- 1983 ASME B5.57: Methods for Performance Evaluation of Computer Numerically Controlled
1984 Lathes and Turning Centers, 1998.
- 1985 ASME/ANSI B5.54: Methods for Performance Evaluation of Computer Numerically Controlled Machining Centers. 2005.
1986
- 1987 OPC Foundation. OPC Unified Architecture Specification, Part 1: Concepts Version 1.00.
1988 July 28, 2006.
- 1989 IEEE STD 1451.0-2007, Standard for a Smart Transducer Interface for Sensors and Actuators – Common Functions, Communication Protocols, and Transducer Electronic Data
1990 Sheet (TEDS) Formats, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH99684,
1991
1992 October 5, 2007.
1993
- 1994 IEEE STD 1451.4-1994, Standard for a Smart Transducer Interface for Sensors and Actuators – Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet
1995 (TEDS) Formats, IEEE Instrumentation and Measurement Society, TC-9, The Institute of
1996
1997 Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH95225, December
1998 15, 2004.

1999 B Fundamentals of Using XML to Encode Response Documents

2000 The MTConnect Standard specifies the structures and constructs that are used to encode
 2001 *response documents*. When these *response documents* are encoded using XML, there are
 2002 additional rules defined by the XML standard that apply for creating an XML compliant
 2003 document. An implementer should refer to the W3C website for additional information on
 2004 XML documentation and implementation details - <http://www.w3.org/XML>.

2005 The following provides specific terms and guidelines referenced in the MTConnect Stan-
 2006 dard for forming *response documents* with XML:

2007 • **tag**: A tag is an XML construct that forms the foundation for an XML expression.
 2008 It defines the scope (beginning and end) of an XML expression. The main types of
 2009 tags are:

2010 • **start-tag**: Designates the beginning on an XML element; e.g., `<element name>`

2011 • **end-tag**: Designates the end on an XML element; e.g., `</element name>`.

2012 Note: If an element has no *child elements* or Character Data (CDATA), the
 2013 end-tag may be shortened to `/>`.

2014 • **Element**: An element is an XML statement that is the primary building block
 2015 for a document encoded using XML. An element begins with a `start-tag` and
 2016 ends with a matching `end-tag`. The characters between the `start-tag` and the
 2017 `end-tag` are the element's content. The content may contain attributes, CDATA,
 2018 and/or other elements. If the content contains additional elements, these elements
 2019 are called *child elements*.

2020 An example would be: `<element name>Content of the Element</element name>`.

2021 • **child element**: An XML element that is contained within a higher-level *parent ele-*
 2022 *ment*. A *child element* is also known as a sub-element. XML allows an unlimited
 2023 hierarchy of *parent element-child element* relationships that establishes the struc-
 2024 ture that defines how the various pieces of information in the document relate to
 2025 each other. A *parent element* may have multiple associated *child elements*.

2026 • **element name**: A descriptive identifier contained in both the `start-tag` and `end-`
 2027 `tag` that provides the name of an XML element.

- 2028 • **Attribute:** A construct consisting of a name–value pair that provides additional
 2029 information about that XML element. The format for an attribute is ‘name=’value’;
 2030 where the value for the attribute is enclosed in a set of quotation (‘’) marks. An XML
 2031 attribute **MUST** only have a single value and each attribute can appear at most once
 2032 in each element. Also, each attribute **MUST** be defined in a *schema* to either be
 2033 required or optional.
- 2034 • An example of attributes for an XML element is *Example 4*:

Example 4: Example of attributes for an element

```
2035 1 <DataItem category="SAMPLE" id="S1load"
2036 2   nativeUnits="PERCENT" type="LOAD"
2037 3   units="PERCENT"/>
```

2038 In this example, `DataItem` is the *element name*. `category`, `id`, `nativeUnits`,
 2039 `type`, and `units` are the names of the attributes. “SAMPLE”, “S1load”, “PERCENT”,
 2040 “LOAD”, and “PERCENT” are the values for each of the respective attributes.

- 2041 • **CDATA:** CDATA is an XML term representing *Character Data*. *Character Data*
 2042 contains a value(s) or text that is associated with an XML element. CDATA can be
 2043 restricted to certain formats, patterns, or words.

2044 An example of CDATA associated with an XML element would be *Example 5*:

Example 5: Example of cdata associated with element

```
2045 1 <Message id="M1">This is some text</Message>
```

2046 In this example, `Message` is the *element name* and `This is some text` is the CDATA.

- 2047 • **namespace:** An XML *namespace* defines a unique vocabulary for named elements
 2048 and attributes in an XML document. An XML document may contain content that is
 2049 associated with multiple *namespaces*. Each *namespace* has its own unique identifier.

2050 Elements and attributes are associated with a specific *namespace* by placing a prefix on
 2051 the name of the element or attribute that associates that name to a specific *namespace*; e.g.,
 2052 `x:MyTarget` associates the element name `MyTarget` with the *namespace* designated
 2053 by `x`: (the prefix).

2054 *namespaces* are used to avoid naming conflicts within an XML document. The nam-
 2055 ing convention used for elements and attributes may be associated with either the default

2056 *namespace* specified in the header of an XML document or they may be associated with
 2057 one or more alternate *namespaces*. All elements or attributes associated with a *namespace*
 2058 that is not the default *namespace*, must include a prefix (e.g., x:) as part of the name of
 2059 the element or attribute to associate it with the proper *namespace*. See *Section C - Schema*
 2060 *and Namespace Declaration Information* for details on the structure for XML headers.

2061 The names of the elements and attributes declared in a *namespace* may be identified with
 2062 a different prefix than the prefix that signifies that specific *namespace*. These prefixes are
 2063 called *namespace* aliases. As an example, MTConnect Standard specific *namespaces* are
 2064 designated as m: and the names of the elements and attributes defined in that *namespace*
 2065 have an alias prefix of mt: which designates these names as MTConnect Standard specific
 2066 vocabulary; e.g., mt:MTConnectDevices.

2067 XML documents are encoded with a hierarchy of elements. In general, XML elements
 2068 may contain *child elements*, CDATA, or both. However, in the MTConnect Standard,
 2069 an element **MUST NOT** contain mixed content; meaning it cannot contain both *child*
 2070 *elements* and CDATA.

2071 The *semantic data model* defined for each *response document* specifies the elements and
 2072 *child elements* that may appear in a document. The *semantic data model* also defines the
 2073 number of times each element and *child element* may appear in the document.

2074 *Example 6* demonstrates the hierarchy of XML elements and *child elements* used to form
 2075 an XML document:

Example 6: Example of hierarchy of XML elements

```

2076 1 <Root Level>      (Parent Element)
2077 2   <First Level>  (Child Element to Root Level and
2078 3   Parent Element to Second Level)
2079 4     <Second Level> (Child Element to First Level
2080 5     and Parent Element to Third Level)
2081 6       <Third Level name="N1"></Third Level>
2082 7       (Child Element to Second Level)
2083 8       <Third Level name="N2"></Third Level>
2084 9       (Child Element to Second Level)
2085 10      <Third Level name="N3"></Third Level>
2086 11      (Child Element to Second Level)
2087 12      </Second Level>  (end-tag for Second Level)
2088 13      </First Level>   (end-tag for First Level)
2089 14      </Root Level>    (end-tag for Root Level)
  
```

2090 In the *Example 6*, *Root Level* and *First Level* have one *child element* (sub-elements) each
 2091 and *Second Level* has three *child elements*; each called *Third Level*. Each *Third Level*
 2092 element has a different name attribute. Each level in the structure is an element and each
 2093 lower level element is a *child element*.

2094 C Schema and Namespace Declaration Information

2095 There are four pseudo-attributes typically included in the header of a *response document*
 2096 that declare the *schema* and *namespace* for the document. Each of these pseudo-attributes
 2097 provides specific information for a client software application to properly interpret the
 2098 content of the *response document*.

2099 The pseudo-attributes include:

2100 • `xmlns:xsi` – The `xsi` portion of this attribute name stands for *XML Schema*
 2101 instance. An *XML Schema* instance provides information that may be used by a
 2102 software application to interpret XML specific information within a document. See
 2103 the W3C website for more details on `xmlns:xsi`.

2104 • `xmlns` – Declares the default *namespace* associated with the content of the *re-*
 2105 *sponse document*. The default *namespace* is considered to apply to all elements and
 2106 attributes whenever the name of the element or attribute does not contain a prefix
 2107 identifying an alternate *namespace*.

2108 The value of this attribute is an URN identifying the name of the file that defines the details
 2109 of the *namespace* content. This URN provides a unique identify for the *namespace*.

2110 • `xmlns:m` – Declares the MTConnect specific *namespace* associated with the con-
 2111 tent of the *response document*. There may be multiple *namespaces* declared for an
 2112 XML document. Each may be associated to the default *namespace* or it may be to-
 2113 tally independent. The `:m` designates that this is a specific MTConnect *namespace*
 2114 which is directly associated with the default *namespace*.

2115 Note: See *Section D - Extensibility* for details regarding extended *namespaces*.

2116 The value associated with this attribute is an URN identifying the name of the file that
 2117 defines the details of the *namespace* content.

2118 • `xsi:schemaLocation` - Declares the name for the *schema* associated with the
 2119 *response document* and the location of the file that contains the details of the *schema*
 2120 for that document.

2121 The value associated with this attribute has two parts:

- 2122 • A URN identifying the name of the specific *XML Schema* instance associated with
2123 the *response document*.
- 2124 • The path to the location where the file describing the specific *XML Schema* instance
2125 is located. If the file is located in the same root directory where the *agent* is installed,
2126 then the local path **MAY** be declared. Otherwise, a fully qualified URL must be
2127 declared to identify the location of the file.

2128 Note: In the format of the value associated with `xsi:schemaLocation`,
2129 the URN and the path to the *schema* file **MUST** be separated by a “space”.

2130 In *Example 7*, the first line is the XML declaration. The second line is a *root element*
2131 called `MTConnectDevices`. The remaining four lines are the pseudo-attributes of
2132 `MTConnectDevices` that declare the XML *schema* and *namespace* associated with
2133 an *MTConnectDevices Response Document*.

Example 7: Example of schema and namespace declaration

```
2134 1 <?xml version="1.0" encoding="UTF-8"?>  
2135 2 <MTConnectDevices  
2136 3   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance  
2137 4   xmlns="urn:mtconnect.org:MTConnectDevices:1.3"  
2138 5   xmlns:m="urn:mtconnect.org:MTConnectDevices:1.3"  
2139 6   xsi:schemaLocation="urn:mtconnect.org:  
2140 7   _____MTConnectDevices:1.3_/schemas/MTConnectDevices\textunderscore_  
2141   1.3.xsd">
```

2142 The format for the values provided for each of the pseudo-attributes **MUST** reference
2143 the *semantic data model* (e.g., `MTConnectDevices`, `MTConnectStreams`, `MTCon-`
2144 `nectAssets`, or `MTConnectError`) and the version (i.e.; 1.1, 1.2, 1.3, etc.) of the
2145 `MTConnect` Standard that depict the *schema* and *namespace(s)* associated with a specific
2146 *response document*.

2147 When an implementer chooses to extend an `MTConnect data model` by adding custom data
2148 types or additional *structural elements*, the *schema* and *namespace* for that *data model*
2149 should be updated to reflect the additional content. When this is done, the *namespace* and
2150 *schema* information in the header should be updated to reflect the URI for the extended
2151 *namespace* and *schema*.

2152 D Extensibility

2153 MTConnect is an extensible standard, which means that implementers **MAY** extend the
 2154 *data models* defined in the various sections of the MTConnect Standard to include infor-
 2155 mation required for a specific implementation. When these *data models* are encoded using
 2156 XML, the methods for extending these *data models* are defined by the rules established
 2157 for extending any XML schema (see the W3C website for more details on extending XML
 2158 data models).

2159 The following are typical extensions that **MAY** be considered in the MTConnect *data*
 2160 *models*:

- 2161 • Additional `type` and `subtype` values for *DataItems*.
- 2162 • Additional *structural elements* as containers.
- 2163 • Additional `Composition` elements.
- 2164 • New `Asset` types that are sub-typed from the abstract `Asset` type.
- 2165 • *child elements* that may be added to specific XML elements contained within the
 2166 *MTConnect Information Models*. These extended elements **MUST** be identified in
 2167 a separate *namespace*.

2168 When extending an MTConnect *data model*, there are some basic rules restricting changes
 2169 to the MTConnect *data models*.

2170 When extending an MTConnect *data model*, an implementer:

- 2171 • **MUST NOT** add new value for category for *DataItems*,
- 2172 • **MUST NOT** add new *root elements*,
- 2173 • **SHOULD NOT** add new *top level Components*, and
- 2174 • **MUST NOT** add any new attributes or include any sub-elements to `Composi-`
 2175 `tion`.

2176 Note: Throughout the documents additional information is provided where
 2177 extensibility may be acceptable or unacceptable to maintain compliance with
 2178 the MTConnect Standard.

2179 When a *schema* representing a *data model* is extended, the *schema* and *namespace* dec-
 2180 laration at the beginning of the corresponding *response document* **MUST** be updated to
 2181 reflect the new *schema* and *namespace* so that a client software application can properly
 2182 validate the *response document*.

2183 An XML example of a *schema* and *namespace* declaration, including an extended *schema*
 2184 and *namespace*, is shown in *Example 8*:

Example 8: Example of extended schema and namespace in declaration

```
2185 1 <?xml version="1.0" encoding="UTF-8"?>
2186 2 <MTConnectDevices
2187 3   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
2188 4   xmlns="urn:mtconnect.org:MTConnectDevices:1.3"
2189 5   xmlns:m="urn:mtconnect.org:MTConnectDevices:1.3"
2190 6   xmlns:x="urn:MyLocation:MyFile:MyVersion"
2191 7   xsi:schemaLocation="urn:MyLocation:MyFile:MyVersion
2192 8   /schemas/MyFileName.xsd" />
```

2193 In this example:

- 2194 • `xmlns:x` is added in Line 6 to identify the *XML Schema* instance for the extended
 2195 *schema*. *element names* identified with an “x” prefix are associated with this specific
 2196 *XML Schema* instance.

2197 Note: The “x” prefix **MAY** be replaced with any prefix that the implementer
 2198 chooses for identifying the extended *schema* and *namespace*.

- 2199 • `xsi:schemaLocation` is modified in Line 7 to associate the *namespace* URN
 2200 with the URL specifying the location of *schema* file.

- 2201 • `MyLocation`, `MyFile`, `MyVersion`, and `MyFileName` in Lines 6 and 7 **MUST**
 2202 be replaced by the actual name, version, and location of the extended *schema*.

2203 When an extended *schema* is implemented, each *structural element*, *DataItem*, and asset
 2204 defined in the extended *schema* **MUST** be identified in each respective *response document*
 2205 by adding a prefix to the XML *element name* associated with that *structural element*,
 2206 *DataItem*, or asset. The prefix identifies the *schema* and *namespace* where that XML
 2207 Element is defined.