MTConnect® Standard

Part 3.0 – Observation Information Model

Version 2.2.0

Prepared for: MTConnect Institute
Prepared from: `MTConnectSysMLModel.xml`
Prepared on: August 8, 2023

August 8, 2023

MTConnect Specification and Materials

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

Table of Contents

Table of Figures

List of Tables

1 Purpose of This Document

This document, *MTConnect Standard: Part 3.0 - Observation Information Model* of the MTConnect Standard, establishes the rules and terminology that describes the information returned by an *MTConnect Agent* from a piece of equipment. The term(Observation Information Model also defines, in *MTConnect Standard: Part 3.0 - Observation Information Model*, the structure for the *response documents* that are returned from an *agent* in response to a *sample request* or *current request*.

MTConnect Standard: Part 3.0 - Observation Information Model is not a stand-alone document. This document is used in conjunction with *MTConnect Standard Part 1.0 - Fundamentals* which defines the fundamentals of the operation of the MTConnect Standard and *MTConnect Standard: Part 2.0 - Device Information Model* that defines the semantic model representing the information that may be returned from a piece of equipment.

> Note: *MTConnect Standard: Part 5.0 - Interface Interaction Model* provides details on extensions to the *Observation Information Model* required to describe the interactions between pieces of equipment.

In the MTConnect Standard, equipment represents any tangible property that is used in the operation of a manufacturing facility. Examples of equipment are machine tools, ovens, sensor units, workstations, software applications, and bar feeders.

2 Terminology and Conventions

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

2.1 General Terms

adapter

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

agent

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

alarm limit

limit used to trigger warning or alarm indicators.

application

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

archetype

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

asset buffer

buffer for *Assets*.

attachment

connection by which one thing is associated with another.

buffer

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

cartesian coordinate system

3D orthogonal coordinate system [(]ISO/IEC 19794-5:2011en).

characteristic

control placed on an element of a *feature* such as its size, location, or form, which may be a specification limit, a nominal with tolerance, or some other numerical or non-numerical control. *Ref QIF 3.0 3.4.29. Ref AS9102-B*.

client

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

Note: Examples include software applications or a function that implements the *request* portion of an *interface interaction model*.

combined standard uncertainty

standard uncertainty of the result of a measurement when that result is obtained from the values of a number of other quantities, equal to the positive square root of a sum of terms, the terms being the variances or covariances of these other quantities weighted according to how the measurement result varies with changes in these quantities. *Ref JCGM 100:2008 2.3.4*

controlled vocabulary

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

data dictionary

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

data model

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

data set

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

data source

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

deprecated

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

deprecation warning

indication that specific content in an *MTConnect Document* may be changed to *deprecated* in a future release of the standard.

document

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

electric current

rate of flow of electric charge.

element

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

extensible

ability for an implementer to extend *MTConnect Information Model* by adding content not currently addressed in the MTConnect Standard.

feature

topological entity(ies) or design requirements related to a geometric model. *Ref QIF 3.0-3.4.59*

force

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

heartbeat

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

higher level

nested element that is above a lower level element.

implementation

specific instantiation of the MTConnect Standard.

information model

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

instance

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

interaction model

model that defines how information is exchanged across an *interface* to enable interactions between independent systems.

interface

means by which communication is achieved between independent systems.

key

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

key-value pair

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

location

place or named space associated with an object or that can be occupied by an object.

lower camel case

first word is lowercase and the remaining words are capitalized and all spaces between words are removed.

lower level

nested element that is below a higher level element.

lower limit

lower conformance boundary for a variable.

lower warning

lower boundary indicating increased concern and supervision may be required.

major

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

maximum

numeric upper constraint.

message

communication in writing, in speech, or by signals.

metadata

data that provides information about other data.

minimum

numeric lower constraint.

minor

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

nominal

ideal or desired value for a variable.

organize

act of containing and owning one or more elements.

organizer

entity that *organizes* one or more elements.

parameter

variable that must be given a value during the execution of a program or a communications command.

part

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

pascal case

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

persistence

method for retaining or restoring information.

position

location that is represented by a point in space relative to a reference.

probe

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

profile

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

requester

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

reset

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

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

responder

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

response document

electronic *document* published by an *MTConnect Agent* in response to a *probe request*, *current request*, *sample request* or *asset request*.

revision

supplemental identifier representing only organizational or editorial changes to a *minor* version document with no changes in the functionality described in that document.

schema

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

semantic data model

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

sensing element

mechanism that provides a signal or measured value.

sequence number

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

specification limit

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

spindle

mechanism that provides rotational capabilities to a piece of equipment.

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

standard

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

standard uncertainty

uncertainty of the result of a measurement expressed as a standard deviation. *Ref JCGM 100:2008 2.3.1*

stereotype

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

subtype

secondary or subordinate type of categorization or classification of information.

table

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

table cell

subdivision of a *table entry* representing a singular value.

table entry

subdivision of a *table* containing a set of *key-value pairs* representing *table cells*.

top level

element that represents the most significant physical or logical functions of a piece of equipment.

type

classification or categorization of information.

uncertainty

uncertainty (of measurement) parameter, associated with the result of a measurement, that characterizes the dispersion of the values that could reasonably be attributed to the measurand. *Ref JCGM 100:2008 2.2.3*

Note: Use of the term uncertainty refers to uncertainty of measurement.

upper limit

upper conformance boundary for a variable.

upper warning

upper boundary indicating increased concern and supervision may be required.

version

unique identifier of the administered item. *Ref ISO/IEC 11179-:2015*

2.2 Information Model Terms

Asset Information Model

information model that provides semantic models for *Assets*.

Device Information Model

information model that describes the physical and logical configuration for a piece of equipment and the data that may be reported by that equipment.

Error Information Model

information model that describes the *response document* returned by an *agent* when it encounters an error while interpreting a *request* for information from a *client* or when an *agent* experiences an error while publishing the *response* to a *request* for information.

MTConnect Information Model

information model that defines the semantics of the MTConnect Standard.

Observation Information Model

information model that describes the *streaming data* reported by a piece of equipment.

2.3 Protocol Terms

asset request

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

current request

request to an *agent* to produce an *MTConnectStreams Response Document* containing the *Observation Information Model* for a snapshot of the latest observations at the moment of the *request* or at a given *sequence number*.

data streaming

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

MTConnect Request

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

MTConnect Response Document

response document published by an *MTConnect Agent*.

MTConnectAssets Response Document

response document published by an *MTConnect Agent* in response to an *asset request*.

MTConnectDevices Response Document

response document published by an *MTConnect Agent* in response to a *probe request*.

MTConnectErrors Response Document

response document published by an *MTConnect Agent* whenever it encounters an error while interpreting an *MTConnect Request*.

MTConnectStreams Response Document

response document published by an *MTConnect Agent* in response to a *current request* or a *sample request*.

probe request

request to an *agent* to produce an *MTConnectDevices Response Document* containing the *Device Information Model*.

protocol

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

publish

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

publish and subscribe

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

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

request

communications method where a *client* transmits a message to an *agent*. That message instructs the *agent* to respond with specific information.

request and response

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

response

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

sample request

request to an *agent* to produce an *MTConnectStreams Response Document* containing the *Observation Information Model* for a set of timestamped observations made by *Components*.

streaming data

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

subscribe

receiving messages in a *publish and subscribe* pattern.

transport protocol

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

2.4 HTTP Terms

HTTP Body

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

HTTP Error Message

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

HTTP Header

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

HTTP Header Field

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

HTTP Message

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

> Note: In MTConnect Standard, it describes the information that is exchanged between an *agent* and a *client*.

HTTP Messaging

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

HTTP Method

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

HTTP Query

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

HTTP Request

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

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

HTTP Request Line

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

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

HTTP Request Method

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

HTTP Request URI

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

HTTP Response

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

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

HTTP Server

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

HTTP Status Code

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

HTTP Version

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

2.5 XML Terms

abstract element

element that defines a set of common characteristics that are shared by a group of elements. An abstract entity cannot appear in a document. In a specific implementation, an abstract entity is replaced by a derived element that is itself not an abstract entity. The characteristics for the derived element are inherited from the abstract entity.

attribute

additional information or property for an *element*.

child element

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

document body

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

document header

portion of the content of an *MTConnect Response Document* that provides information from an *agent* defining version information, storage capacity, protocol, and other information associated with the management of the data stored in or retrieved from the *agent*.

element name

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

namespace

organizes information into logical groups.

parent element

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

root element

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

structural element

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

XML Document

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

XML Schema

schema defining a specific document encoded in XML.

2.6 MTConnect Terms

Asset

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

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

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

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

Component

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

Composition

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

Configuration

configuration for a *Component*

DataItem

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

Device

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

MTConnect Agent

agent for the *MTConnect Information Model*.

MTConnect Document

document that represents a Part(s) of the MTConnect Standard.

MTConnect Event

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

MTConnect Interface

interaction model for interoperability between pieces of equipment.

Observation

observation that provides telemetry data for a *DataItem*.

2.7 Acronyms

2D

two-dimensional

3D

three-dimensional

AI

artificial intelligence

ALM

application lifecycle management

AMT

The Association for Manufacturing Technology

ANSI

American National Standards Institute

AP

Application Protocol

API

application programming interface

ASME

American Society of Mechanical Engineers

ASTM

American Society for Testing and Materials

AWS

American Welding Society

BDD

block definition diagram

BOM

bill of materials

BST

Board on Standardization and Testing

C&R

cause and remedy

CA

certificate authority

CAD

computer-aided design

CAE

computer-aided engineering

CAI

computer-aided inspection

CAM

computer-aided manufacturing

CAx

computer-aided technologies

CDATA

Character Data

CFD

computational fluid dynamics

CM

configuration management

CMS

coordinate-measurement system

CNC

Computer Numerical Controller

CNRI

Corporation for National Research Initiatives

CPM

Core Product Model

CPM2

Revised Core Product Model

CPSC

Consumer Product Safety Commission

cUAV

configurable unmanned aerial vehicle

DARPA

Defense Advanced Research Projects Agency

DER

designated-engineering representative

DFM

design for manufacturing

DLA

Defense Logistics Agency

DMC

digital manufacturing certificate

DMSC

Dimensional Metrology Standards Consortium

DNS

Domain Name System

DoD

U.S. Department of Defense

DOI

Distributed Object Identifier

DRM

digital rights management

ECR

engineering change request

ERP

enterprise resource planning

FAA

Federal Aviation Administration

FAIR

first article inspection reporting

FDA

Food and Drug Administration

FEA

finite-element analysis

GD&T

geometric dimensions and tolerances

GID

: global identifier

HMI

: Human Machine Interface

HTML

: Hypertext Markup Language

HTTP

: Hypertext Transfer Protocol

HTTPS

: Hypertext Transfer Protocol over Secure Sockets Layer

I/O

: in-out

ID

: identifier

IEEE

: Institute of Electrical and Electronics Engineers

IIoT

: industrial internet of things

INCOSE

: International Council on Systems Engineering

IP

: intellectual property

ISO

: International Standards Organization

ISS

: International Space Station

ISV

: Independent Software Vendor

IT

information technology

ITU-T

Telecommunication Standardization Sector of the International Telecommunication Union

JSON

JavaScript Object Notation

JT

Jupiter Tesselation

LHS

Lifecycle Handler System

LIFT

Lifecycle Information Framework and Technology

LOI

Lifecycle Object Identifier

MAC

media access control

MADE

Manufacturing Automation and Design Engineering

MBD

model-based definition

MBE

Model-Based Enterprise

MBI

model-based inspection

MBM

model-based manufacturing

MBSD

model-based standards development

MBSE

model-based systems engineering

MEDALS

Military Engineering Data Asset Locator System

MES

manufacturing execution system

MOI

manufacturing object identifier

MOM

Message Orienged Middleware

MQTT

Message Queuing Telemetry Transport

MTC

Manufacturing Technology Centre

NASA

National Aeronautics and Space Administration

NC

numerical control

NIST

National Institute of Standards and Technology

NMTOKEN

Name Token

NNMI

National Network of Manufacturing Innovation

NSF

National Science Foundation

NTSC

National Transportation Safety Board

OASIS

Organization for the Advancement of Structured Information Standards

ODI

Open Data Institute

OEM

original equipment manufacturer

OOI

Ocean Observatories Initiative

OPC

OLE for Process Control

OSLC

Open Services for Lifecycle Collaboration

OSTP

Office of Science and Technology Policy

OT

operational technology

OWL

Ontology Web Language

PDF

Portable Document Format

PDM

product-data management

PDQ

product-data quality

PHM

prognosis and health monitoring

PI

principal investigator

PLC

Programmable Logic Controller

PLCS

Product Life Cycle Support

PLM

product lifecycle management

PLOT

product lifecycle of trust

PMI

product and manufacturing information

PMS

Production Management System

PRC

Product Representation Compact

PSI

Physical Science Informatics

PTAB

Primary Trustworthy Digital Repository Authorization Body Ltd.

QIF

Quality Information Framework

QMS

quality management system

QName

Qualified Name

RDF

Resource Description Framework

REST

Representational State Transfer

RII

receiving and incoming inspection

S/MIME

Secure/Multipurpose Internet Mail Extensions

SaaS

software-as-a-service

SAML

Security Assertion Markup Language

SC

Standards Committee

SCADA

Supervisory Control And Data Acquisition

SDO

Standards Development Organization

SFTP

Secure File Transfer Protocol

SKOS

Simple Knowledge Organization System

SLH

system lifecycle handler

SLR

systematic literature review

SME

small-to-medium enterprise

SMOPAC

Smart Manufacturing Operations Planning and Control

SMS Test Bed

Smart Manufacturing Systems Test Bed

SOA

service-oriented architecture

SPMM

semantic-based product metamodel

SSL

Secure Sockets Layer

STEP

Standard for the Exchange of Product Model Data

STEP AP242

Standard for the Exchange of Product Model Data Application Protocol 242

STL

Stereolithography

SysML

Systems Modeling Language

TCP/IP

Transmission Control Protocol/Internet Protocol

TDP

technical data package

TLS

Transport Layer Security

TSM

Total System Model

UA

Unified Architecture

UAL

Unified Architecture Language

UML
: Unified Modeling Language

URI
: Uniform Resource Identifier

URL
: Uniform Resource Locator

URN
: Uniform Resource Name

UTC
: Coordinated Universal Time

UUID
: Universally Unique Identifier

V&V
: verification and validation

W3C
: World Wide Web Consortium

WSN
: Wirth Syntax Notation

WWW
: World Wide Web

X.509-PKI
: Public Key Infrastructure

X.509-PMI
: Privilege Management Infrastructure

XML
: Extensible Markup Language

XPath
: XML Path Language

XSD
: XML Schema Definitions

2.8 MTConnect References

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

[MTConnect Part 2.0] *MTConnect Standard: Part 2.0 - Device Information Model*. Version 2.0.

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

[MTConnect Part 4.0] *MTConnect Standard: Part 4.0 - Asset Information Model*. Version 2.0.

[MTConnect Part 5.0] *MTConnect Standard: Part 5.0 - Interface Interaction Model*. Version 2.0.

3 Observation Information Model

The *Observation Information Model* provides a representation of the data reported by a piece of equipment used for a manufacturing process, or used for any other purpose. Additional descriptive information associated with the reported data is defined by the MTConnectDevices entity, which is described in *MTConnect Standard: Part 2.0 - Device Information Model*.

Information defined in the *Observation Information Model* allows a software application to (1) determine the *Observations* for *DataItems* returned from a piece of equipment and (2) interpret the data associated with those *Observations* with the same meaning, value, and context that it had at its original source. To do this, the software application issues one of two HTTP requests to an *agent* associated with a piece of equipment. They are:

- sample: Returns a designated number of time stamped *Observations* from an *agent* associated with a piece of equipment; subject to any HTTP filtering associated with the request. See Agent in *MTConnect Standard Part 1.0 - Fundamentals* for details on the sample HTTP request.
- current: Returns a snapshot of either the most recent values or the values at a given sequence number for all *Observations* associated with a piece of equipment from an *agent*; subject to any HTTP filtering associated with the request. See Agent in *MTConnect Standard Part 1.0 - Fundamentals* for details on the current HTTP request.

An *agent* responds to either the sample or current HTTP request with an *MTConnectStreams Response Document*. This document contains information describing *Observations* reported by an *agent* associated with a piece of equipment. A client software application may correlate the information provided in the *MTConnectStreams Response Document* with the physical and logical structure for that piece of equipment defined in the MTConnectDevices entity to form a clear and unambiguous understanding of the information provided. (See details on the structure for a piece of equipment described in *MTConnect Standard: Part 2.0 - Device Information Model*).

Streams for an MTConnectStreams entity contains a DeviceStream entity for each piece of equipment represented in the document. Each DeviceStream is comprised of two primary types of entities – *Components* and *Observations*. The contents of the DeviceStream container are described in detail in this document, *MTConnect Standard: Part 3.0 - Observation Information Model* of the MTConnect Standard.

Components are defined for both the MTConnectDevices and the MTConnectStreams

entities. These *Components* are used to provide a logical organization of the information provided in each entity.

- For an `MTConnectDevices` entity: *Components* organize information that represents the physical and logical parts and sub-parts of a piece of equipment. (See `Component` in *MTConnect Standard: Part 2.0 - Device Information Model* for more details on *Components* used in the `MTConnectDevices` entity).
- For an `MTConnectStreams` entity: *Components* provide the structure to organize the data returned from a piece of equipment and establishes the proper context for that data. The *Components* specifically defined for `MTConnectStreams` are `DeviceStream` (see *Section 3.1 - DeviceStream*) and `ComponentStream` (see *Section 3.2 - ComponentStream*).

`DeviceStream` and `ComponentStream` entities have a direct correlation to each of the *Component* defined in the `MTConnectDevices` entity.

Within each `ComponentStream` entity in the `MTConnectStreams` entity, *Observations* are modeled as `Observation` entities. The three types of `Observation` entity are `Sample`, `Event`, and `Condition`. (See *Section 5 - Observation Types* for more information on these entities.)

3.1 DeviceStream

organizes data reported from a `Device`.

`DeviceStream` **MUST** be provided for each `Device` reporting data in an *MTConnect-Streams Response Document*.

If the response to the request for data from an *agent* does not contain any data for a specific `Device`, an empty `DeviceStream` entity **MAY** be created to indicate that the `Device` exists, but there was no data available.

3.1.1 Value Properties of DeviceStream

Table 1 lists the Value Properties of `DeviceStream`.

Value Property name	Value Property type	Multiplicity
name	ID	1
uuid	ID	1

Table 1: Value Properties of DeviceStream

Descriptions for Value Properties of DeviceStream:

- name

  name of the Device.

  The value reported for name,DeviceStream **MUST** be the same as the value defined for the name,Device attribute of the same Device in the *MTConnect-Devices Response Document*.

- uuid

  uuid of the Device.

  The value reported for uuid,DeviceStream **MUST** be the same as the value defined for the uuid,Device attribute of the same Device in the *MTConnect-Devices Response Document*.

3.1.2 Part Properties of DeviceStream

Table 2 lists the Part Properties of DeviceStream.

Part Property name	Multiplicity
ComponentStream	1..*

Table 2: Part Properties of DeviceStream

Descriptions for Part Properties of DeviceStream:

- ComponentStream

  organizes the data associated with each Component entity defined for a Device in the associated *MTConnectDevices Response Document*.

  See *Section 3.2 - ComponentStream*.

3.2 ComponentStream

organizes the data associated with each Component entity defined for a Device in the associated *MTConnectDevices Response Document*.

At least one of Sample, Event, or Condition **MUST** be organized by a ComponentStream entity.

3.2.1 Value Properties of ComponentStream

Table 3 lists the Value Properties of ComponentStream.

Value Property name	Value Property type	Multiplicity
component	string	1
componentId	ID	1
name	string	0..1
nativeName	string	0..1
uuid	ID	0..1

Table 3: Value Properties of ComponentStream

Descriptions for Value Properties of ComponentStream:

- component

  identifies the Component type associated with the ComponentStream.

  Examples of component are Device, Controller, Linear and Loader.

- componentId

  identifier of the Component as defined by the id attribute in the *MTConnectDevices Response Document*.

- name

  name of the Component associated with the ComponentStream.

- nativeName

  common name of the Component associated with the ComponentStream.

- uuid

  uuid of the Component associated with the ComponentStream.

3.2.2 Reference Properties of ComponentStream

Table 4 lists the Reference Properties of `ComponentStream`.

Reference Property name	Multiplicity
`Event` (organized by `Events`)	0..*
`Sample` (organized by `Samples`)	0..*
`Condition` (organized by `Conditions`)	0..*

Table 4: Reference Properties of ComponentStream

Descriptions for Reference Properties of `ComponentStream`:

- `Event`

  `Observation` that is a discrete piece of information from a piece of equipment.

  `Events` groups one or more `Event` entities. See *Section 3.5 - Event*.

- `Sample`

  `Observation` that is continuously changing or analog data value.

  `Samples` groups one or more `Sample` entities. See *Section 3.4 - Sample*.

- `Condition`

  `Observation` that provides the condition of a piece of equipment or a *Component*.

  `Conditions` groups one or more `Condition` entities. See *Section 3.6 - Condition*.

  Note: In the XML representation, `Conditions` **MUST** appear as `Condition` element in the *MTConnectStreams Response Document*.

3.3 Observation

abstract entity that provides telemetry data for a `DataItem` at a point in time.

Note: See *Section B.1 - Observations Schema Diagrams* for XML schema. The XML schema also shows differences in XML representation of certain MTConnect entities.

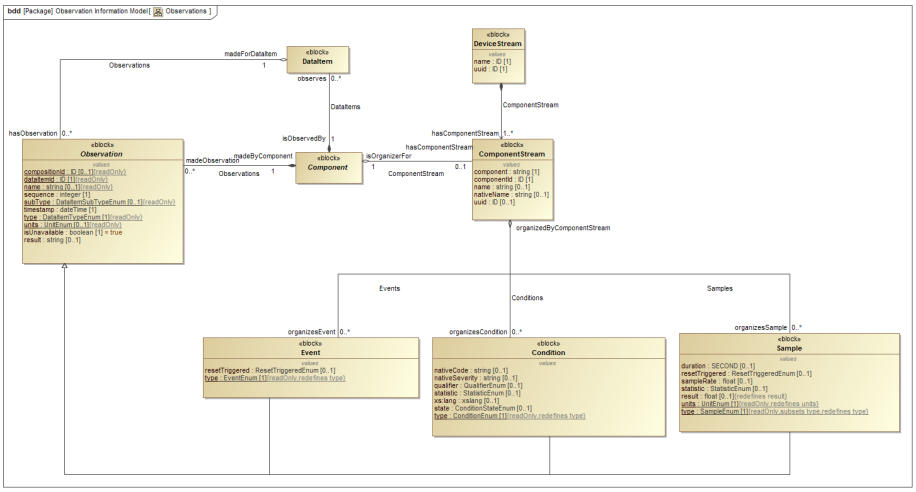

Figure 1: Observations

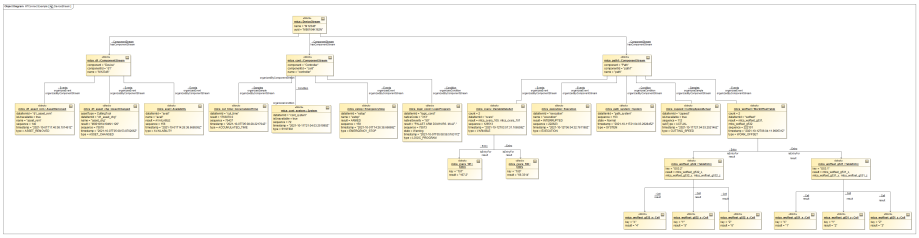

Figure 2: DeviceStream Example

Figure 2 shows a complete example of `DeviceStream` for the `Device` shown in *Figure 2: Component Example* in *MTConnect Standard: Part 2.0 - Device Information Model*.

Note: See *Example 1* for the XML representation of the same example.

This section provides semantic information for the `Observation` model.

Note: See *Section B.1 - Observations Schema Diagrams* for XML schema of `Observation` types.

3.3.0.1 Observations made for DataItem

`Component` observes `DataItem` entities and creates `Observation` entities for the `DataItem` entities. See Figure 1.

`Observation` entities made by a `Component` are organized by a `ComponentStream` which is specifically created for that `Component`.

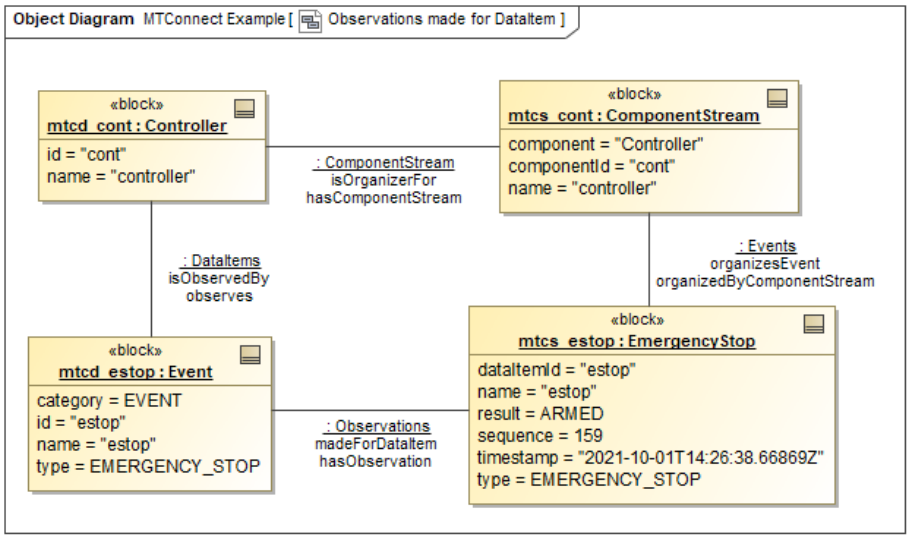

Figure 3: Observations made for DataItem Example

Note: See *Section C.2 - Observations made for DataItem Example* for how XML representation of the same example is separated into *MTConnectDevices Response Document* and *MTConnectStreams Response Document*.

Figure 3 is a subset of Figure 2. It shows an example of the association between a `DataItem` `Event` type (`EMERGENCY_STOP`) and an `Observation` `Event` type (`EmergencyStop`). See *Section 3.3.0.2 - Naming Convention for Observation types*.

Figure 3 also shows example of the association between a `Component` type (`Controller`) and related `ComponentStream`.

3.3.0.2 Naming Convention for Observation types

The name of an `Observation` type **MUST** derive from the `DataItem` property `type` converted to Pascal-Case by removing underscores (_) and capitalizing each word. The conversion **MUST NOT** apply to the following abbreviated words: `PH`, `AC`, `DC` and `URI`. `MTCONNECT` **MUST** be converted to `MTConnect`. See Figure 3 for an example.

The name of an `Observation` type reported in the *MTConnectStreams Response Document* is extended when the `representation` property is used to further describe that `DataItem` in the *MTConnectDevices Response Document*. See *Section 4 - Representations* for more details.

3.3.1 Value Properties of Observation

Table 5 lists the Value Properties of `Observation`.

Value Property name	Value Property type	Multiplicity
`compositionId`	`ID`	0..1
`dataItemId`	`ID`	1
`name`	`string`	0..1
`sequence`	`integer`	1
`subType`	`DataItemSubTypeEnum`	0..1
`timestamp`	`datetime`	1
`type`	`DataItemTypeEnum`	1
`units`	`UnitEnum`	0..1
`isUnavailable`	`boolean`	1

Table 5: Value Properties of Observation

Descriptions for Value Properties of `Observation`:

- `compositionId`

  identifier of the `Composition` entity defined in the *MTConnectDevices Response Document* associated with the data reported for the `Observation`.

- `dataItemId`

  unique identifier of the `DataItem` associated with this `Observation`.

  `dataItemId` **MUST** match the `id` attribute of the `DataItem` defined in the *MTConnectDevices Response Document*.

- `name`

  name of the `DataItem` associated with this `Observation`.

  `name` **MUST** match the `name` attribute of the `DataItem` defined in the *MTConnectDevices Response Document*.

- `sequence`

  number representing the sequential position of an occurrence of an observation in the data buffer of an *agent*.

  `sequence` **MUST** have a value represented as an unsigned 64-bit value from 1 to $2^{64} - 1$.

- `subType`

  subtype of the `DataItem` associated with this `Observation`.

  `subType` **MUST** match the `subType` attribute of the `DataItem` defined in the *MTConnectDevices Response Document*.

  The value of `subType` **MUST** be one of the `DataItemSubTypeEnum` enumeration.

- `timestamp`

  most accurate time available to a piece of equipment that represents the point in time that the data reported was measured.

- `type`

  type of the `DataItem` associated with this `Observation`.

  `type` **MUST** match the `type` attribute of the `DataItem` defined in the *MTConnectDevices Response Document*.

  The value of `type` **MUST** be one of the `DataItemTypeEnum` enumeration.

- `units`

  units of the `DataItem` associated with this `Observation`.

  `units` **MUST** match the `units` attribute of the `DataItem` defined in the *MTConnectDevices Response Document*.

  The value of `units` **MUST** be one of the `UnitEnum` enumeration.

- `isUnavailable`

  when `true`, `result` is indeterminate.

  > Note 1 to entry: In XML, when `isUnavailable` is `true`, the XML `CDATA` of the `Observation` **MUST** be `UNAVAILABLE`. “‘xml

  <Execution dataItemId=”…” …>UNAVAILABLE</Execution> “‘

  > Note 2 to entry: In JavaScript Object Notation (JSON), when `isUnavailable` is `true`, the JSON `value` of the `Observation` **MUST** be `UNAVAILABLE`. `json "Execution" : [ "dataItemId": "..." ..., "value": "UNAVAILABLE" ]`

- `result`

  observation of the `Observation` entity.

  The default value type for `result` is `string`.

Note 1 to entry: in XML the `Observation:result` is the `CDATA` of the Observation *element*.

~~~~xml <Execution dataItemId="..." ...>READY</Execution> ~~~~

Note 2 to entry: in JSON the `Observation:result` is the member `value` of the Observation object.

~~~~json "Execution" : ["dataItemId": "..." ..., "value": "READY"] ~~~~

3.4 Sample

`Observation` that is continuously changing or analog data value.

It provides the information and data reported from a piece of equipment for those `DataItem` entities defined with a `category,DataItem` property of `SAMPLE` in the *MTConnect-Devices Response Document*.

`Sample` **MUST** always be reported in `float`.

Figure 4 shows `Sample` type examples. It also shows an example for when the `result` is not available (`dataItemId=cspeed`).

Note: See *Example 4* for the XML representation of the same example.

The following *Section 3.4.1 - Value Properties of Sample* lists the additional and/or updated attributes for `Sample`.

The value of `Sample` **MUST** be `float`.

The `units` for `Sample` **MUST** always be specified.

3.4.1 Value Properties of Sample

Table 6 lists the Value Properties of `Sample`.

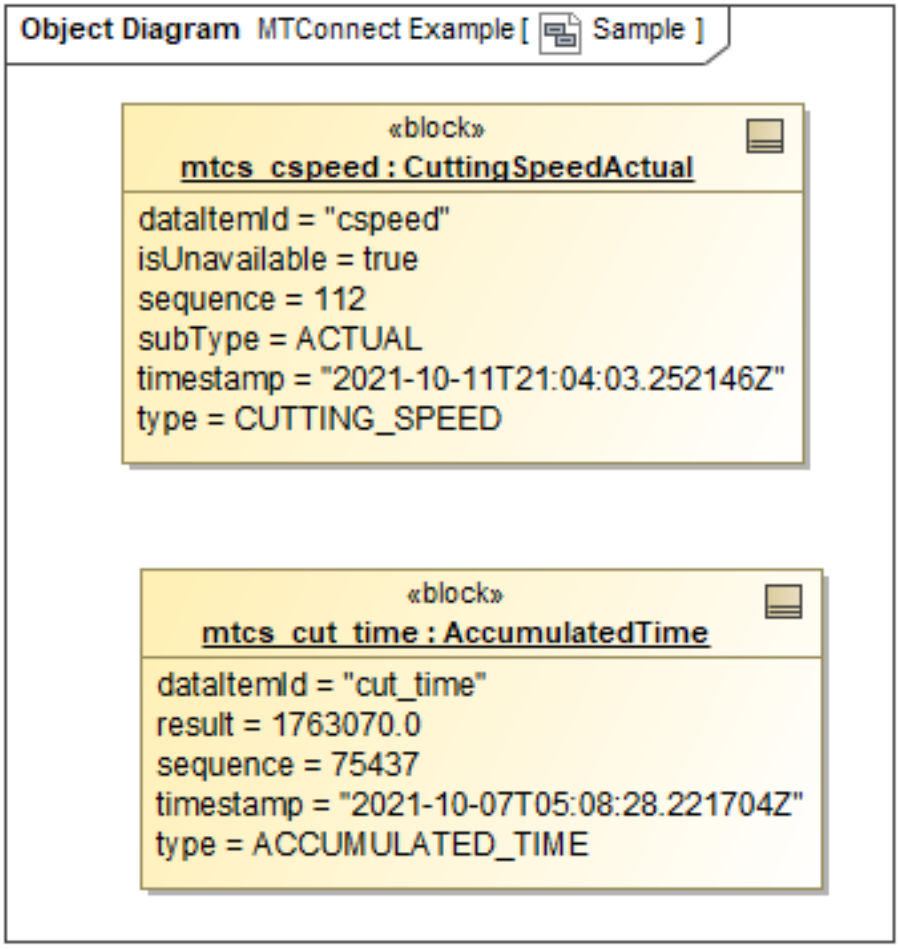

Figure 4: Sample Example

Value Property name	Value Property type	Multiplicity
duration	SECOND	0..1
resetTriggered	ResetTriggeredEnum	0..1
sampleRate	float	0..1
statistic	StatisticEnum	0..1

Table 6: Value Properties of Sample

Descriptions for Value Properties of `Sample`:

- `duration`

  time-period over which the data was collected.

  `duration` **MUST** be provided when the `statistic` attribute of the `DataItem` is defined in the *MTConnectDevices Response Document*.

- `resetTriggered`

  identifies when a reported value has been reset and what has caused that reset to occur for those `DataItem` entities that may be periodically reset to an initial value.

  `resetTriggered` **MUST** only be provided for the specific occurrence of a `DataItem` reported in the *MTConnectStreams Response Document* when the reset occurred.

  `ResetTriggeredEnum` Enumeration:

  - `ACTION_COMPLETE`
    `result` of the `Observation` that is measuring an action or operation was reset upon completion of that action or operation.
  - `ANNUAL`
    `result` of the `Observation` was reset at the end of a 12-month period.
  - `DAY`
    `result` of the `Observation` was reset at the end of a 24-hour period.
  - `MAINTENANCE`
    `result` of the `Observation` was reset upon completion of a maintenance event.
  - `MANUAL`
    `result` of the `Observation` was reset based on a physical reset action.
  - `MONTH`
    `result` of the `Observation` was reset at the end of a monthly period.
  - `POWER_ON`
    `result` of the `Observation` was reset when power was applied to the piece of equipment after a planned or unplanned interruption of power has occurred.
  - `SHIFT`
    `result` of the `Observation` was reset at the end of a work shift.
  - `WEEK`
    `result` of the `Observation` was reset at the end of a 7-day period.

- `sampleRate`

  rate at which successive samples of the value are recorded.

- `statistic`

  type of statistical calculation defined by the `statistic` attribute of the `DataItem` defined in the *MTConnectDevices Response Document*.

  The value of `statistic` **MUST** be one of the `StatisticEnum` enumeration.

3.5 Event

`Observation` that is a discrete piece of information from a piece of equipment.

It provides the information and data reported from a piece of equipment for those `DataItem` entities defined with a `category,DataItem` property of `EVENT` in the *MTConnectDevices Response Document*.

Figure 5 shows `Event` type examples. It also shows an example for when the `result` is not available (`dataItemId=d1_asset_rem`).

Note: See *Example 5* for the XML representation of the same example.

The following *Section 3.5.1 - Value Properties of Event* lists the additional and/or updated attributes for `Event`.

3.5.1 Value Properties of Event

Table 7 lists the Value Properties of `Event`.

Value Property name	Value Property type	Multiplicity
`resetTriggered`	`ResetTriggeredEnum`	0..1

Table 7: Value Properties of Event

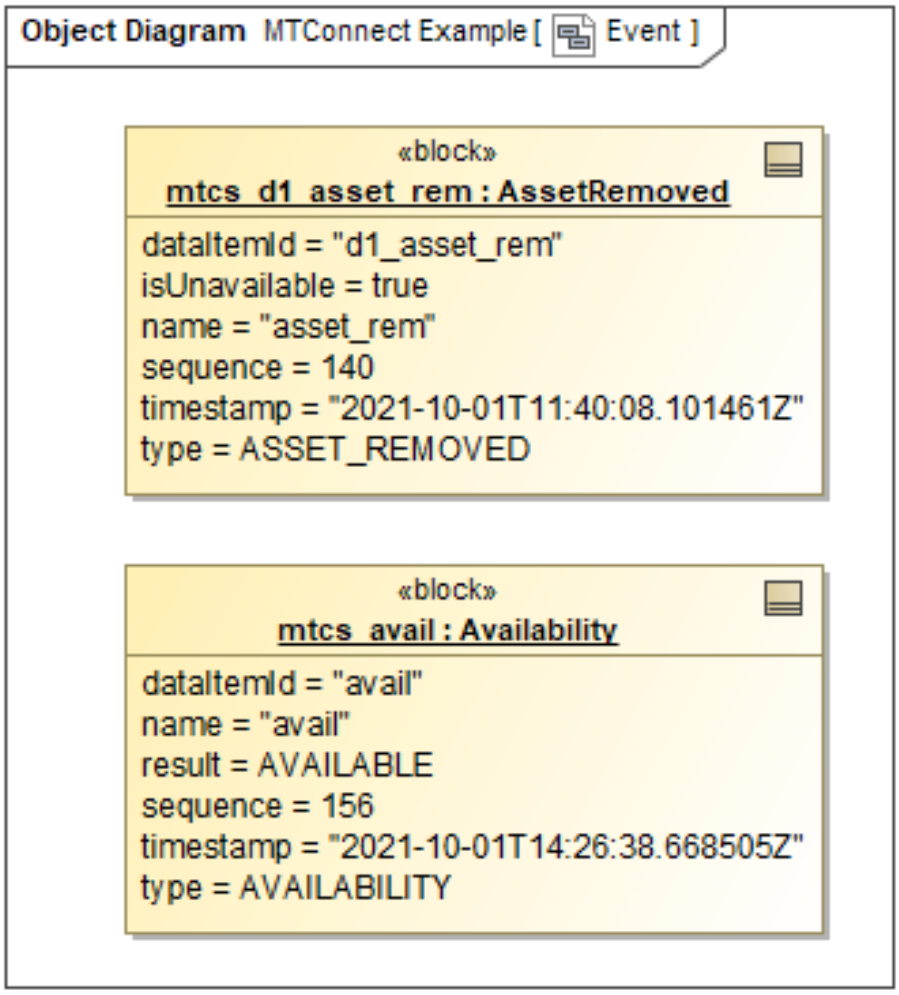

Figure 5: Event Example

Descriptions for Value Properties of `Event`:

- `resetTriggered`

  identifies when a reported value has been reset and what has caused that reset to occur for those `DataItem` entities that may be periodically reset to an initial value.

  `resetTriggered` **MUST** only be provided for the specific occurrence of a `DataItem` reported in the *MTConnectStreams Response Document* when the reset occurred.

  The value of `resetTriggered` **MUST** be one of the `ResetTriggeredEnum` enumeration.

3.6 Condition

Observation that provides the condition of a piece of equipment or a *Component*.

It provides the information and data reported from a piece of equipment for those DataItem entities defined with a category,DataItem property of CONDITION in the *MTConnectDevices Response Document*.

Figure 6 shows Condition type examples for various state: Normal (dataItemId = path_system) and Warning (dataItemId = logic_cond). It also shows an example for when the state is not available (dataItemId = cont_system).

Note: See *Example 6* for the XML representation of the same example.

The following *Section 3.6.1 - Value Properties of Condition* lists the additional and/or updated attributes for Condition.

3.6.1 Value Properties of Condition

Table 8 lists the Value Properties of Condition.

Value Property name	Value Property type	Multiplicity
nativeCode	string	0..1
nativeSeverity	string	0..1
qualifier	QualifierEnum	0..1
statistic	StatisticEnum	0..1
xs:lang	xslang	0..1
state	ConditionStateEnum	0..1

Table 8: Value Properties of Condition

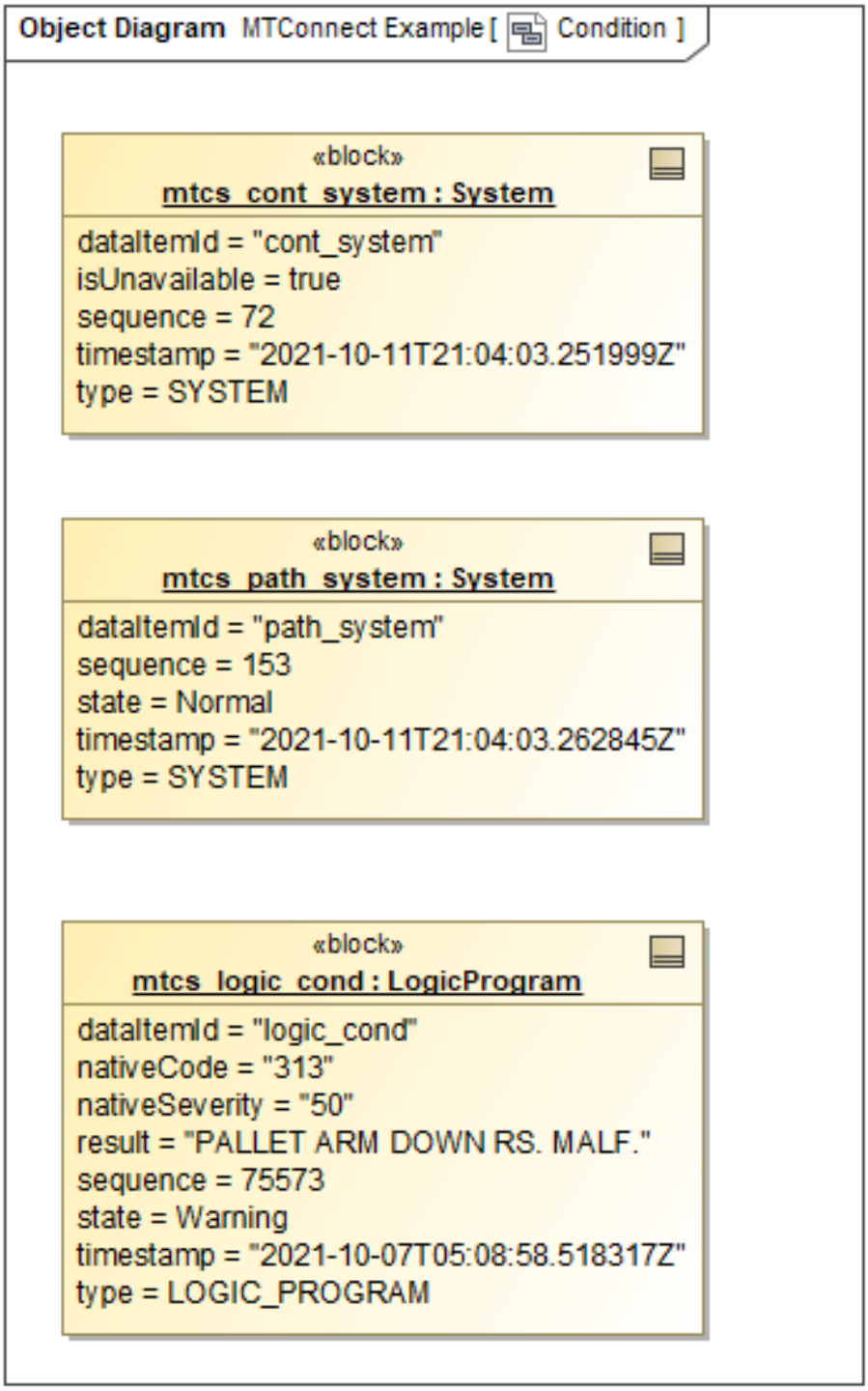

Figure 6: Condition Example

Descriptions for Value Properties of `Condition`:

- `nativeCode`

  native code (usually an alpha-numeric value) generated by the controller of a piece of equipment providing a reference identifier for a `Condition`.

  This is the same information an operator or maintenance personnel may see as a reference code designating a specific fault code provided by the piece of equipment.

- `nativeSeverity`

  severity information to a client software application if the piece of equipment designates a severity level to a fault.

- `qualifier`

  additional information regarding a condition state associated with the measured value of a process variable.

  `qualifier` defines whether the condition state represented indicates a measured value that is above or below an expected value of a process variable.

  `QualifierEnum` Enumeration:

  - `HIGH`

    measured value is greater than the expected value for a process variable.

  - `LOW`

    measured value is less than the expected value for a process variable.

- `statistic`

  `statistic` provides additional information describing the meaning of the `Condition` element.

  `statistic` **MUST** match the `statistic` attribute of the `DataItem` entity defined in the *MTConnectDevices Response Document*.

  The value of `statistic` **MUST** be one of the `StatisticEnum` enumeration.

- `xs:lang`

  specifies the language of the `result` returned for the `Condition`.

  See *Ref IETF RFC 4646* (http://www.ietf.org/rfc/rfc4646.txt).

- `state`

  condition state of the piece of equipment or `Component`.

  `ConditionStateEnum` Enumeration:

- `Fault`
  condition state that requires intervention to continue operation to function properly.
- `Normal`
  condition state that indicates operation within specified limits.
- `Warning`
  condition state that requires concern and supervision and may become hazardous if no action is taken.

4 Representations

This section provides semantic information for the Representation model.

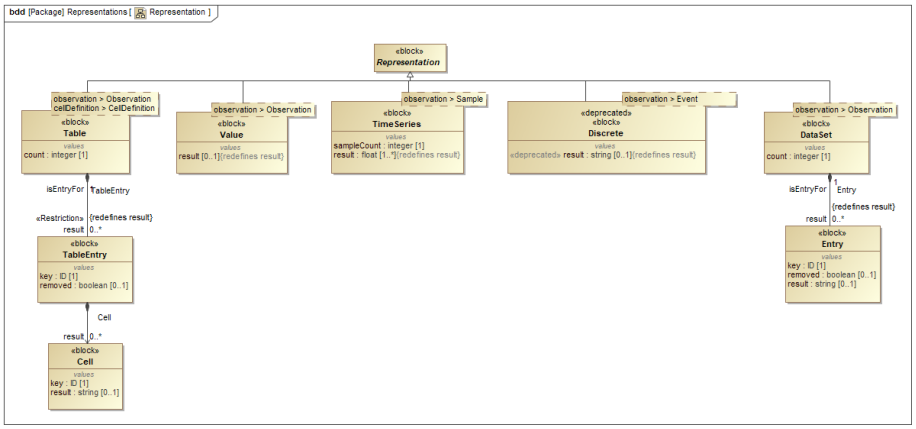

Figure 7: Representation

Note: See *Section B.2 - Representation Schema Diagrams* for XML schema.

4.1 Representation

specifies the format and structure of the result for an Observation.

The Representation type for an Observation is defined by the associated DataItem's property representation in the *MTConnectDevices Response Document*.

Value is the default Representation type for all Observation types.

The name of the Observation type is modified for all Representation types other than Value by appending the pascal case of the Representation type.

Example: The name for Sample Observation type Temperaturewith Representation type of TimeSeries becomes TemperatureTimeSeries.

4.2 Value

default Representation type for all Observation types where result of the Observation types is an MTConnect data type. See *Section 6.1 - DataTypes*.

4.3 TimeSeries

Representation for an Observation composed of a series of sampled data.

TimeSeries for an Observation is defined by the associated DataItem's property representation as TIME_SERIES.

DataItem with TIME_SERIES representation **MUST** have a category of SAMPLE.

Figure 8 shows the model for Temperature (Sample type) with a Representation type of TimeSeries.

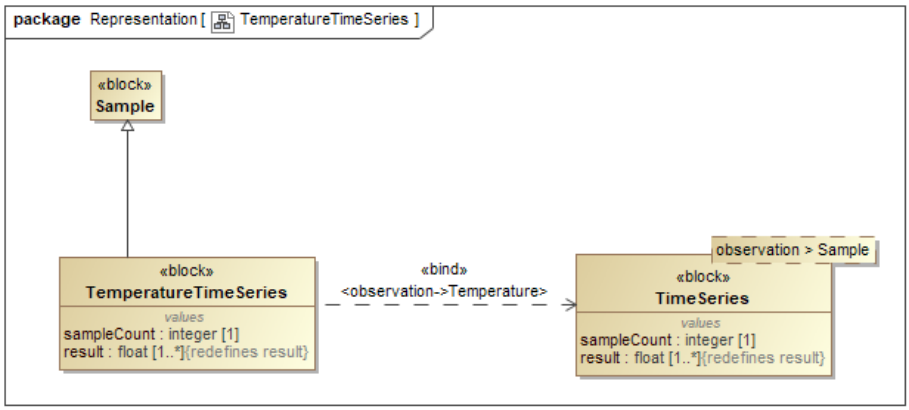

Figure 8: TemperatureTimeSeries

Note: See *Section B.2 - Representation Schema Diagrams* for XML schema.

TimeSeries **MUST** report multiple values at fixed intervals in a single Observation. At minimum, one of DataItem or Observation **MUST** specify the sampleRate in Hertz(values/second); fractional rates are permitted. When the Observation and the DataItem specify the sampleRate, the Observation sampleRate supersedes the DataItem.

The Observation **MUST** set the timestamp to the time the last value was observed. The duration **MAY** indicate the time interval from the first to the last value in the series.

Section 4.3.1 - Value Properties of TimeSeries defines additional attributes for an Observation with TimeSeries Representation type.

4.3.1 Value Properties of TimeSeries

Table 9 lists the Value Properties of `TimeSeries`.

Value Property name	Value Property type	Multiplicity
`sampleCount`	`integer`	1

Table 9: Value Properties of TimeSeries

Descriptions for Value Properties of `TimeSeries`:

- `sampleCount`
  number of values given for the `Observation`.

4.4 <<deprecated>>Discrete

DEPRECATED `Representation` for an `Observation` where each discrete occurrence of the data may have the same value as the previous occurrence of the data.

`Discrete` for an `Observation` is defined by the associated `DataItem`'s property `representation` as `DISCRETE`.

`DataItem` with `DISCRETE` `representation` **MUST** have a `category` of `EVENT`.

MTConnect Version 1.5 replaced `representation` `DISCRETE` with a `discrete` property for `DataItem`.

Each occurrence of the `Observation` **MAY** have the same value as the previous occurrence, and **MUST NOT** suppress duplicates.

Examples of `Discrete`: A `PartCount` reporting the completion of each part using a 1 to indicate completion of a single part, a `Message` that occurs each time a door opens.

4.5 DataSet

`Representation` for an `Observation` composed of value(s) represented as a set of *key-value pairs*.

`DataSet` for an `Observation` is defined by the associated `DataItem`'s property `representation` as `DATA_SET`.

DataItem with DATA_SET representation **MUST** have a category of SAMPLE or EVENT.

Figure 9 shows the model for Variable (Event type) with a Representation type of DataSet.

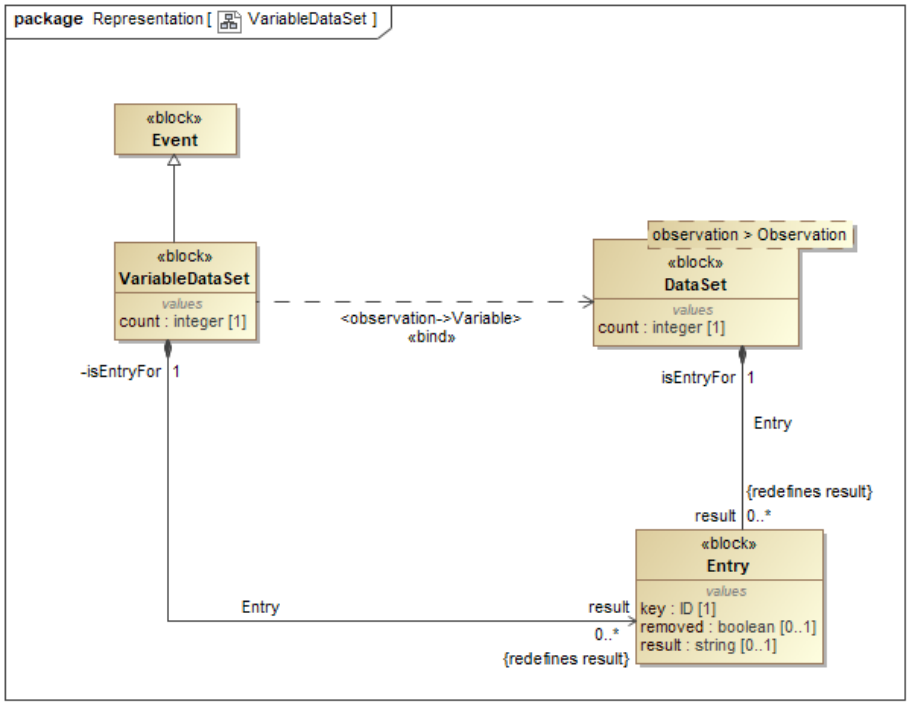

Figure 9: VariableDataSet

DataSet reports multiple values as a set of *key-value pair* where each *key* **MUST** be unique. The representation of the *key-value pair* is an Entry. The value of each Entry **MUST** have the same constraints and format as the Observation defined for the VALUE representation for the DataItem type (See Value).

The meaning of each Entry **MAY** be provided as the DataItem EntryDefinition.

Figure 10 shows Event Observation type Variable with a Representation type of DataSet.

Note: See *Example 7* for the XML representation of the same example.

4.5.0.1 Management of Data Set Observations

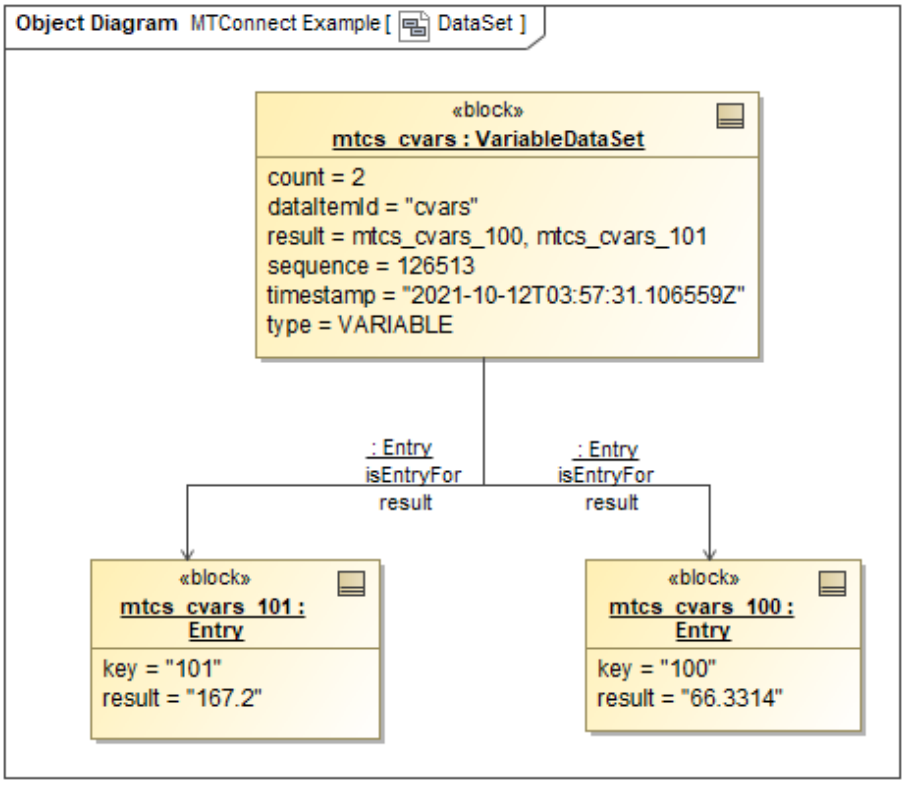

Figure 10: DataSet Example

An *agent* **MUST** maintain the current state of the `DataSet` as described in *MTConnect Standard Part 1.0 - Fundamentals*.

One or more *key-value pairs* **MAY** be added, removed, or changed in an `Observation`. An *agent* **MUST** publish the changes to one or more *key-value pairs* as a single `Observation`. An *agent* **MUST** indicate the removal of a *key-value pair* from a `DataSet` using the `removed` attribute equal `true`.

When the `DataItem` `discrete`,`DataItem` attribute is `false` or is not present, an *agent* in response to a *sample request* **MUST** only publish the changed *key-value pair* since the previous state of the `DataSet`.

When the `DataItem` `discrete`,`DataItem` attribute is `true`, an *agent*, in response to a *sample request*, **MUST** report all *key-value pairs* ignoring the state of the `DataSet`.

When an *agent* responds to a *current request*, the *response document* **MUST** include the full set of *key-value pairs*. If the *current request* includes an `at` query parameter, the *agent* **MUST** provide the set of *key-value pairs* at the *sequence number*.

When an Observation *reset* occurs, the DataSet **MUST** remove all *key-value pairs* making the set empty. The Observation **MAY** simultaneously populate the DataSet with new *key-value pairs*. The previous entries **MUST NOT** be included and **MUST NOT** have removed attribute equal true.

When the Observation is UNAVAILABLE the DataSet **MUST** remove all *key-value pairs* making the set empty.

4.5.1 Value Properties of DataSet

Table 10 lists the Value Properties of DataSet.

Value Property name	Value Property type	Multiplicity
count	integer	1

Table 10: Value Properties of DataSet

Descriptions for Value Properties of DataSet:

- count

  number of Entry elements for the Observation.

4.5.2 Part Properties of DataSet

Table 11 lists the Part Properties of DataSet.

Part Property name	Multiplicity
Entry	0..*

Table 11: Part Properties of DataSet

Descriptions for Part Properties of DataSet:

- Entry

  key-value pair published as part of a DataSet.

  See *Section 4.7 - Entry*.

4.6 Table

`Representation` for an `Observation` composed of two-dimensional sets of *key-value pairs* where the `Entry` represents rows containing sets of *key-value pairs* given by `Cell` entities.

`Table` for an `Observation` is defined by the associated `DataItem`'s property `representation` as `TABLE`.

`DataItem` with `TABLE` `representation` **MUST** have a `category` of `SAMPLE` or `EVENT`.

Figure 11 shows the model for `WorkOffset` (`Event` type) with a `Representation` type of `Table`.

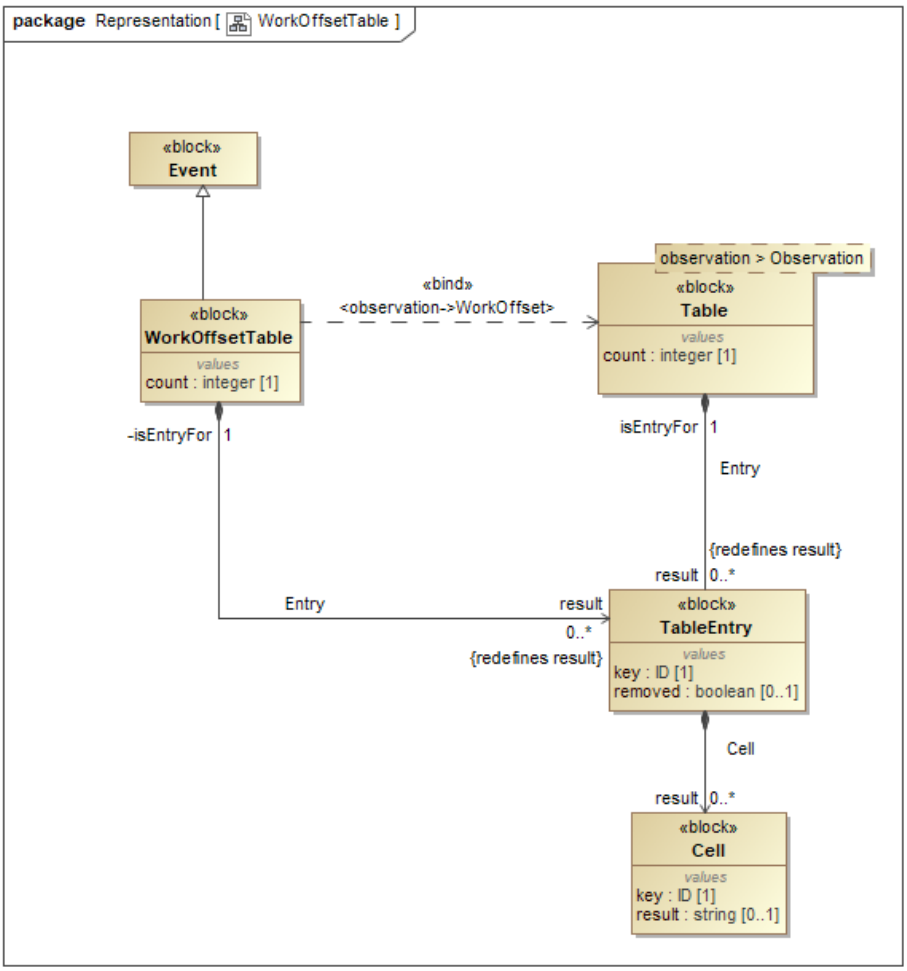

Figure 11: WorkOffsetTable

Note: See *Section B.2 - Representation Schema Diagrams* for XML schema.

`Table` has the same behavior as the `DataSet` for change tracking, clearing, and history. When an `Entry` changes, all `Cell` entities update at the same time; they are not tracked separately like `Entry`.

The meaning of each `Entry` and `Cell` **MAY** be provided as the `DataItem` `EntryDefinition` and `CellDefinition`.

The `Entry` `key` attribute **MUST** be the unique identity of the `Entry` within an `Observation`. The `Cell` `key` attribute **MUST** be the unique identity of the `Cell` within an `Entry`.

Figure 12 shows `Event` `Observation` type `WorkOffset` with a `Representation` type of `Table`.

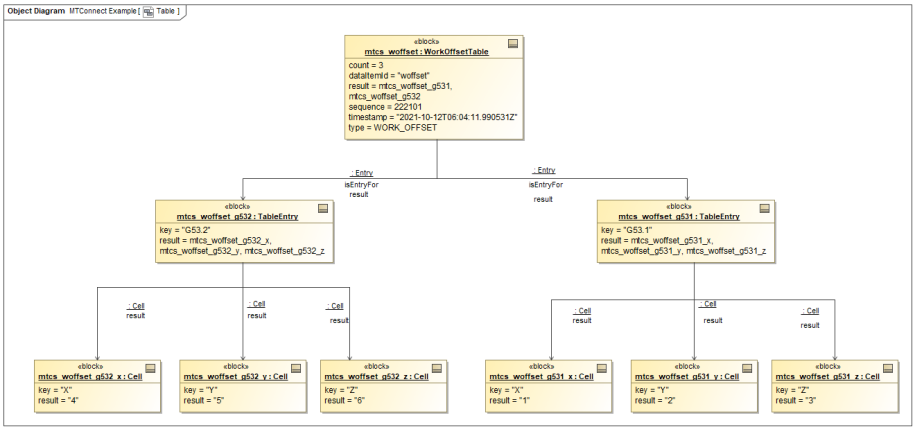

Figure 12: Table Example

Note: See *Example 8* for the XML representation of the same example.

4.6.1 Value Properties of Table

Table 12 lists the Value Properties of `Table`.

Value Property name	Value Property type	Multiplicity
`count`	`integer`	1

Table 12: Value Properties of Table

Descriptions for Value Properties of `Table`:

- `count`
  number of *key-value pairs* represented as `Entry` entities.

4.6.2 Part Properties of Table

Table 13 lists the Part Properties of `Table`.

Part Property name	Multiplicity
`TableEntry`	0..*

Table 13: Part Properties of Table

Descriptions for Part Properties of `Table`:

- `TableEntry`
  key-value pair published as part of a `Table`.

  Note: In the XML representation, `TableEntry` **MUST** appear as `Entry`.

  See *Section 4.8 - TableEntry*.

4.7 Entry

key-value pair published as part of a `DataSet`.

4.7.0.1 Constraints for Entry Values

The value of each `Entry` **MUST** have the same restrictions as the value of an observation with `representation` of `VALUE`.

An `Entry` **MAY** be further constrained by the `DataItem` definition (see *MTConnect Standard: Part 2.0 - Device Information Model*), for example a `VariableDataSet` having a string value **MAY** have a floating-point `Temperature` value. A restriction **MUST NOT** be broadened or removed, for example, the value "READY" **MUST NOT** occur with a `TemperatureDataSet` constrained to floating-point numbers.

The *MTConnect Standard: Part 2.0 - Device Information Model* `DataItem` `Definition` **MAY** provide the `type` and `units` of an `Entry` for a `key`.

4.7.1 Value Properties of Entry

Table 14 lists the Value Properties of `Entry`.

Value Property name	Value Property type	Multiplicity
`key`	`ID`	1
`removed`	`boolean`	0..1

Table 14: Value Properties of Entry

Descriptions for Value Properties of `Entry`:

- `key`
  unique identifier for each *key-value pair*.
- `removed`
  removal indicator of a *key-value pair*.

4.8 TableEntry

key-value pair published as part of a `Table`.

Note: In the XML representation, `TableEntry` **MUST** appear as `Entry`.

4.8.1 Value Properties of TableEntry

Table 15 lists the Value Properties of `TableEntry`.

Value Property name	Value Property type	Multiplicity
`key`	`ID`	1
`removed`	`boolean`	0..1

Table 15: Value Properties of TableEntry

Descriptions for Value Properties of `TableEntry`:

- `key`
  unique identifier for each *key-value pair*.

- `removed`

  removal indicator of a *key-value pair*.

4.8.2 Part Properties of TableEntry

Table 16 lists the Part Properties of `TableEntry`.

Part Property name	Multiplicity
`Cell`	0..*

Table 16: Part Properties of TableEntry

Descriptions for Part Properties of `TableEntry`:

- `Cell`

  key-value pair published as part of a `TableEntry`.

  See *Section 4.9 - Cell*.

4.9 Cell

key-value pair published as part of a `TableEntry`.

4.9.0.1 Constraints for Cell Values

The value of each `Cell` **MUST** have the same restrictions as the value of an observation with `representation` of `VALUE`.

An `Cell` **MAY** be further constrained by the `DataItem` definition (see *MTConnect Standard: Part 2.0 - Device Information Model*), for example a `VariableDataSet` having a string value **MAY** have a floating-point `Temperature` value. A restriction **MUST NOT** be broadened or removed, for example, the value `READY` **MUST NOT** occur with a `TemperatureDataSet` constrained limited to floating-point numbers.

The *MTConnect Standard: Part 2.0 - Device Information Model* `DataItem` `Definition` **MAY** provide the `type` and `units` of a `Cell` for a `key`.

4.9.1 Value Properties of Cell

Table 17 lists the Value Properties of `Cell`.

Value Property name	Value Property type	Multiplicity
`key`	`ID`	1

Table 17: Value Properties of Cell

Descriptions for Value Properties of `Cell`:

- `key`
  unique identifier for each *key-value pair*.

5 Observation Types

This section provides semantic information for the Observation types.

Observation entities are instantiated as Sample, Event or Condition entities based upon the category and type attributes defined for the corresponding DataItem.

See Figure 2 for examples on how the Observation types are organized within ComponentStream.

5.1 Condition Types

This section provides semantic information for Condition types.

Condition types are reported differently from other Observation types. They are reported based on the condition state for each Condition.

The type and subType (where applicable) properties for a Condition **MAY** be any of the type and subType attributes defined for SAMPLE category or EVENT category DataItem listed in the *Device Information Model*.

This section lists additional Condition types that have been defined to represent the health and fault status of *Components*.

5.1.1 Actuator

indication of a fault associated with an actuator.

5.1.2 Communications

indication that the piece of equipment has experienced a communications failure.

5.1.3 DataRange

indication that the value of the data associated with a measured value or a calculation is outside of an expected range.

5.1.4 LogicProgram

indication that an error occurred in the logic program or programmable logic controller (PLC) associated with a piece of equipment.

5.1.5 MotionProgram

indication that an error occurred in the motion program associated with a piece of equipment.

5.1.6 System

general purpose indication associated with an electronic component of a piece of equipment or a controller that represents a fault that is not associated with the operator, program, or hardware.

5.2 Event Types

This section provides semantic information for `Event` types.

5.2.1 ActivationCount

accumulation of the number of times a function has attempted to, or is planned to attempt to, activate or be performed.

The default `subType` of `ActivationCount` is `ALL`.

The value of `ActivationCount` **MUST** be `integer`.

5.2.1.1 Subtypes of ActivationCount

- `ABORTED`

  accumulation of actions or activities that were attempted, but terminated before they could be completed.

- `ALL`

  accumulation of all actions, items, or activities being counted independent of the outcome.

- `BAD`

  accumulation of actions, items, or activities being counted that do not conform to specification or expectation.

- `COMPLETE`

  accumulation of actions, items, or activities that have been completed, independent of the outcome.

- `FAILED`

  accumulation of actions or activities that were attempted, but failed to complete or resulted in an unexpected or unacceptable outcome.

- `GOOD`

  accumulation of actions, items, or activities being counted that conform to specification or expectation.

- `REMAINING`

  accumulation of actions, items, or activities yet to be counted.

- `TARGET`

  goal of the operation or process.

5.2.2 ActiveAxes

set of axes currently associated with a `Path` or `Controller`.

The value of `ActiveAxes` **MUST** be a list of `string` of size `0..*`.

5.2.3 ActuatorState

operational state of an apparatus for moving or controlling a mechanism or system.

`ActuatorStateEnum` Enumeration:

- ACTIVE
  Actuator is operating.
- INACTIVE
  Actuator is not operating.

5.2.4 AdapterSoftwareVersion

originator's software version of the *adapter*.

5.2.5 AdapterURI

Uniform Resource Identifier (URI) of the *adapter*.

5.2.6 <<deprecated>>Alarm

DEPRECATED: Replaced with CONDITION category data items in Version 1.1.0.

5.2.6.1 Value Properties of Alarm

Table 18 lists the Value Properties of Alarm.

Value Property name	Value Property type	Multiplicity
<<deprecated>> code	AlarmCodeEnum	1
<<deprecated>> severity	AlarmSeverityEnum	1
<<deprecated>> nativeCode	string	1
<<deprecated>> state	AlarmStateEnum	1
<<deprecated>> lang	xslang	0..1

Table 18: Value Properties of Alarm

Descriptions for Value Properties of Alarm:

- <<deprecated>> code
  type of alarm.

- `<<deprecated>> severity`
  severity of the alarm.
- `<<deprecated>> nativeCode`
  native code for the piece of equipment.
- `<<deprecated>> state`
  state of the alarm.
- `<<deprecated>> lang`
  specifies the language of the alarm text.
  See *Ref IETF RFC 4646* (http://www.ietf.org/rfc/rfc4646.txt).

5.2.7 AlarmLimit

set of limits used to trigger warning or alarm indicators.

The `Entry key` **MUST** be one or more from the `AlarmLimitResult` keys.

`AlarmLimitResult` keys:

- `UpperLimit`
  upper conformance boundary for a variable.

  Note: immediate concern or action may be required.

  The value of `UpperLimit` **MUST** be `float`.
- `UpperWarning`
  upper boundary indicating increased concern and supervision may be required.
  The value of `UpperWarning` **MUST** be `float`.
- `LowerWarning`
  lower boundary indicating increased concern and supervision may be required.
  The value of `LowerWarning` **MUST** be `float`.
- `LowerLimit`
  lower conformance boundary for a variable.

  Note: immediate concern or action may be required.

  The value of `LowerLimit` **MUST** be `float`.

5.2.8 Application

application on a `Component`.

A `subType` **MUST** always be specified.

5.2.8.1 Subtypes of Application

- `INSTALL_DATE`
  date the hardware or software was installed.
  The value of `Application` **MUST** be `datetime`. See *Section 6.1.5 - datetime*.
  `datetime` Enumeration:
- `LICENSE`
  license code to validate or activate the hardware or software.
- `MANUFACTURER`
  corporate identity for the maker of the hardware or software.
- `RELEASE_DATE`
  date the hardware or software was released for general use.
  The value of `Application` **MUST** be `datetime`. See *Section 6.1.5 - datetime*.
  `datetime` Enumeration:
- `VERSION`
  version of the hardware or software.

5.2.9 AssetChanged

`assetId` of the *Asset* that has been added or changed.

5.2.9.1 Value Properties of AssetChanged

Table 19 lists the Value Properties of `AssetChanged`.

Value Property name	Value Property type	Multiplicity
assetType	string	0..1
hash	string	0..1

Table 19: Value Properties of AssetChanged

Descriptions for Value Properties of AssetChanged:

- assetType

  type of Asset changed. See *MTConnect Standard: Part 4.0 - Asset Information Model* for details on the Asset model.

- hash

  condensed message digest from a secure one-way hash function. *Ref FIPS PUB 180-4*

5.2.10 AssetCount

data set of the number of *Assets* of a given type for a *Device*.

The value of AssetCount **MUST** be integer.

5.2.11 AssetRemoved

assetId of the *Asset* that has been removed.

5.2.11.1 Value Properties of AssetRemoved

Table 20 lists the Value Properties of AssetRemoved.

Value Property name	Value Property type	Multiplicity
assetType	string	0..1
hash	string	0..1

Table 20: Value Properties of AssetRemoved

Descriptions for Value Properties of AssetRemoved:

- `assetType`

  type of `Asset` removed. See *MTConnect Standard: Part 4.0 - Asset Information Model* for details on the `Asset` model.

- `hash`

  condensed message digest from a secure one-way hash function. *Ref FIPS PUB 180-4*

5.2.12 Availability

agent's ability to communicate with the data source.

`AvailabilityEnum` Enumeration:

- `AVAILABLE`

  data source is active and capable of providing data.

- `UNAVAILABLE`

  data source is either inactive or not capable of providing data.

5.2.13 AxisCoupling

describes the way the axes will be associated to each other.

This is used in conjunction with `COUPLED_AXES` to indicate the way they are interacting.

`AxisCouplingEnum` Enumeration:

- `MASTER`

  axis is the master of the `CoupledAxes`.

- `SLAVE`

  axis is a slave to the `CoupledAxes`.

- `SYNCHRONOUS`

  axes are not physically connected to each other but are operating together in lock-step.

- `TANDEM`

  axes are physically connected to each other and operate as a single unit.

5.2.14 AxisFeedrateOverride

value of a signal or calculation issued to adjust the feedrate of an individual linear type axis.

The value of `AxisFeedrateOverride` **MUST** be `float`.

5.2.14.1 Subtypes of AxisFeedrateOverride

- `JOG`

  relating to momentary activation of a function or a movement.

  DEPRECATION WARNING: May be deprecated in the future.

  When the `JOG` subtype of `AxisFeedrateOverride` is applied, the resulting commanded feedrate for the axis is limited to the value of the original `JOG` subtype of the `AxisFeedrate` multiplied by the value of the `JOG` subtype of `AxisFeedrateOverride`.

- `PROGRAMMED`

  directive value without offsets and adjustments.

- `RAPID`

  performing an operation faster or in less time than nominal rate.

5.2.15 AxisInterlock

state of the axis lockout function when power has been removed and the axis is allowed to move freely.

`AxisInterlockEnum` Enumeration:

- `ACTIVE`

  axis lockout function is activated, power has been removed from the axis, and the axis is allowed to move freely.

- INACTIVE

  axis lockout function has not been activated, the axis may be powered, and the axis is capable of being controlled by another component.

5.2.16 AxisState

state of a Linear or Rotary component representing an axis.

AxisStateEnum Enumeration:

- HOME

  axis is in its home position.

- PARKED

  axis has been moved to a fixed position and is being maintained in that position either electrically or mechanically.

  Action is required to release the axis from this position.

- STOPPED

  axis is stopped.

- TRAVEL

  axis is in motion.

5.2.17 BatteryState

present status of the battery.

BatteryStateEnum Enumeration:

- CHARGED

  Component is at it's maximum rated charge level.

- CHARGING

  Component's charge is increasing.

- DISCHARGED

  Component is at it's minimum charge level.

- DISCHARGING

  Component's charge is decreasing.

5.2.18 Block

line of code or command being executed by a Controller entity.

5.2.19 BlockCount

total count of the number of blocks of program code that have been executed since execution started.

The value of BlockCount **MUST** be integer.

5.2.20 CharacteristicPersistentId

Universally Unique Identifier (UUID) of the *characteristic*.

The value of CharacteristicPersistentId **MUST** be ID. See *Section 6.1.2 - ID*.

5.2.21 CharacteristicStatus

pass/fail result of the measurement.

CharacteristicStatusEnum Enumeration:

- BASIC_OR_THEORETIC_EXACT_DIMENSION

  nominal provided without tolerance limits. *Ref QIF 3:2018 5.10.2.6*

- FAIL

  measurement is not within acceptable tolerances.

- INDETERMINATE
  measurement cannot be determined.
- NOT_ANALYZED
  measurement cannot be evaluated.
- PASS
  measurement is within acceptable tolerances.
- REWORK
  failed, but acceptable constraints achievable by utilizing additional manufacturing processes.
- SYSTEM_ERROR
  measurement is indeterminate due to an equipment failure.
- UNDEFINED
  status of measurement cannot be determined.

5.2.22 ChuckInterlock

state of an interlock function or control logic state intended to prevent the associated Chuck component from being operated.

ChuckInterlockEnum Enumeration:

- ACTIVE
  chuck cannot be unclamped.
- INACTIVE
  chuck can be unclamped.

5.2.22.1 Subtypes of ChuckInterlock

- MANUAL_UNCLAMP
  indication of the state of an operator controlled interlock that can inhibit the ability to initiate an unclamp action of an electronically controlled chuck.

When ChuckInterlockManualUnclamp is ACTIVE, it is expected that a chuck cannot be unclamped until ChuckInterlockManualUnclamp is set to INACTIVE.

5.2.23 ChuckState

operating state of a mechanism that holds a part or stock material during a manufacturing process.

It may also represent a mechanism that holds any other mechanism in place within a piece of equipment.

ChuckStateEnum Enumeration:

- CLOSED

  Chuck is closed to the point of a positive confirmation.

- OPEN

  Chuck is open to the point of a positive confirmation.

- UNLATCHED

  Chuck is not closed to the point of a positive confirmation and not open to the point of a positive confirmation.

  It is in an intermediate position.

5.2.24 ClockTime

time provided by a timing device at a specific point in time.

The value of ClockTime **MUST** be datetime. See *Section 6.1.5 - datetime*.

5.2.25 <<deprecated>>Code

programmatic code being executed.

DEPRECATED in *Version 1.1*.

5.2.26 ComponentData

tabular Event that represents a Component where the EntryDefinition identifies the Component and the CellDefinitions define the Component's observed DataItems.

If the Component multiplicity can be determined, the device model **MUST** use a fixed set of Components.

ComponentData **MUST** provide a DataItem Definition.

5.2.27 CompositionState

operating state of a mechanism represented by a Composition entity.

A subType **MUST** always be specified.

5.2.27.1 Subtypes of CompositionState

- ACTION
  indication of the operating state of a mechanism.
  CompositionStateActionEnum Enumeration:
  - ACTIVE
    Composition is operating.
  - INACTIVE
    Composition is not operating.
- LATERAL
  indication of the position of a mechanism that may move in a lateral direction.
  CompositionStateLateralEnum Enumeration:
  - LEFT
    position of the Composition is oriented to the left to the point of a positive confirmation.
  - RIGHT
    position of the Composition is oriented to the right to the point of a positive confirmation.

- TRANSITIONING
  position of the Composition is not oriented to the right to the point of a positive confirmation and is not oriented to the left to the point of a positive confirmation.
  It is in an intermediate position.

- MOTION

  indication of the open or closed state of a mechanism.

  CompositionStateMotionEnum Enumeration:

  - CLOSED
    position of the Composition is closed to the point of a positive confirmation.
  - OPEN
    position of the Composition is open to the point of a positive confirmation.
  - UNLATCHED
    position of the Composition is not open to the point of a positive confirmation and is not closed to the point of a positive confirmation.
    It is in an intermediate position.

- SWITCHED

  indication of the activation state of a mechanism.

  CompositionStateSwitchedEnum Enumeration:

  - OFF
    activation state of the Composition is in an OFF condition, it is not operating, or it is not powered.
  - ON
    activation state of the Composition is in an ON condition, it is operating, or it is powered.

- VERTICAL

  indication of the position of a mechanism that may move in a vertical direction.

  CompositionStateVerticalEnum Enumeration:

  - DOWN
    position of the Composition element is oriented in a downward direction to the point of a positive confirmation.

- TRANSITIONING
  position of the Composition element is not oriented in an upward direction to the point of a positive confirmation and is not oriented in a downward direction to the point of a positive confirmation.
  It is in an intermediate position.
- UP
  position of the Composition element is oriented in an upward direction to the point of a positive confirmation.

5.2.28 ConnectionStatus

status of the connection between an *adapter* and an *agent*.

ConnectionStatusEnum Enumeration:

- CLOSED
  no connection at all.
- ESTABLISHED
  open connection.
  The normal state for the data transfer phase of the connection.
- LISTEN
  agent is waiting for a connection request from an *adapter*.

5.2.29 ControlLimit

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

The Entry key **MUST** be one or more from the ControlLimitResult keys.

ControlLimitResult keys:

- UpperLimit
  upper conformance boundary for a variable.

  Note: immediate concern or action may be required.

The value of `UpperLimit` **MUST** be `float`.

- `UpperWarning`

  upper boundary indicating increased concern and supervision may be required.

  The value of `UpperWarning` **MUST** be `float`.

- `Nominal`

  ideal or desired value for a variable.

  The value of `Nominal` **MUST** be `float`.

- `LowerWarning`

  lower boundary indicating increased concern and supervision may be required.

  The value of `LowerWarning` **MUST** be `float`.

- `LowerLimit`

  lower conformance boundary for a variable.

  Note: immediate concern or action may be required.

  The value of `LowerLimit` **MUST** be `float`.

5.2.30 ControllerMode

current mode of the `Controller` component.

`ControllerModeEnum` Enumeration:

- `AUTOMATIC`

  `Controller` is configured to automatically execute a program.

- `EDIT`

  `Controller` is currently functioning as a programming device and is not capable of executing an active program.

- `<<deprecated>> FEED_HOLD`

  axes of the device are commanded to stop, but the spindle continues to function.

- MANUAL

  Controller is not executing an active program.

  It is capable of receiving instructions from an external source – typically an operator. The Controller executes operations based on the instructions received from the external source.

- MANUAL_DATA_INPUT

  operator can enter a series of operations for the Controller to perform.

  The Controller will execute this specific series of operations and then stop.

- SEMI_AUTOMATIC

  Controller is operating in a mode that restricts the active program from processing its next process step without operator intervention.

5.2.31 ControllerModeOverride

setting or operator selection that changes the behavior of a piece of equipment.

ControllerModeOverrideEnum Enumeration:

- OFF

  ControllerModeOverride is in the OFF state and the mode override is inactive.

- ON

  ControllerModeOverride is in the ON state and the mode override is active.

A subType **MUST** always be specified.

5.2.31.1 Subtypes of ControllerModeOverride

- DRY_RUN

  setting or operator selection used to execute a test mode to confirm the execution of machine functions.

  When DRY_RUN is ON, the equipment performs all of its normal functions, except no part or product is produced. If the equipment has a spindle, spindle operation is suspended.

- MACHINE_AXIS_LOCK

  setting or operator selection that changes the behavior of the controller on a piece of equipment.

  When MACHINE_AXIS_LOCK is ON, program execution continues normally, but no equipment motion occurs.

- OPTIONAL_STOP

  setting or operator selection that changes the behavior of the controller on a piece of equipment.

  The program execution is stopped after a specific program block is executed when OPTIONAL_STOP is ON.

  In the case of a G-Code program, a program block containing a M01 code designates the command for an OPTIONAL_STOP.

  Execution **MUST** change to OPTIONAL_STOP after a program block specifying an optional stop is executed and the ControllerModeOverride OPTIONAL_STOP selection is ON.

- SINGLE_BLOCK

  setting or operator selection that changes the behavior of the controller on a piece of equipment.

  Program execution is paused after each block of code is executed when SINGLE_BLOCK is ON.

  When SINGLE_BLOCK is ON, Execution **MUST** change to INTERRUPTED after completion of each block of code.

- TOOL_CHANGE_STOP

  setting or operator selection that changes the behavior of the controller on a piece of equipment.

  Program execution is paused when a command is executed requesting a cutting tool to be changed.

  Execution **MUST** change to INTERRUPTED after completion of the command requesting a cutting tool to be changed and TOOL_CHANGE_STOP is ON.

5.2.32 CoupledAxes

set of associated axes.

The value of CoupledAxes **MUST** be a list of string of size 0..*.

5.2.33 CycleCount

accumulation of the number of times a cyclic function has attempted to, or is planned to attempt to execute.

The default `subType` of `CycleCount` is `ALL`.

The value of `CycleCount` **MUST** be `integer`.

5.2.33.1 Subtypes of CycleCount

- `ABORTED`

  accumulation of actions or activities that were attempted, but terminated before they could be completed.

- `ALL`

  accumulation of all actions, items, or activities being counted independent of the outcome.

- `BAD`

  accumulation of actions, items, or activities being counted that do not conform to specification or expectation.

- `COMPLETE`

  accumulation of actions, items, or activities that have been completed, independent of the outcome.

- `FAILED`

  accumulation of actions or activities that were attempted, but failed to complete or resulted in an unexpected or unacceptable outcome.

- `GOOD`

  accumulation of actions, items, or activities being counted that conform to specification or expectation.

- `REMAINING`

  accumulation of actions, items, or activities yet to be counted.

- `TARGET`

  goal of the operation or process.

5.2.34 DateCode

time and date code associated with a material or other physical item.

The value of `DateCode` **MUST** be `datetime`. See *Section 6.1.5 - datetime*.

5.2.34.1 Subtypes of DateCode

- `EXPIRATION`

  time and date code relating to the expiration or end of useful life for a material or other physical item.

- `FIRST_USE`

  time and date code relating the first use of a material or other physical item.

- `MANUFACTURE`

  time and date code relating to the production of a material or other physical item.

5.2.35 DeactivationCount

accumulation of the number of times a function has attempted to, or is planned to attempt to, deactivate or cease.

The default `subType` of `DeactivationCount` is `ALL`.

The value of `DeactivationCount` **MUST** be `integer`.

5.2.35.1 Subtypes of DeactivationCount

- `ABORTED`

  accumulation of actions or activities that were attempted, but terminated before they could be completed.

- `ALL`

  accumulation of all actions, items, or activities being counted independent of the outcome.

- BAD

  accumulation of actions, items, or activities being counted that do not conform to specification or expectation.

- COMPLETE

  accumulation of actions, items, or activities that have been completed, independent of the outcome.

- FAILED

  accumulation of actions or activities that were attempted, but failed to complete or resulted in an unexpected or unacceptable outcome.

- GOOD

  accumulation of actions, items, or activities being counted that conform to specification or expectation.

- REMAINING

  accumulation of actions, items, or activities yet to be counted.

- TARGET

  goal of the operation or process.

5.2.36 DeviceAdded

UUID of new device added to an *MTConnect Agent*.

5.2.36.1 Value Properties of DeviceAdded

Table 21 lists the Value Properties of DeviceAdded.

Value Property name	Value Property type	Multiplicity
hash	string	0..1

Table 21: Value Properties of DeviceAdded

Descriptions for Value Properties of DeviceAdded:

- hash

  condensed message digest from a secure one-way hash function. *Ref FIPS PUB 180-4*

5.2.37 DeviceChanged

UUID of the device whose *metadata* has changed.

5.2.37.1 Value Properties of DeviceChanged

Table 22 lists the Value Properties of `DeviceChanged`.

Value Property name	Value Property type	Multiplicity
`hash`	`string`	0..1

Table 22: Value Properties of DeviceChanged

Descriptions for Value Properties of `DeviceChanged`:

- `hash`

  condensed message digest from a secure one-way hash function. *Ref FIPS PUB 180-4*

5.2.38 DeviceRemoved

UUID of a device removed from an *MTConnect Agent*.

5.2.38.1 Value Properties of DeviceRemoved

Table 23 lists the Value Properties of `DeviceRemoved`.

Value Property name	Value Property type	Multiplicity
`hash`	`string`	0..1

Table 23: Value Properties of DeviceRemoved

Descriptions for Value Properties of `DeviceRemoved`:

- `hash`

  condensed message digest from a secure one-way hash function. *Ref FIPS PUB 180-4*

5.2.39 DeviceUuid

identifier of another piece of equipment that is temporarily associated with a component of this piece of equipment to perform a particular function.

5.2.40 Direction

direction of motion.

<<deprecated>> DirectionEnum Enumeration:

- <<deprecated>> CLOCKWISE
  clockwise rotation using the right-hand rule.
- <<deprecated>> COUNTER_CLOCKWISE
  counter-clockwise rotation using the right-hand rule.
- <<deprecated>> NEGATIVE
- <<deprecated>> POSITIVE

A subType **MUST** always be specified.

5.2.40.1 Subtypes of Direction

- LINEAR
  direction of motion of a linear motion.
  DirectionLinearEnum Enumeration:
  - NEGATIVE
    linear position is decreasing.
  - NONE
    no direction.
  - POSITIVE
    linear position is increasing.

- ROTARY

  rotational direction of a rotary motion using the right hand rule convention.

  DirectionRotaryEnum Enumeration:

  - CLOCKWISE

    clockwise rotation using the right-hand rule.
  - COUNTER_CLOCKWISE

    counter-clockwise rotation using the right-hand rule.
  - NONE

    no direction.

5.2.41 DoorState

operational state of a Door component or composition element.

DoorStateEnum Enumeration:

- CLOSED

  Door is closed to the point of a positive confirmation.
- OPEN

  Door is open to the point of a positive confirmation.
- UNLATCHED

  Door is not closed to the point of a positive confirmation and not open to the point of a positive confirmation.

  It is in an intermediate position.

5.2.42 EmergencyStop

state of the emergency stop signal for a piece of equipment, controller path, or any other component or subsystem of a piece of equipment.

EmergencyStopEnum Enumeration:

- ARMED

  emergency stop circuit is complete and the piece of equipment, component, or composition is allowed to operate.

- TRIGGERED

  operation of the piece of equipment, component, or composition is inhibited.

5.2.43 EndOfBar

indication of whether the end of a piece of bar stock being feed by a bar feeder has been reached.

EndOfBarEnum Enumeration:

- NO

  EndOfBar has not been reached.

- YES

  EndOfBar has been reached.

The default subType of EndOfBar is PRIMARY.

5.2.43.1 Subtypes of EndOfBar

- AUXILIARY

  when multiple locations on a piece of bar stock are referenced as the indication for the EndOfBar, the additional location(s) **MUST** be designated as AUXILIARY indication(s) for the EndOfBar.

- PRIMARY

  specific applications **MAY** reference one or more locations on a piece of bar stock as the indication for the EndOfBar.

  The main or most important location **MUST** be designated as the PRIMARY indication for the EndOfBar.

  If no subType is specified, PRIMARY **MUST** be the default EndOfBar indication.

5.2.44 EquipmentMode

indication that a piece of equipment, or a sub-part of a piece of equipment, is performing specific types of activities.

`EquipmentModeEnum` Enumeration:

- `OFF`
  equipment is not functioning in the mode designated by the `subType`.
- `ON`
  equipment is functioning in the mode designated by the `subType`.

A `subType` **MUST** always be specified.

5.2.44.1 Subtypes of EquipmentMode

- `DELAY`
  elapsed time of a temporary halt of action.
- `LOADED`
  indication that the sub-parts of a piece of equipment are under load.
  Example: For traditional machine tools, this is an indication that the cutting tool is assumed to be engaged with the part.
- `OPERATING`
  indication that the major sub-parts of a piece of equipment are powered or performing any activity whether producing a part or product or not.
  Example: For traditional machine tools, this includes when the piece of equipment is `WORKING` or it is idle.
- `POWERED`
  indication that primary power is applied to the piece of equipment and, as a minimum, the controller or logic portion of the piece of equipment is powered and functioning or components that are required to remain on are powered.
  Example: Heaters for an extrusion machine that required to be powered even when the equipment is turned off.

- WORKING

  indication that a piece of equipment is performing any activity, the equipment is active and performing a function under load or not.

  Example: For traditional machine tools, this includes when the piece of equipment is LOADED, making rapid moves, executing a tool change, etc.

5.2.45 Execution

execution status of the Component.

ExecutionEnum Enumeration:

- ACTIVE

  Component is actively executing an instruction.

- FEED_HOLD

  motion of the active axes are commanded to stop at their current position.

- INTERRUPTED

  Component suspends the execution of the program due to an external signal.

  Action is required to resume execution.

- OPTIONAL_STOP

  command from the program has intentionally interrupted execution.

  The Component **MAY** have another state that indicates if the execution is interrupted or the execution ignores the interrupt instruction.

- PROGRAM_COMPLETED

  program completed execution.

- <<deprecated>> PROGRAM_OPTIONAL_STOP

  program has been intentionally optionally stopped using an M01 or similar code.

  DEPRECATED in *version 1.4* and replaced with OPTIONAL_STOP.

- PROGRAM_STOPPED

  command from the program has intentionally interrupted execution.

  Action is required to resume execution.

- `READY`

  `Component` is ready to execute instructions.

  It is currently idle.

- `STOPPED`

  `Component` program is not `READY` to execute.

- `WAIT`

  `Component` suspends execution while a secondary operation executes.

  Execution resumes automatically once the secondary operation completes.

5.2.46 FeatureMeasurement

tabular representation of assessing elements of a *feature*.

`FeatureMeasurement` **MAY** include a *characteristic* in which case it **MAY** include a `CHARACTERISTIC_STATUS`.

The `Entry key` **MUST** be one or more from the `FeatureMeasurementResult` keys.

`FeatureMeasurementResult` keys:

- `MeasurementId`

  identifier of this measurement.

  The value of `MeasurementId` **MUST** be `ID`. See *Section 6.1.2 - ID*.

- `FeaturePersistentId`

  UUID of the feature.

  The value of `FeaturePersistentId` **MUST** be `ID`. See *Section 6.1.2 - ID*.

- `CharacteristicPersistentId`

  UUID of the characteristic.

  The value of `CharacteristicPersistentId` **MUST** be `ID`. See *Section 6.1.2 - ID*.

- `MeasurementType`
  class of measurement being performed. *Ref QIF 3:2018 Section 6.3*
  Examples: `POINT`, `RADIUS`, `ANGLE`, `LENGTH`, etc.
  The value of `MeasurementType` **MUST** be `string`.
- `MeasurementValue`
  measurement based on the measurement type.
  The value of `MeasurementValue` **MUST** be `float`.
- `MeasurementUnits`
  engineering units of the measurement.
  The value of `MeasurementUnits` **MUST** be `string`.
- `CharacteristicStatus`
  pass/fail result of the measurement.
  The value of `CharacteristicStatus` **MUST** be one of the `CharacteristicStatusEnum` enumeration.
  `CharacteristicStatusEnum` Enumeration:
  - `BASIC_OR_THEORETIC_EXACT_DIMENSION`
    nominal provided without tolerance limits. *Ref QIF 3:2018 5.10.2.6*
  - `FAIL`
    measurement is not within acceptable tolerances.
  - `INDETERMINATE`
    measurement cannot be determined.
  - `NOT_ANALYZED`
    measurement cannot be evaluated.
  - `PASS`
    measurement is within acceptable tolerances.
  - `REWORK`
    failed, but acceptable constraints achievable by utilizing additional manufacturing processes.
  - `SYSTEM_ERROR`
    measurement is indeterminate due to an equipment failure.
  - `UNDEFINED`
    status of measurement cannot be determined.

- `UncertaintyType`
  method used to compute *standard uncertainty*.
  `UncertaintyTypeEnum` Enumeration:
  - `COMBINED`
    combined standard uncertainty.
  - `MEAN`
    standard uncertainty using arithmetic mean or average the observations. *Ref JCGM 100:2008 4.2*
- `Uncertainty`
  uncertainty specified by `UNCERTAINTY_TYPE`.
  The value of `Uncertainty` **MUST** be `float`.

5.2.47 FeaturePersisitentId

UUID of a *feature*. *Ref ISO 10303 AP 242/239*.

The value of `FeaturePersisitentId` **MUST** be `ID`. See *Section 6.1.2 - ID*.

5.2.48 Firmware

embedded software of a `Component` .

A `subType` **MUST** always be specified.

5.2.48.1 Subtypes of Firmware

- `INSTALL_DATE`
  date the hardware or software was installed.
  The value of `Firmware` **MUST** be `datetime`. See *Section 6.1.5 - datetime*.
  `datetime` Enumeration:
- `LICENSE`
  license code to validate or activate the hardware or software.

- MANUFACTURER

  corporate identity for the maker of the hardware or software.

- RELEASE_DATE

  date the hardware or software was released for general use.

  The value of Firmware **MUST** be datetime. See *Section 6.1.5 - datetime*.

  datetime Enumeration:

- VERSION

  version of the hardware or software.

5.2.49 FixtureId

identifier for a fixture.

5.2.50 FunctionalMode

current intended production status of the Component.

FunctionalModeEnum Enumeration:

- MAINTENANCE

  Component is not currently producing product.

  It is currently being repaired, waiting to be repaired, or has not yet been returned to a normal production status after maintenance has been performed.

- PROCESS_DEVELOPMENT

  Component is being used to prove-out a new process, testing of equipment or processes, or any other active use that does not result in the production of product.

- PRODUCTION

  Component is currently producing product, ready to produce product, or its current intended use is to be producing product.

- SETUP

  Component is not currently producing product.

  It is being prepared or modified to begin production of product.

- `TEARDOWN`

  `Component` is not currently producing product.

  Typically, it has completed the production of a product and is being modified or returned to a neutral state such that it may then be prepared to begin production of a different product.

5.2.51 Hardness

hardness of a material.

The value of `Hardness` **MUST** be `float`.

A `subType` **MUST** always be specified.

5.2.51.1 Subtypes of Hardness

- `BRINELL`

  scale to measure the resistance to deformation of a surface.

- `LEEB`

  scale to measure the elasticity of a surface.

- `MOHS`

  scale to measure the resistance to scratching of a surface.

- `ROCKWELL`

  scale to measure the resistance to deformation of a surface.

- `SHORE`

  scale to measure the resistance to deformation of a surface.

- `VICKERS`

  scale to measure the resistance to deformation of a surface.

5.2.52 Hardware

hardware of a `Component`.

A `subType` **MUST** always be specified.

5.2.52.1 Subtypes of Hardware

- `INSTALL_DATE`
  date the hardware or software was installed.
  The value of `Hardware` **MUST** be `datetime`. See *Section 6.1.5 - datetime*.
  `datetime` Enumeration:
- `LICENSE`
  license code to validate or activate the hardware or software.
- `MANUFACTURER`
  corporate identity for the maker of the hardware or software.
- `RELEASE_DATE`
  date the hardware or software was released for general use.
  The value of `Hardware` **MUST** be `datetime`. See *Section 6.1.5 - datetime*.
  `datetime` Enumeration:
- `VERSION`
  version of the hardware or software.

5.2.53 HostName

name of the host computer supplying data.

5.2.54 LeakDetect

indication designating whether a leak has been detected.

`LeakDetectEnum` Enumeration:

- DETECTED

  leak is currently being detected.

- NOT_DETECTED

  leak is currently not being detected.

5.2.55 Library

software library on a Component

A subType **MUST** always be specified.

5.2.55.1 Subtypes of Library

- INSTALL_DATE

  date the hardware or software was installed.

  The value of Library **MUST** be datetime. See *Section 6.1.5 - datetime*.

  datetime Enumeration:

- LICENSE

  license code to validate or activate the hardware or software.

- MANUFACTURER

  corporate identity for the maker of the hardware or software.

- RELEASE_DATE

  date the hardware or software was released for general use.

  The value of Library **MUST** be datetime. See *Section 6.1.5 - datetime*.

  datetime Enumeration:

- VERSION

  version of the hardware or software.

5.2.56 <<deprecated>>Line

current line of code being executed.

DEPRECATED in *Version 1.4.0*.

5.2.56.1 Subtypes of Line

- `MAXIMUM`
  maximum line number of the code being executed.
- `MINIMUM`
  minimum line number of the code being executed.

5.2.57 LineLabel

identifier for a `Block` of code in a `Program`.

5.2.58 LineNumber

position of a block of program code within a control program.

The value of `LineNumber` **MUST** be `integer`.

5.2.58.1 Subtypes of LineNumber

- `ABSOLUTE`
  position of a block of program code relative to the beginning of the control program.
- `INCREMENTAL`
  position of a block of program code relative to the occurrence of the last `LineLabel` encountered in the control program.

5.2.59 LoadCount

accumulation of the number of times an operation has attempted to, or is planned to attempt to, load materials, parts, or other items.

The default `subType` of `LoadCount` is `ALL`.

The value of `LoadCount` **MUST** be `integer`.

5.2.59.1 Subtypes of LoadCount

- `ABORTED`

  accumulation of actions or activities that were attempted, but terminated before they could be completed.

- `ALL`

  accumulation of all actions, items, or activities being counted independent of the outcome.

- `BAD`

  accumulation of actions, items, or activities being counted that do not conform to specification or expectation.

- `COMPLETE`

  accumulation of actions, items, or activities that have been completed, independent of the outcome.

- `FAILED`

  accumulation of actions or activities that were attempted, but failed to complete or resulted in an unexpected or unacceptable outcome.

- `GOOD`

  accumulation of actions, items, or activities being counted that conform to specification or expectation.

- `REMAINING`

  accumulation of actions, items, or activities yet to be counted.

- `TARGET`

  goal of the operation or process.

5.2.60 LockState

state or operating mode of a `Lock`.

`LockStateEnum` Enumeration:

- `LOCKED`

  mechanism is engaged and preventing the associated `Component` from being opened or operated.

- `UNLOCKED`

  mechanism is disengaged and the associated `Component` is able to be opened or operated.

5.2.61 MTConnectVersion

reference version of the MTConnect Standard supported by the *adapter*.

5.2.62 MaintenanceList

actions or activities to be performed in support of a piece of equipment.

If the `INTERVAL` `key` is not provided, it is assumed `ABSOLUTE`.

If the `DIRECTION` `key` is not provided, it is assumed `UP`.

If the `UNITS` `key` is not provided, it is assumed to be `COUNT`.

The `Entry` `key` **MUST** be one or more from the `MaintenanceListResult` keys.

`MaintenanceListResult` keys:

- `Value`

  current interval value of the activity.

  The value of `Value` **MUST** be `float`.

- `Interval`

interval of the value observed.

`MaintenanceListIntervalEnum` Enumeration:

  - `ABSOLUTE`
  - `INCREMENTAL`

- `NextServiceDate`

  next date/time stamp that maintenance should be performed.

  The value of `NextServiceDate` **MUST** be `datetime`. See *Section 6.1.5 - datetime*.

- `Reset`

  last date/time stamp of the observation was reset.

  The value of `Reset` **MUST** be `datetime`. See *Section 6.1.5 - datetime*.

- `Severity`

  level of severity on a scale of 1-10.

  The value of `Severity` **MUST** be `integer`.

- `Direction`

  direction of the value observed.

  `MaintenanceListDirectionEnum` Enumeration:

  - `DOWN`
  - `UP`

- `Name`

  identifier of the maintenance activity.

  The value of `Name` **MUST** be `string`.

- `LastServiceDate`

  last date/time stamp that maintenance was performed.

  The value of `LastServiceDate` **MUST** be `datetime`. See *Section 6.1.5 - datetime*.

- `Units`

  same as `DataItem` `units`. See *MTConnect Standard: Part 2.0 - Device Information Model*.

  The value of `Units` **MUST** be one of the `UnitEnum` enumeration.

- `Target`

  target value of the next maintenance.

  The value of `Target` **MUST** be `float`.

5.2.63 Material

identifier of a material used or consumed in the manufacturing process.

5.2.64 MaterialLayer

identifies the layers of material applied to a part or product as part of an additive manufacturing process.

The value of `MaterialLayer` **MUST** be `integer`.

5.2.64.1 Subtypes of MaterialLayer

- `ACTUAL`

  measured or reported value of an observation.

- `TARGET`

  goal of the operation or process.

5.2.65 MeasurementType

class of measurement being performed. *Ref QIF 3:2018 Section 6.3*

Examples: `POINT`, `RADIUS`, `ANGLE`, `LENGTH`, etc.

5.2.66 MeasurementUnits

engineering units of the measurement.

5.2.67 MeasurementValue

measurement based on the measurement type.

The value of `MeasurementValue` **MUST** be `double`.

5.2.68 Message

information to be transferred from a piece of equipment to a client software application.

5.2.68.1 Value Properties of Message

Table 24 lists the Value Properties of `Message`.

Value Property name	Value Property type	Multiplicity
`nativeCode`	`string`	0..1

Table 24: Value Properties of Message

Descriptions for Value Properties of `Message`:

- `nativeCode`
  control system local identification of the information being transferred.

5.2.69 Network

network details of a `Component`.

A `subType` **MUST** always be specified.

5.2.69.1 Subtypes of Network

- `GATEWAY`
  Gateway for the component network.

- `IPV4_ADDRESS`
  IPV4 network address of the component.
- `IPV6_ADDRESS`
  IPV6 network address of the component.
- `MAC_ADDRESS`
  Media Access Control Address.
  The unique physical address of the network hardware.
- `SUBNET_MASK`
  SubNet mask for the component network.
- `VLAN_ID`
  layer2 Virtual Local Network (VLAN) ID for the component network.
- `WIRELESS`
  identifies whether the connection type is wireless.
  `NetworkWirelessEnum` Enumeration:
  - `NO`
  - `YES`

5.2.70 NetworkPort

number of the TCP/IP or UDP/IP port for the connection endpoint.

The value of `NetworkPort` **MUST** be `integer`.

5.2.71 OperatingMode

state of `Component` or `Composition` that describes the automatic or manual operation of the entity.

`OperatingModeEnum` Enumeration:

- `AUTOMATIC`
  automatically execute instructions from a recipe or program.

Note: Setpoint comes from a recipe.

- MANUAL

  execute instructions from an external agent or person.

  Note 1 to entry: Valve or switch is manipulated by an agent/person.

  Note 2 to entry: Direct control of the PID output. % of the range: A user manually sets the % output, not the setpoint.

- SEMI_AUTOMATIC

  executes a single instruction from a recipe or program.

  Note 1 to entry: Setpoint is entered and fixed, but the PID is controlling.

  Note 2 to entry: Still goes through the PID control system.

  Note 3 to entry: Manual fixed entry from a recipe.

5.2.72 OperatingSystem

Operating System (OS) of a Component.

A subType **MUST** always be specified.

5.2.72.1 Subtypes of OperatingSystem

- INSTALL_DATE

  date the hardware or software was installed.

  The value of OperatingSystem **MUST** be datetime. See *Section 6.1.5 - datetime*.

  datetime Enumeration:

- LICENSE

  license code to validate or activate the hardware or software.

- MANUFACTURER

  corporate identity for the maker of the hardware or software.

- RELEASE_DATE

  date the hardware or software was released for general use.

  The value of OperatingSystem **MUST** be datetime. See *Section 6.1.5 - datetime*.

  datetime Enumeration:

- VERSION

  version of the hardware or software.

5.2.73 OperatorId

identifier of the person currently responsible for operating the piece of equipment.

5.2.74 PalletId

identifier for a pallet.

5.2.75 PartCount

aggregate count of parts.

The value of PartCount **MUST** be integer.

The default subType of PartCount is ALL.

5.2.75.1 Subtypes of PartCount

- ABORTED

  accumulation of actions or activities that were attempted, but terminated before they could be completed.

- ALL

  accumulation of all actions, items, or activities being counted independent of the outcome.

- BAD

  accumulation of actions, items, or activities being counted that do not conform to specification or expectation.

- COMPLETE

  accumulation of actions, items, or activities that have been completed, independent of the outcome.

- FAILED

  accumulation of actions or activities that were attempted, but failed to complete or resulted in an unexpected or unacceptable outcome.

- GOOD

  accumulation of actions, items, or activities being counted that conform to specification or expectation.

- REMAINING

  accumulation of actions, items, or activities yet to be counted.

- TARGET

  goal of the operation or process.

5.2.76 PartCountType

interpretation of PART_COUNT.

PartCountTypeEnum Enumeration:

- BATCH

  pre-specified group of items.

- EACH

  count is of individual items.

5.2.77 PartDetect

indication designating whether a part or work piece has been detected or is present.

PartDetectEnum Enumeration:

- NOT_PRESENT
  part or work piece is not detected or is not present.
- PRESENT
  part or work piece is detected or is present.

5.2.78 PartGroupId

identifier given to a collection of individual parts.

If no subType is specified, UUID is default.

The default subType of PartGroupId is UUID.

5.2.78.1 Subtypes of PartGroupId

- BATCH
  identifier that references a group of parts produced in a batch.
- HEAT_TREAT
  identifier used to reference a material heat number.
- LOT
  identifier that references a group of parts tracked as a lot.
- RAW_MATERIAL
  material that is used to produce parts.
- UUID
  universally unique identifier as specified in ISO 11578 or RFC 4122.

5.2.79 PartId

identifier of a part in a manufacturing operation.

5.2.80 PartKindId

identifier given to link the individual occurrence to a class of parts, typically distinguished by a particular part design.

If no `subType` is specified, `UUID` is default.

The default `subType` of `PartKindId` is `UUID`.

5.2.80.1 Subtypes of PartKindId

- `PART_FAMILY`

  identifier given to a group of parts having similarities in geometry, manufacturing process, and/or functions.

- `PART_NAME`

  word or set of words by which a part is known, addressed, or referred to.

- `PART_NUMBER`

  identifier of a particular part design or model.

- `UUID`

  universally unique identifier as specified in ISO 11578 or RFC 4122.

5.2.81 <<deprecated>>PartNumber

identifier of a part or product moving through the manufacturing process.

DEPRECATED in *Version 1.7*. `PART_NUMBER` is now a `subType` of `PART_KIND_ID`.

5.2.82 PartProcessingState

particular condition of the part occurrence at a specific time.

`PartProcessingStateEnum` Enumeration:

- IN_PROCESS

  part occurrence is actively being processed.

- IN_TRANSIT

  part occurrence is being transported to its destination.

- NEEDS_PROCESSING

  part occurrence is not actively being processed, but the processing has not ended.

  Processing requirements exist that have not yet been fulfilled. This is the default entry state when the part occurrence is originally received. In some cases, the part occurrence may return to this state while it waits for additional processing to be performed.

- PROCESSING_ENDED

  part occurrence is no longer being processed.

  A general state when the reason for termination is unknown.

- PROCESSING_ENDED_ABORTED

  processing of the part occurrence has come to a premature end.

- PROCESSING_ENDED_COMPLETE

  part occurrence has completed processing successfully.

- PROCESSING_ENDED_LOST

  terminal state when the part occurrence has been removed from the equipment by an external entity and it no longer exists at the equipment.

- PROCESSING_ENDED_REJECTED

  part occurrence has been processed completely. However, the processing may have a problem.

- PROCESSING_ENDED_SKIPPED

  part occurrence has been skipped for processing on the piece of equipment.

- PROCESSING_ENDED_STOPPED

  process has been stopped during the processing.

  The part occurrence will require special treatment.

- TRANSIT_COMPLETE

  part occurrence has been placed at its designated destination.

- WAITING_FOR_TRANSIT

  part occurrence is waiting for transit.

5.2.83 PartStatus

state or condition of a part.

If unique identifier is given, part status is for that individual. If group identifier is given without a unique identifier, then the status is assumed to be for the whole group.

`PartStatusEnum` Enumeration:

- `FAIL`

  part does not conform to some given requirements.

- `PASS`

  part conforms to given requirements.

5.2.84 PartUniqueId

identifier given to a distinguishable, individual part.

If no `subType` is specified, `UUID` is default.

The default `subType` of `PartUniqueId` is `UUID`.

5.2.84.1 Subtypes of PartUniqueId

- `RAW_MATERIAL`

  material that is used to produce parts.

- `SERIAL_NUMBER`

  serial number that uniquely identifies a specific part.

- `UUID`

  universally unique identifier as specified in ISO 11578 or RFC 4122.

5.2.85 PathFeedrateOverride

value of a signal or calculation issued to adjust the feedrate for the axes associated with a `Path` component that may represent a single axis or the coordinated movement of multiple axes.

The value of `PathFeedrateOverride` **MUST** be `float`.

5.2.85.1 Subtypes of PathFeedrateOverride

- `JOG`
  relating to momentary activation of a function or a movement.
  DEPRECATION WARNING: May be deprecated in the future.
- `PROGRAMMED`
  directive value without offsets and adjustments.
- `RAPID`
  performing an operation faster or in less time than nominal rate.

5.2.86 PathMode

describes the operational relationship between a `Path` entity and another `Path` entity for pieces of equipment comprised of multiple logical groupings of controlled axes or other logical operations.

`PathModeEnum` Enumeration:

- `INDEPENDENT`
  path is operating independently and without the influence of another path.
- `MASTER`
  path provides information or state values that influences the operation of other `DataItem` of similar type.
- `MIRROR`
  axes associated with the path are mirroring the motion of the `MASTER` path.

- SYNCHRONOUS

  physical or logical parts which are not physically connected to each other but are operating together.

5.2.87 PowerState

indication of the status of the source of energy for an entity to allow it to perform its intended function or the state of an enabling signal providing permission for the entity to perform its functions.

PowerStateEnum Enumeration:

- OFF

  source of energy for an entity or the enabling signal providing permission for the entity to perform its function(s) is not present or is disconnected.

- ON

  source of energy for an entity or the enabling signal providing permission for the entity to perform its function(s) is present and active.

5.2.87.1 Subtypes of PowerState

- CONTROL

  state of the enabling signal or control logic that enables or disables the function or operation of the entity.

- LINE

  state of the power source for the entity.

5.2.88 <<deprecated>>PowerStatus

status of the Component.

DEPRECATED in *Version 1.1.0*.

<<deprecated>> PowerStatusEnum Enumeration:

- <<deprecated>> OFF
- <<deprecated>> ON

5.2.89 ProcessAggregateId

identifier given to link the individual occurrence to a group of related occurrences, such as a process step in a process plan.

5.2.89.1 Subtypes of ProcessAggregateId

- ORDER_NUMBER

  identifier of the authorization of the process occurrence. Synonyms include “job id”, “work order”.

- PROCESS_PLAN

  identifier of the process plan that this occurrence belongs to. Synonyms include “routing id”, “job id”.

- PROCESS_STEP

  identifier of the step in the process plan that this occurrence corresponds to. Synonyms include “operation id”.

5.2.90 ProcessKindId

identifier given to link the individual occurrence to a class of processes or process definition.

5.2.90.1 Subtypes of ProcessKindId

- ISO_STEP_EXECUTABLE

  reference to a ISO 10303 Executable.

- PROCESS_NAME

  word or set of words by which a process being executed (process occurrence) by the device is known, addressed, or referred to.

- UUID
  universally unique identifier as specified in ISO 11578 or RFC 4122.

5.2.91 ProcessOccurrenceId

identifier of a process being executed by the device.

5.2.91.1 Subtypes of ProcessOccurrenceId

- ACTIVITY
  phase or segment of a recipe or program.
- OPERATION
  step of a discrete manufacturing process.
- RECIPE
  process as part of product production; can be a subprocess of a larger process.
- SEGMENT
  phase of a recipe process.

5.2.92 ProcessState

particular condition of the process occurrence at a specific time.

ProcessStateEnum Enumeration:

- ABORTED
  process occurrence has come to a premature end and cannot be resumed.
- ACTIVE
  process occurrence is actively executing.
- COMPLETE
  process occurrence is now finished.

- INITIALIZING
  device is preparing to execute the process occurrence.
- INTERRUPTED
  process occurrence has been stopped and may be resumed.
- READY
  process occurrence is ready to be executed.

5.2.93 ProcessTime

time and date associated with an activity or event.

A subType **MUST** always be specified.

5.2.93.1 Subtypes of ProcessTime

- COMPLETE
  time and date associated with the completion of an activity or event.
- START
  boundary when an activity or an event commences.
- TARGET_COMPLETION
  projected time and date associated with the end or completion of an activity or event.

5.2.94 Program

name of the logic or motion program being executed by the Controller component.

5.2.94.1 Subtypes of Program

- ACTIVE
  identity of the logic or motion program currently executing.

- ACTIVITY

  phase or segment of a recipe or program.

- MAIN

  identity of the primary logic or motion program currently being executed.

  It is the starting nest level in a call structure and may contain calls to sub programs.

- OPERATION

  step of a discrete manufacturing process.

- RECIPE

  process as part of product production; can be a subprocess of a larger process.

- SCHEDULE

  identity of a control program that is used to specify the order of execution of other programs.

- SEGMENT

  phase of a recipe process.

5.2.95 ProgramComment

comment or non-executable statement in the control program.

5.2.95.1 Subtypes of ProgramComment

- ACTIVE

  identity of the logic or motion program currently executing.

- MAIN

  identity of the primary logic or motion program currently being executed.

  It is the starting nest level in a call structure and may contain calls to sub programs.

- SCHEDULE

  identity of a control program that is used to specify the order of execution of other programs.

5.2.96 ProgramEdit

indication of the status of the Controller components program editing mode.

A program may be edited while another is executed.

ProgramEditEnum Enumeration:

- ACTIVE

  Controller is in the program edit mode.

- NOT_READY

  Controller is being inhibited by a function from entering the program edit mode.

- READY

  Controller is capable of entering the program edit mode and no function is inhibiting a change to that mode.

5.2.97 ProgramEditName

name of the program being edited.

This is used in conjunction with ProgramEdit when in ACTIVE state.

5.2.98 ProgramHeader

non-executable header section of the control program.

The default subType of ProgramHeader is MAIN.

5.2.98.1 Subtypes of ProgramHeader

- ACTIVE

  identity of the logic or motion program currently executing.

- MAIN

  identity of the primary logic or motion program currently being executed.

  It is the starting nest level in a call structure and may contain calls to sub programs.

- SCHEDULE

  identity of a control program that is used to specify the order of execution of other programs.

5.2.99 ProgramLocation

URI for the source file associated with Program.

5.2.99.1 Subtypes of ProgramLocation

- ACTIVE

  identity of the logic or motion program currently executing.

- MAIN

  identity of the primary logic or motion program currently being executed.

  It is the starting nest level in a call structure and may contain calls to sub programs.

- SCHEDULE

  identity of a control program that is used to specify the order of execution of other programs.

5.2.100 ProgramLocationType

defines whether the logic or motion program defined by Program is being executed from the local memory of the controller or from an outside source.

ProgramLocationTypeEnum Enumeration:

- EXTERNAL

  not managed by the controller.

- LOCAL

  managed by the controller.

5.2.100.1 Subtypes of ProgramLocationType

- `ACTIVE`
  identity of the logic or motion program currently executing.
- `MAIN`
  identity of the primary logic or motion program currently being executed.
  It is the starting nest level in a call structure and may contain calls to sub programs.
- `SCHEDULE`
  identity of a control program that is used to specify the order of execution of other programs.

5.2.101 ProgramNestLevel

indication of the nesting level within a control program that is associated with the code or instructions that is currently being executed.

If an initial value is not defined, the nesting level associated with the highest or initial nesting level of the program **MUST** default to zero (0).

The value of `ProgramNestLevel` **MUST** be `integer`.

5.2.102 RotaryMode

current operating mode for a `Rotary` type axis.

`RotaryModeEnum` Enumeration:

- `CONTOUR`
  position of the axis is being interpolated.
- `INDEX`
  axis is configured to index.
- `SPINDLE`
  axis is functioning as a spindle.

5.2.103 RotaryVelocityOverride

percentage change to the velocity of the programmed velocity for a `Rotary` axis.

This command represents a percentage change to the velocity calculated by a logic or motion program or set by a switch for a `Rotary` type axis.

The value of `RotaryVelocityOverride` **MUST** be `float`.

5.2.104 Rotation

three space angular displacement of an object or coordinate system relative to a *cartesian coordinate system*.

The `units` of `Rotation` **MUST** be `DEGREE_3D`.

The value of `Rotation` **MUST** be a list of `float` of size `0..3`.

5.2.105 SensorAttachment

attachment between a sensor and an entity.

The `Entry key` **MUST** be one or more from the `SensorAttachmentResult` keys.

`SensorAttachmentResult` keys:

- `SensorId`

  identity of a sensor used to observe some measurement of an item.

  The value of `SensorId` **MUST** be `ID`. See *Section 6.1.2 - ID*.

5.2.106 SensorState

detection result of a sensor.

The default `subType` of `SensorState` is `BINARY`.

5.2.106.1 Subtypes of SensorState

- BINARY

  detection result of a sensor.

  where the state is observed as a binary data type.

  The value of SensorState **MUST** be binary. See *Section 6.1.18 - binary*.

  binary Enumeration:

- BOOLEAN

  detection result of a sensor.

  where the state is observed as a boolean data type.

  The value of SensorState **MUST** be boolean.

  boolean Enumeration:

- DETECT

  detection result of a sensor.

  where the state is indicated by the presence or existence of something.

  The value of SensorState **MUST** be one of the CompositionStateSwitchedEnum enumeration.

- ENUMERATED

  detection result of a sensor.

  where the state is observed as a set containing a restricted number of discrete values where each discrete value is named and unique. *Ref ISO 21961:2003, 013*

  The value of SensorState **MUST** be integer.

  integer Enumeration:

5.2.107 SerialNumber

serial number associated with a Component, Asset, or Device.

5.2.108 SpecificationLimit

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

The `Entry key` **MUST** be one or more from the `SpecificationLimitResult` keys.

`SpecificationLimitResult` keys:

- `UpperLimit`
  upper conformance boundary for a variable.

  Note: immediate concern or action may be required.

  The value of `UpperLimit` **MUST** be `float`.
- `Nominal`
  ideal or desired value for a variable.
  The value of `Nominal` **MUST** be `float`.
- `LowerLimit`
  lower conformance boundary for a variable.

  Note: immediate concern or action may be required.

  The value of `LowerLimit` **MUST** be `float`.

5.2.109 SpindleInterlock

indication of the status of the spindle for a piece of equipment when power has been removed and it is free to rotate.

`SpindleInterlockEnum` Enumeration:

- `ACTIVE`
  power has been removed and the spindle cannot be operated.
- `INACTIVE`
  spindle has not been deactivated.

5.2.110 ToolAssetId

identifier of an individual tool asset.

5.2.111 ToolGroup

identifier for the tool group associated with a specific tool. Commonly used to designate spare tools.

5.2.112 <<deprecated>>ToolId

identifier of the tool currently in use for a given `Path`.

DEPRECATED in *Version 1.2.0*. See `TOOL_ASSET_ID`.

5.2.113 ToolNumber

identifier assigned by the `Controller` component to a cutting tool when in use by a piece of equipment.

5.2.114 ToolOffset

reference to the tool offset variables applied to the active cutting tool.

The value of `ToolOffset` **MUST** be `float`.

A `subType` **MUST** always be specified.

5.2.114.1 Subtypes of ToolOffset

- `LENGTH`
  reference to a length type tool offset variable.
- `RADIAL`
  reference to a radial type tool offset variable.

5.2.115 ToolOffsets

tabular representation of properties of each addressable tool offset.

5.2.116 TransferCount

accumulation of the number of times an operation has attempted to, or is planned to attempt to, transfer materials, parts, or other items from one location to another.

The default `subType` of `TransferCount` is `ALL`.

The value of `TransferCount` **MUST** be `integer`.

5.2.116.1 Subtypes of TransferCount

- `ABORTED`

  accumulation of actions or activities that were attempted, but terminated before they could be completed.

- `ALL`

  accumulation of all actions, items, or activities being counted independent of the outcome.

- `BAD`

  accumulation of actions, items, or activities being counted that do not conform to specification or expectation.

- `COMPLETE`

  accumulation of actions, items, or activities that have been completed, independent of the outcome.

- `FAILED`

  accumulation of actions, items, or activities being counted that do not conform to specification or expectation.

- `GOOD`

  accumulation of actions, items, or activities being counted that conform to specification or expectation.

- REMAINING

  accumulation of actions, items, or activities yet to be counted.

- TARGET

  goal of the operation or process.

5.2.117 Translation

three space linear displacement of an object or coordinate system relative to a *cartesian coordinate system*.

The units of Translation **MUST** be MILLIMETER_3D.

The value of Translation **MUST** be a list of float of size 0..3.

5.2.118 Uncertainty

uncertainty specified by UncertaintyType.

The value of Uncertainty **MUST** be double.

5.2.119 UncertaintyType

method used to compute *standard uncertainty*.

The value of UncertaintyType **MUST** be one of the UncertaintyTypeEnum enumeration.

UncertaintyTypeEnum Enumeration:

- COMBINED

  combined standard uncertainty.

- MEAN

  standard uncertainty using arithmetic mean or average the observations. *Ref JCGM 100:2008 4.2*

5.2.120 UnloadCount

accumulation of the number of times an operation has attempted to, or is planned to attempt to, unload materials, parts, or other items.

The default `subType` of `UnloadCount` is `ALL`.

The value of `UnloadCount` **MUST** be `integer`.

5.2.120.1 Subtypes of UnloadCount

- `ABORTED`

  accumulation of actions or activities that were attempted, but terminated before they could be completed.

- `ALL`

  accumulation of all actions, items, or activities being counted independent of the outcome.

- `BAD`

  accumulation of actions, items, or activities being counted that do not conform to specification or expectation.

- `COMPLETE`

  accumulation of actions, items, or activities that have been completed, independent of the outcome.

- `FAILED`

  accumulation of actions, items, or activities being counted that do not conform to specification or expectation.

- `GOOD`

  accumulation of actions, items, or activities being counted that conform to specification or expectation.

- `REMAINING`

  accumulation of actions, items, or activities yet to be counted.

- `TARGET`

  goal of the operation or process.

5.2.121 User

identifier of the person currently responsible for operating the piece of equipment.

A `subType` **MUST** always be specified.

5.2.121.1 Subtypes of User

- `MAINTENANCE`

  identifier of the person currently responsible for performing maintenance on the piece of equipment.

- `OPERATOR`

  identifier of the person currently responsible for operating the piece of equipment.

- `SET_UP`

  identifier of the person currently responsible for preparing a piece of equipment for production or restoring the piece of equipment to a neutral state after production.

5.2.122 ValveState

state of a valve is one of open, closed, or transitioning between the states.

`ValveStateEnum` Enumeration:

- `CLOSED`

  `ValveState` where flow is not possible, the aperture is static, and the valve is completely shut.

- `CLOSING`

  valve is transitioning from an `OPEN` state to a `CLOSED` state.

- `OPEN`

  `ValveState` where flow is allowed and the aperture is static.

  Note: For a binary value, `OPEN` indicates the valve has the maximum possible aperture.

- `OPENING`

  valve is transitioning from a `CLOSED` state to an `OPEN` state.

5.2.122.1 Subtypes of ValveState

- ACTUAL
  measured or reported value of an observation.
- PROGRAMMED
  directive value without offsets and adjustments.

5.2.123 Variable

data whose meaning may change over time due to changes in the operation of a piece of equipment or the process being executed on that piece of equipment.

5.2.124 WaitState

indication of the reason that Execution is reporting a value of WAIT.

When Execution result is not WAIT, the isUnavailable property of WaitState **MUST** be true.

WaitStateEnum Enumeration:

- MATERIAL_LOAD
  execution is waiting while material is being loaded.
- MATERIAL_UNLOAD
  execution is waiting while material is being unloaded.
- PART_LOAD
  execution is waiting while one or more discrete workpieces are being loaded.
- PART_UNLOAD
  execution is waiting while one or more discrete workpieces are being unloaded.
- PAUSING
  execution is waiting while the equipment is pausing but the piece of equipment has not yet reached a fully paused state.

- POWERING_DOWN

  execution is waiting while the equipment is powering down but has not fully reached a stopped state.

- POWERING_UP

  execution is waiting while the equipment is powering up and is not currently available to begin producing parts or products.

- RESUMING

  execution is waiting while the equipment is resuming the production cycle but has not yet resumed execution.

- SECONDARY_PROCESS

  execution is waiting while another process is completed before the execution can resume.

- TOOL_LOAD

  execution is waiting while a tool or tooling is being loaded.

- TOOL_UNLOAD

  execution is waiting while a tool or tooling is being unloaded.

5.2.125 Wire

identifier for the type of wire used as the cutting mechanism in Electrical Discharge Machining or similar processes.

5.2.126 WorkOffset

reference to offset variables for a work piece or part.

5.2.127 WorkOffsets

tabular representation of properties of each addressable work offset.

5.2.128 WorkholdingId

identifier for the current workholding or part clamp in use by a piece of equipment.

5.3 Sample Types

This section provides semantic information for `Sample` types.

5.3.1 Acceleration

positive rate of change of velocity.

The `units` of `Acceleration` **MUST** be `MILLIMETER/SECOND`^{2}.

The default `subType` of `Acceleration` is `ACTUAL`.

5.3.1.1 Subtypes of Acceleration

- `ACTUAL`

  measured or reported value of an observation.

- `COMMANDED`

  directive value including adjustments such as an offset or overrides.

- `PROGRAMMED`

  directive value without offsets and adjustments.

5.3.2 AccumulatedTime

accumulated time for an activity or event.

The `units` of `AccumulatedTime` **MUST** be `SECOND`.

5.3.3 <<deprecated>>Amperage

strength of electrical current.

DEPRECATED in *Version 1.6*. Replaced by `AMPERAGE_AC` and `AMPERAGE_DC`.

The `units` of `Amperage` **MUST** be `AMPERE`.

5.3.3.1 Subtypes of Amperage

- `ACTUAL`
  measured or reported value of an observation.
  DEPRECATED in *Version 1.6*.
- `ALTERNATING`
  measurement of alternating voltage or current.
  If not specified further in statistic, defaults to RMS voltage.
  DEPRECATED in *Version 1.6*.
- `DIRECT`
  measurement of DC current or voltage.
  DEPRECATED in *Version 1.6*.
- `TARGET`
  goal of the operation or process.
  DEPRECATED in *Version 1.6*.

5.3.4 AmperageAC

electrical current that reverses direction at regular short intervals.

The `units` of `AmperageAC` **MUST** be `AMPERE`.

A `subType` **MUST** always be specified.

5.3.4.1 Subtypes of AmperageAC

- ACTUAL
  measured or reported value of an observation.
- COMMANDED
  directive value including adjustments such as an offset or overrides.
- PROGRAMMED
  directive value without offsets and adjustments.

5.3.5 AmperageDC

electric current flowing in one direction only.

The units of AmperageDC **MUST** be AMPERE.

A subType **MUST** always be specified.

5.3.5.1 Subtypes of AmperageDC

- ACTUAL
  measured or reported value of an observation.
- COMMANDED
  directive value including adjustments such as an offset or overrides.
- PROGRAMMED
  directive value without offsets and adjustments.

5.3.6 Angle

angular position.

The units of Angle **MUST** be DEGREE.

5.3.6.1 Subtypes of Angle

- ACTUAL
  measured or reported value of an observation.
- COMMANDED
  directive value including adjustments such as an offset or overrides.

5.3.7 AngularAcceleration

positive rate of change of angular velocity.

The units of AngularAcceleration **MUST** be DEGREE/SECOND^2.

The default subType of AngularAcceleration is ACTUAL.

5.3.7.1 Subtypes of AngularAcceleration

- ACTUAL
  measured or reported value of an observation.
- COMMANDED
  directive value including adjustments such as an offset or overrides.
- PROGRAMMED
  directive value without offsets and adjustments.

5.3.8 AngularDeceleration

negative rate of change of angular velocity.

The units of AngularDeceleration **MUST** be DEGREE/SECOND^2.

The default subType of AngularDeceleration is ACTUAL.

5.3.8.1 Subtypes of AngularDeceleration

- ACTUAL
  measured or reported value of an observation.
- COMMANDED
  directive value including adjustments such as an offset or overrides.
- PROGRAMMED
  directive value without offsets and adjustments.

5.3.9 AngularVelocity

rate of change of angular position.

The units of AngularVelocity **MUST** be DEGREE/SECOND.

5.3.10 AssetUpdateRate

average rate of change of values for assets in the MTConnect streams.

The average is computed over a rolling window defined by the implementation.

The units of AssetUpdateRate **MUST** be COUNT/SECOND.

5.3.11 AxisFeedrate

feedrate of a linear axis.

The units of AxisFeedrate **MUST** be MILLIMETER/SECOND.

5.3.11.1 Subtypes of AxisFeedrate

- ACTUAL
  measured or reported value of an observation.

- COMMANDED

  directive value including adjustments such as an offset or overrides.

- JOG

  relating to momentary activation of a function or a movement.

  DEPRECATION WARNING: May be deprecated in the future.

- OVERRIDE

  operator's overridden value.

- PROGRAMMED

  directive value without offsets and adjustments.

- RAPID

  performing an operation faster or in less time than nominal rate.

5.3.12 BatteryCapacity

maximum rated charge a battery is capable of maintaining based on the battery discharging at a specified current over a specified time period.

The `units` of `BatteryCapacity` **MUST** be `COULOMB`.

The default `subType` of `BatteryCapacity` is `ACTUAL`.

5.3.12.1 Subtypes of BatteryCapacity

- ACTUAL

  measured or reported value of an observation.

- TARGET

  goal of the operation or process.

5.3.13 BatteryCharge

value of the battery's present capacity expressed as a percentage of the battery's maximum rated capacity.

The `units` of `BatteryCharge` **MUST** be `PERCENT`.

The default `subType` of `BatteryCharge` is `ACTUAL`.

5.3.13.1 Subtypes of BatteryCharge

- `ACTUAL`
  measured or reported value of an observation.
- `TARGET`
  goal of the operation or process.

5.3.14 CapacityFluid

fluid capacity of an object or container.

The `units` of `CapacityFluid` **MUST** be `MILLILITER`.

5.3.15 CapacitySpatial

geometric capacity of an object or container.

The `units` of `CapacitySpatial` **MUST** be `CUBIC_MILLIMETER`.

5.3.16 ChargeRate

value of the current being supplied to the `Component` for the purpose of charging.

The `units` of `ChargeRate` **MUST** be `AMPERE`.

The default `subType` of `ChargeRate` is `ACTUAL`.

5.3.16.1 Subtypes of ChargeRate

- `ACTUAL`
  measured or reported value of an observation.
- `TARGET`
  goal of the operation or process.

5.3.17 Concentration

percentage of one component within a mixture of components.

The `units` of `Concentration` **MUST** be `PERCENT`.

5.3.18 Conductivity

ability of a material to conduct electricity.

The `units` of `Conductivity` **MUST** be `SIEMENS/METER`.

5.3.19 CuttingSpeed

speed difference (relative velocity) between the cutting mechanism and the surface of the workpiece it is operating on.

The `units` of `CuttingSpeed` **MUST** be `MILLIMETER/SECOND`.

5.3.19.1 Subtypes of CuttingSpeed

- `ACTUAL`
  measured or reported value of an observation.
- `COMMANDED`
  directive value including adjustments such as an offset or overrides.

- PROGRAMMED
  directive value without offsets and adjustments.

5.3.20 Deceleration

negative rate of change of velocity.

The units of Deceleration **MUST** be MILLIMETER/SECOND^2.

The default subType of Deceleration is ACTUAL.

5.3.20.1 Subtypes of Deceleration

- ACTUAL
  measured or reported value of an observation.
- COMMANDED
  directive value including adjustments such as an offset or overrides.
- PROGRAMMED
  directive value without offsets and adjustments.

5.3.21 Density

volumetric mass of a material per unit volume of that material.

The units of Density **MUST** be MILLIGRAM/CUBIC_MILLIMETER.

5.3.22 DepositionAccelerationVolumetric

rate of change in spatial volume of material deposited in an additive manufacturing process.

The units of DepositionAccelerationVolumetric **MUST** be CUBIC_MILLIMETER/SECOND^2

5.3.22.1 Subtypes of DepositionAccelerationVolumetric

- ACTUAL
  measured or reported value of an observation.
- COMMANDED
  directive value including adjustments such as an offset or overrides.

5.3.23 DepositionDensity

density of the material deposited in an additive manufacturing process per unit of volume.

The units of DepositionDensity **MUST** be MILLIGRAM/CUBIC_MILLIMETER.

5.3.23.1 Subtypes of DepositionDensity

- ACTUAL
  measured or reported value of an observation.
- COMMANDED
  directive value including adjustments such as an offset or overrides.

5.3.24 DepositionMass

mass of the material deposited in an additive manufacturing process.

The units of DepositionMass **MUST** be MILLIGRAM.

5.3.24.1 Subtypes of DepositionMass

- ACTUAL
  measured or reported value of an observation.
- COMMANDED
  directive value including adjustments such as an offset or overrides.

5.3.25 DepositionRateVolumetric

rate at which a spatial volume of material is deposited in an additive manufacturing process.

The `units` of `DepositionRateVolumetric` **MUST** be `CUBIC_MILLIMETER/SECOND`.

5.3.25.1 Subtypes of DepositionRateVolumetric

- `ACTUAL`
  measured or reported value of an observation.
- `COMMANDED`
  directive value including adjustments such as an offset or overrides.

5.3.26 DepositionVolume

spatial volume of material to be deposited in an additive manufacturing process.

The `units` of `DepositionVolume` **MUST** be `CUBIC_MILLIMETER`.

5.3.26.1 Subtypes of DepositionVolume

- `ACTUAL`
  measured or reported value of an observation.
- `COMMANDED`
  directive value including adjustments such as an offset or overrides.

5.3.27 DewPoint

temperature at which moisture begins to condense, corresponding to saturation for a given absolute humidity.

The `units` of `DewPoint` **MUST** be `CELSIUS`.

5.3.28 Diameter

dimension of a diameter.

The `units` of `Diameter` **MUST** be `MILLIMETER`.

5.3.29 DischargeRate

value of current being drawn from the `Component`.

The `units` of `DischargeRate` **MUST** be `AMPERE`.

The default `subType` of `DischargeRate` is `ACTUAL`.

5.3.29.1 Subtypes of DischargeRate

- `ACTUAL`
  measured or reported value of an observation.
- `TARGET`
  goal of the operation or process.

5.3.30 Displacement

change in position of an object.

The `units` of `Displacement` **MUST** be `MILLIMETER`.

5.3.31 DisplacementAngular

absolute value of the change in angular position around a vector

> Note: The displacement vector **MAY** be defined by the motion of the owning `Component`.

The `units` of `DisplacementAngular` **MUST** be `DEGREE`.

5.3.32 DisplacementLinear

absolute value of the change in position along a vector.

Note: The displacement vector **MAY** be defined by the motion of the owning `Component`.

The `units` of `DisplacementLinear` **MUST** be `MILLIMETER`.

5.3.33 ElectricalEnergy

`Wattage` used or generated by a component over an interval of time.

The `units` of `ElectricalEnergy` **MUST** be `WATT_SECOND`.

5.3.34 EquipmentTimer

amount of time a piece of equipment or a sub-part of a piece of equipment has performed specific activities.

The `units` of `EquipmentTimer` **MUST** be `SECOND`.

A `subType` **MUST** always be specified.

5.3.34.1 Subtypes of EquipmentTimer

- `DELAY`

  elapsed time of a temporary halt of action.

- `LOADED`

  time that the sub-parts of a piece of equipment are under load.

  Example: For traditional machine tools, this is a measurement of the time that the cutting tool is assumed to be engaged with the part.

- `OPERATING`

  time that the major sub-parts of a piece of equipment are powered or performing any activity whether producing a part or product or not.

  Example: For traditional machine tools, this includes `WORKING`, plus idle time.

- `POWERED`

  time that primary power is applied to the piece of equipment and, as a minimum, the controller or logic portion of the piece of equipment is powered and functioning or components that are required to remain on are powered.

  Example: Heaters for an extrusion machine that are required to be powered even when the equipment is turned off.

- `WORKING`

  time that a piece of equipment is performing any activity the equipment is active and performing a function under load or not.

  Example: For traditional machine tools, this includes `LOADED`, plus rapid moves, tool changes, etc.

5.3.35 FillLevel

amount of a substance remaining compared to the planned maximum amount of that substance.

The `units` of `FillLevel` **MUST** be `PERCENT`.

5.3.36 Flow

rate of flow of a fluid.

The `units` of `Flow` **MUST** be `LITER/SECOND`.

5.3.37 FollowingError

difference between actual and commanded position at any specific point in time during a motion.

The `units` of `FollowingError` **MUST** be `COUNT`.

5.3.37.1 Subtypes of FollowingError

- ACTUAL
  measured or reported value of an observation.

5.3.38 FollowingErrorAngular

angular difference between the commanded encoder/resolver position and the actual encoder/resolver position at any specified point in time during a motion.

The units of FollowingErrorAngular **MUST** be DEGREE.

5.3.38.1 Subtypes of FollowingErrorAngular

- ACTUAL
  measured or reported value of an observation.

5.3.39 FollowingErrorLinear

difference between the commanded encoder/resolver position and the actual encoder/resolver position at any specified point in time during a motion.

The units of FollowingErrorLinear **MUST** be MILLIMETER.

5.3.39.1 Subtypes of FollowingErrorLinear

- ACTUAL
  measured or reported value of an observation.

5.3.40 Frequency

number of occurrences of a repeating event per unit time.

The units of Frequency **MUST** be HERTZ.

5.3.41 <<deprecated>>GlobalPosition

position in three-dimensional space.

DEPRECATED in Version 1.1.

The `units` of `GlobalPosition` **MUST** be `MILLIMETER`.

5.3.41.1 Subtypes of GlobalPosition

- `ACTUAL`
  measured or reported value of an observation.
- `COMMANDED`
  directive value including adjustments such as an offset or overrides.

5.3.42 GravitationalAcceleration

acceleration relative to Earth's gravity of 9.80665 `METER/SECOND`^{2}.

The `units` of `GravitationalAcceleration` **MUST** be `GRAVITATIONAL_ACCELERATION`.

5.3.43 GravitationalForce

force relative to earth's gravity.

Note: $Mass \times GravitationalAcceleration$

The `units` of `GravitationalForce` **MUST** be `GRAVITATIONAL_FORCE`.

5.3.44 HumidityAbsolute

amount of water vapor expressed in grams per cubic meter.

The `units` of `HumidityAbsolute` **MUST** be `GRAM/CUBIC_METER`.

5.3.44.1 Subtypes of HumidityAbsolute

- ACTUAL
  measured or reported value of an observation.
- COMMANDED
  directive value including adjustments such as an offset or overrides.

5.3.45 HumidityRelative

amount of water vapor present expressed as a percent to reach saturation at the same temperature.

The units of HumidityRelative **MUST** be PERCENT.

5.3.45.1 Subtypes of HumidityRelative

- ACTUAL
  measured or reported value of an observation.
- COMMANDED
  directive value including adjustments such as an offset or overrides.

5.3.46 HumiditySpecific

ratio of the water vapor present over the total weight of the water vapor and air present expressed as a percent.

The units of HumiditySpecific **MUST** be PERCENT.

5.3.46.1 Subtypes of HumiditySpecific

- ACTUAL
  measured or reported value of an observation.

- COMMANDED
  directive value including adjustments such as an offset or overrides.

5.3.47 Length

length of an object.

The units of Length **MUST** be MILLIMETER.

5.3.47.1 Subtypes of Length

- REMAINING
  remaining total length of an object.
- STANDARD
  standard or original length of an object.
- USEABLE
  remaining usable length of an object.

5.3.48 <<deprecated>>Level

level of a resource.

DEPRECATED in *Version 1.2*. See FILL_LEVEL.

The units of Level **MUST** be PERCENT.

5.3.49 LinearForce

force applied to a mass in one direction only.

The units of LinearForce **MUST** be NEWTON.

5.3.50 Load

actual versus the standard rating of a piece of equipment.

The `units` of `Load` **MUST** be `PERCENT`.

5.3.51 Mass

mass of an object(s) or an amount of material.

The `units` of `Mass` **MUST** be `KILOGRAM`.

5.3.52 ObservationUpdateRate

average rate of change of values for data items in the MTConnect streams. The average is computed over a rolling window defined by the implementation.

The `units` of `ObservationUpdateRate` **MUST** be `COUNT/SECOND`.

5.3.53 Openness

percentage open where 100% is fully open and 0% is fully closed.

The `units` of `Openness` **MUST** be `PERCENT`.

5.3.54 Orientation

angular position of a plane or vector relative to a *cartesian coordinate system*

The `units` of `Orientation` **MUST** be `DEGREE_3D`.

The value of `Orientation` **MUST** be a list of `float` of size `0..3`.

5.3.54.1 Subtypes of Orientation

- ACTUAL
  measured or reported value of an observation.
- COMMANDED
  directive value including adjustments such as an offset or overrides.

5.3.55 PH

acidity or alkalinity of a solution.

The units of PH **MUST** be PH.

5.3.56 PathFeedrate

feedrate for the axes, or a single axis, associated with a Path component.

The units of PathFeedrate **MUST** be MILLIMETER/SECOND.

5.3.56.1 Subtypes of PathFeedrate

- ACTUAL
  measured or reported value of an observation.
- COMMANDED
  directive value including adjustments such as an offset or overrides.
- JOG
  relating to momentary activation of a function or a movement.
  DEPRECATION WARNING: May be deprecated in the future.
- OVERRIDE
  operator's overridden value.
  DEPRECATED in *Version 1.3*.
- PROGRAMMED
  directive value without offsets and adjustments.

- `RAPID`
  performing an operation faster or in less time than nominal rate.

5.3.57 PathFeedratePerRevolution

feedrate for the axes, or a single axis.

The `units` of `PathFeedratePerRevolution` **MUST** be `MILLIMETER/REVOLUTION`.

5.3.57.1 Subtypes of PathFeedratePerRevolution

- `ACTUAL`
  measured or reported value of an observation.
- `COMMANDED`
  directive value including adjustments such as an offset or overrides.
- `PROGRAMMED`
  directive value without offsets and adjustments.

5.3.58 PathPosition

position of a control point associated with a `Controller` or a `Path`.

The `units` of `PathPosition` **MUST** be `MILLIMETER_3D`.

The value of `PathPosition` **MUST** be a list of `float` of size `0..3`.

5.3.58.1 Subtypes of PathPosition

- `ACTUAL`
  measured or reported value of an observation.
- `COMMANDED`
  directive value including adjustments such as an offset or overrides.

- PROBE
  position provided by a measurement probe.
  DEPRECATION WARNING: May be deprecated in the future.
- TARGET
  goal of the operation or process.

5.3.59 Position

point along an axis in a *cartesian coordinate system*.

The units of Position **MUST** be MILLIMETER.

5.3.59.1 Subtypes of Position

- ACTUAL
  measured or reported value of an observation.
- COMMANDED
  directive value including adjustments such as an offset or overrides.
- PROGRAMMED
  directive value without offsets and adjustments.
- TARGET
  goal of the operation or process.

5.3.60 PositionCartesian

point in a *cartesian coordinate system*.

The units of PositionCartesian **MUST** be MILLIMETER_3D.

The value of PositionCartesian **MUST** be a list of float of size 0..3.

5.3.61 PowerFactor

ratio of real power flowing to a load to the apparent power in that AC circuit.

The `units` of `PowerFactor` **MUST** be `PERCENT`.

5.3.62 Pressure

force per unit area measured relative to atmospheric pressure.

Commonly referred to as gauge pressure.

The `units` of `Pressure` **MUST** be `PASCAL`.

5.3.63 PressureAbsolute

The force per unit area measured relative to a vacuum.

The `units` of `PressureAbsolute` **MUST** be `PASCAL`.

5.3.64 PressurizationRate

change of pressure per unit time.

The `units` of `PressurizationRate` **MUST** be `PASCAL/SECOND`.

5.3.64.1 Subtypes of PressurizationRate

- `ACTUAL`
  measured or reported value of an observation.
- `COMMANDED`
  directive value including adjustments such as an offset or overrides.
- `PROGRAMMED`
  directive value without offsets and adjustments.

5.3.65 ProcessTimer

amount of time a piece of equipment has performed different types of activities associated with the process being performed at that piece of equipment.

The `units` of `ProcessTimer` **MUST** be `SECOND`.

A `subType` **MUST** always be specified.

5.3.65.1 Subtypes of ProcessTimer

- `DELAY`

  elapsed time of a temporary halt of action.

- `PROCESS`

  time from the beginning of production of a part or product on a piece of equipment until the time that production is complete for that part or product on that piece of equipment.

  This includes the time that the piece of equipment is running, producing parts or products, or in the process of producing parts.

5.3.66 Resistance

degree to which a substance opposes the passage of an electric current.

The `units` of `Resistance` **MUST** be `OHM`.

5.3.67 RotaryVelocity

rotational speed of a rotary axis.

The `units` of `RotaryVelocity` **MUST** be `REVOLUTION/MINUTE`.

5.3.67.1 Subtypes of RotaryVelocity

- ACTUAL
  measured or reported value of an observation.
- COMMANDED
  directive value including adjustments such as an offset or overrides.
- OVERRIDE
  The operators overridden value.
- PROGRAMMED
  directive value without offsets and adjustments.

5.3.68 SettlingError

difference between actual and commanded position at the end of a motion.

The `units` of `SettlingError` **MUST** be `COUNT`.

5.3.68.1 Subtypes of SettlingError

- ACTUAL
  measured or reported value of an observation.

5.3.69 SettlingErrorAngular

angular difference between the commanded encoder/resolver position, and the actual encoder/resolver position when motion is complete.

The `units` of `SettlingErrorAngular` **MUST** be `DEGREE`.

5.3.69.1 Subtypes of SettlingErrorAngular

- ACTUAL
  measured or reported value of an observation.

5.3.70 SettlingErrorLinear

difference between the commanded encoder/resolver position, and the actual encoder/resolver position when motion is complete.

The `units` of `SettlingErrorLinear` **MUST** be `MILLIMETER`.

5.3.70.1 Subtypes of SettlingErrorLinear

- `ACTUAL`
  measured or reported value of an observation.

5.3.71 SoundLevel

sound level or sound pressure level relative to atmospheric pressure.

The `units` of `SoundLevel` **MUST** be `DECIBEL`.

The default `subType` of `SoundLevel` is `A_SCALE`.

5.3.71.1 Subtypes of SoundLevel

- `A_SCALE`
  A Scale weighting factor. This is the default weighting factor if no factor is specified
- `B_SCALE`
  B Scale weighting factor
- `C_SCALE`
  C Scale weighting factor
- `D_SCALE`
  D Scale weighting factor
- `NO_SCALE`
  No weighting factor on the frequency scale

5.3.72 <<deprecated>>SpindleSpeed

rotational speed of the rotary axis.

DEPRECATED in *Version 1.2*. Replaced by `ROTARY_VELOCITY`.

The `units` of `SpindleSpeed` **MUST** be `REVOLUTION/MINUTE`.

5.3.72.1 Subtypes of SpindleSpeed

- `ACTUAL`

  measured or reported value of an observation.

  DEPRECATED in *Version 1.3*.

- `COMMANDED`

  directive value including adjustments such as an offset or overrides.

  DEPRECATED in *Version 1.3*.

- `OVERRIDE`

  operator's overridden value.

  DEPRECATED in *Version 1.3*.

5.3.73 Strain

amount of deformation per unit length of an object when a load is applied.

The `units` of `Strain` **MUST** be `PERCENT`.

5.3.74 Temperature

degree of hotness or coldness measured on a definite scale.

The `units` of `Temperature` **MUST** be `CELSIUS`.

5.3.75 Tension

force that stretches or elongates an object.

The `units` of `Tension` **MUST** be `NEWTON`.

5.3.76 Tilt

angular displacement.

The `units` of `Tilt` **MUST** be `MICRO_RADIAN`.

5.3.77 Torque

turning force exerted on an object or by an object.

The `units` of `Torque` **MUST** be `NEWTON_METER`.

5.3.78 Velocity

rate of change of position of a `Component`.

The `units` of `Velocity` **MUST** be `MILLIMETER/SECOND`.

5.3.79 Viscosity

fluid's resistance to flow.

The `units` of `Viscosity` **MUST** be `PASCAL_SECOND`.

5.3.80 VoltAmpere

apparent power in an electrical circuit, equal to the product of root-mean-square (RMS) voltage and RMS current (commonly referred to as VA).

The `units` of `VoltAmpere` **MUST** be `VOLT_AMPERE`.

5.3.81 VoltAmpereReactive

reactive power in an AC electrical circuit (commonly referred to as VAR).

The `units` of `VoltAmpereReactive` **MUST** be `VOLT_AMPERE_REACTIVE`.

5.3.82 <<deprecated>>Voltage

electrical potential between two points.

DEPRECATED in *Version 1.6*. Replaced by `VOLTAGE_AC` and `VOLTAGE_DC`.

The `units` of `Voltage` **MUST** be `VOLT`.

5.3.82.1 Subtypes of Voltage

- `ACTUAL`
  measured or reported value of an observation.
  DEPRECATED in *Version 1.6*.
- `ALTERNATING`
  alternating voltage or current.
  If not specified further in statistic, defaults to RMS voltage.
  DEPRECATED in *Version 1.6*.
- `DIRECT`
  DC current or voltage.
  DEPRECATED in *Version 1.6*.
- `TARGET`
  goal of the operation or process.
  DEPRECATED in *Version 1.6*.

5.3.83 VoltageAC

electrical potential between two points in an electrical circuit in which the current periodically reverses direction.

The `units` of `VoltageAC` **MUST** be `VOLT`.

A `subType` **MUST** always be specified.

5.3.83.1 Subtypes of VoltageAC

- `ACTUAL`
  measured or reported value of an observation.
- `COMMANDED`
  directive value including adjustments such as an offset or overrides.
- `PROGRAMMED`
  directive value without offsets and adjustments.

5.3.84 VoltageDC

electrical potential between two points in an electrical circuit in which the current is unidirectional.

The `units` of `VoltageDC` **MUST** be `VOLT`.

A `subType` **MUST** always be specified.

5.3.84.1 Subtypes of VoltageDC

- `ACTUAL`
  measured or reported value of an observation.
- `COMMANDED`
  directive value including adjustments such as an offset or overrides.
- `PROGRAMMED`
  directive value without offsets and adjustments.

5.3.85 VolumeFluid

fluid volume of an object or container.

The `units` of `VolumeFluid` **MUST** be `MILLILITER`.

5.3.85.1 Subtypes of VolumeFluid

- `ACTUAL`
  measured or reported value of an observation.
- `CONSUMED`
  reported or measured value of the amount used in the manufacturing process.
- `ENDED`
  boundary when an activity or an event terminates.
- `PART`
  reported or measured value of amount included in the *part*.
- `START`
  boundary when an activity or an event commences.
- `WASTE`
  reported or measured value of the amount discarded.

5.3.86 VolumeSpatial

geometric volume of an object or container.

The `units` of `VolumeSpatial` **MUST** be `CUBIC_MILLIMETER`.

5.3.86.1 Subtypes of VolumeSpatial

- `ACTUAL`
  measured or reported value of an observation.

- CONSUMED
  reported or measured value of the amount used in the manufacturing process.
- ENDED
  boundary when an activity or an event terminates.
- PART
  reported or measured value of amount included in the *part*.
- START
  boundary when an activity or an event commences.
- WASTE
  reported or measured value of the amount discarded

5.3.87 Wattage

power flowing through or dissipated by an electrical circuit or piece of equipment.

The `units` of `Wattage` **MUST** be `WATT`.

5.3.87.1 Subtypes of Wattage

- ACTUAL
  measured or reported value of an observation.
- TARGET
  goal of the operation or process.

5.3.88 XDimension

dimension of an entity relative to the X direction of the referenced coordinate system.

The `units` of `XDimension` **MUST** be `MILLIMETER`.

5.3.89 YDimension

dimension of an entity relative to the Y direction of the referenced coordinate system.

The `units` of `YDimension` **MUST** be `MILLIMETER`.

5.3.90 ZDimension

dimension of an entity relative to the Z direction of the referenced coordinate system.

The `units` of `ZDimension` **MUST** be `MILLIMETER`.

6 Profile

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

6.1 DataTypes

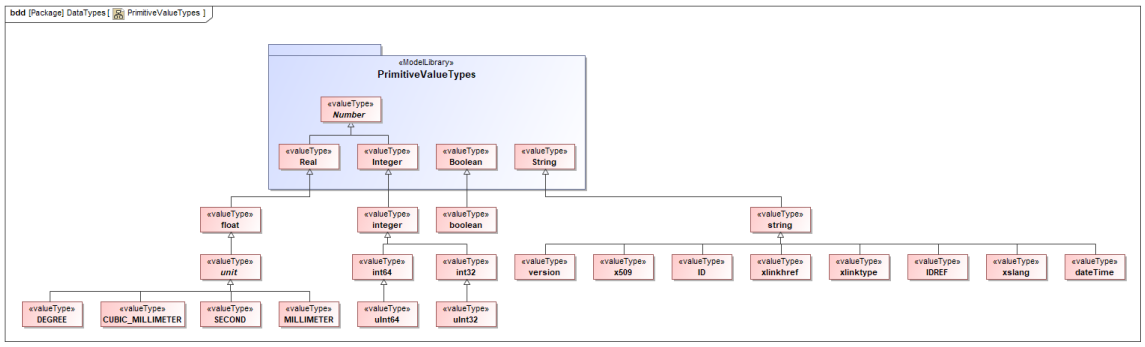

Figure 13: DataTypes

6.1.1 boolean

primitive type.

6.1.2 ID

string that represents an identifier (ID).

6.1.3 string

primitive type.

6.1.4 float

primitive type.

6.1.5 datetime

string that represents timestamp in ISO 8601 format.

6.1.6 integer

primitive type.

6.1.7 xlinktype

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

6.1.8 xslang

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

6.1.9 SECOND

float that represents time in seconds.

6.1.10 IDREF

string that represents a reference to an `ID`.

6.1.11 xlinkhref

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

6.1.12 x509

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

6.1.13 int32

32-bit integer.

6.1.14 int64

64-bit integer.

6.1.15 version

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

6.1.16 uint32

32-bit unsigned integer.

6.1.17 uint64

64-bit unsigned integer.

6.1.18 binary

base-2 numeral system or binary numeral system represented by two digits: “0” and “1”.

6.1.19 double

primitive type.

6.2 Stereotypes

6.2.1 organizer

element that *organizes* other elements of a type.

6.2.2 deprecated

element that has been deprecated.

6.2.3 extensible

enumeration that can be extended.

6.2.4 informative

element that is descriptive and non-normative.

6.2.5 valueType

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

6.2.6 normative

element that has been added to the standard.

6.2.7 observes

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

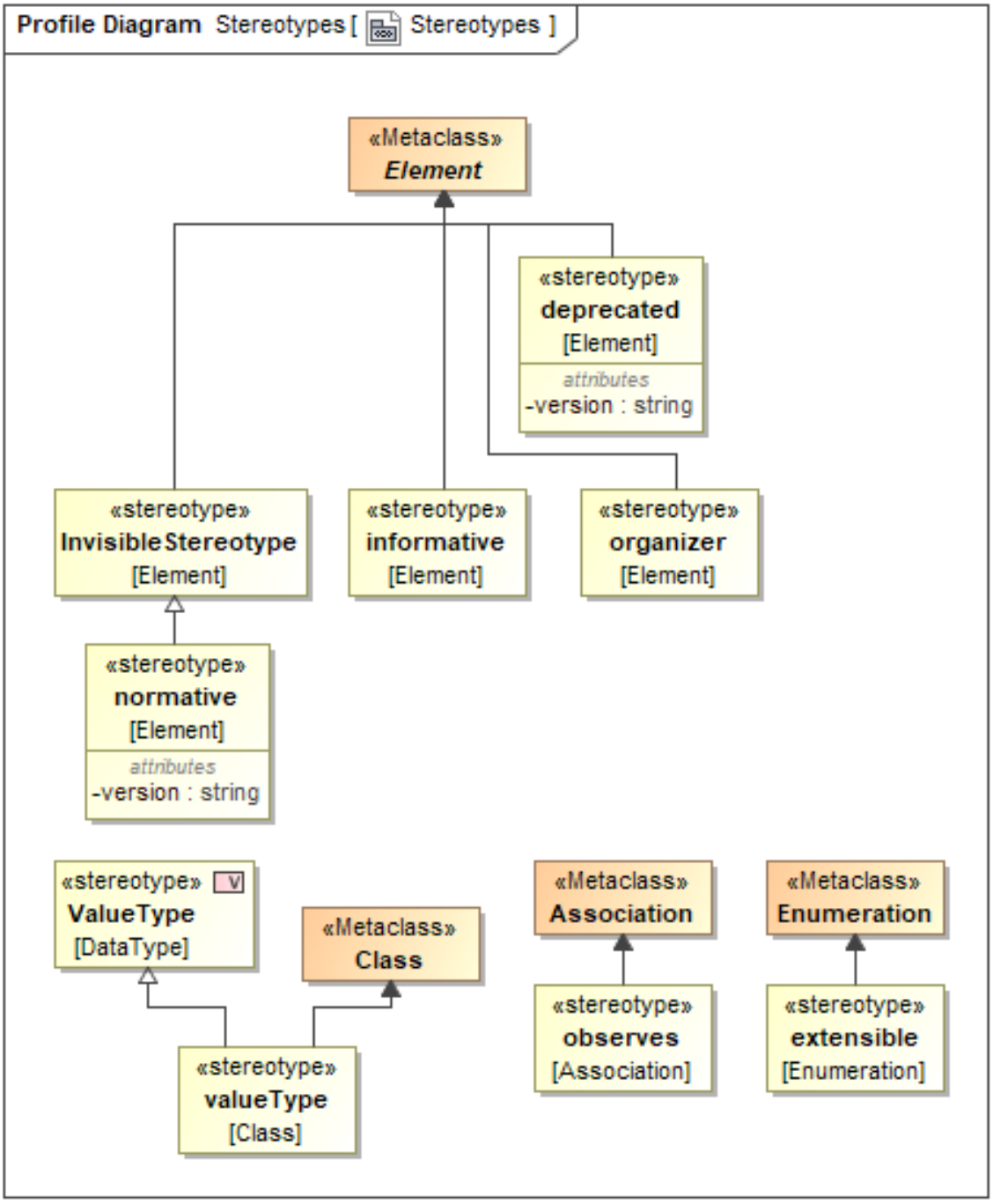

Figure 14: Stereotypes

Appendices

A Bibliography

Engineering Industries Association. EIA Standard - EIA-274-D, Interchangeable Variable, Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically Controlled Machines. Washington, D.C. 1979.

ISO TC 184/SC4/WG3 N1089. ISO/DIS 10303-238: Industrial automation systems and integration Product data representation and exchange Part 238: Application Protocols: Application interpreted model for computerized numerical controllers. Geneva, Switzerland, 2004.

International Organization for Standardization. ISO 14649: Industrial automation systems and integration – Physical device control – Data model for computerized numerical controllers – Part 10: General process data. Geneva, Switzerland, 2004.

International Organization for Standardization. ISO 14649: Industrial automation systems and integration – Physical device control – Data model for computerized numerical controllers – Part 11: Process data for milling. Geneva, Switzerland, 2000.

International Organization for Standardization. ISO 6983/1 – Numerical Control of machines – Program format and definition of address words – Part 1: Data format for positioning, line and contouring control systems. Geneva, Switzerland, 1982.

Electronic Industries Association. ANSI/EIA-494-B-1992, 32 Bit Binary CL (BCL) and 7 Bit ASCII CL (ACL) Exchange Input Format for Numerically Controlled Machines. Washington, D.C. 1992.

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

International Organization for Standardization. ISO 10303-11: 1994, Industrial automation systems and integration Product data representation and exchange Part 11: Description methods: The EXPRESS language reference manual. Geneva, Switzerland, 1994.

International Organization for Standardization. ISO 10303-21: 1996, Industrial automation systems and integration – Product data representation and exchange – Part 21: Implementation methods: Clear text encoding of the exchange structure. Geneva, Switzerland, 1996.

H.L. Horton, F.D. Jones, and E. Oberg. Machinery's Handbook. Industrial Press, Inc.

New York, 1984.

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

ASME B5.57: Methods for Performance Evaluation of Computer Numerically Controlled Lathes and Turning Centers, 1998.

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

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

IEEE STD 1451.0-2007, Standard for a Smart Transducer Interface for Sensors and Actuators – Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH99684, October 5, 2007.

IEEE STD 1451.4-1994, Standard for a Smart Transducer Interface for Sensors and Actuators – Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, N.Y. 10016, SH95225, December 15, 2004.

B XML Schema Diagrams

B.1 Observations Schema Diagrams

See `Streams` element in `MTConnectStreams` schema.

See `DeviceStream` element in `MTConnectStreams` schema.

See `ComponentStream` element in `MTConnectStreams` schema.

See `Sample` element in `MTConnectStreams` schema.

See `Event` element in `MTConnectStreams` schema.

See `Condition` element in `MTConnectStreams` schema.

B.2 Representation Schema Diagrams

See `AbsTimeSeries` element in `MTConnectStreams` schema.

See `PartCountDiscrete` element in `MTConnectStreams` schema.

See `VariableDataSet` element in `MTConnectStreams` schema.

See `Entry` element in `MTConnectStreams` schema.

See `WorkOffsetTable` element in `MTConnectStreams` schema.

C XML Examples

C.1 DeviceStream Example

Example 1: DeviceStream Example

```
1 <Streams>
2   <DeviceStream name="M12346" uuid="M8010W4194N">
3     <ComponentStream component="Device" name="M12346" componentId="
        d1">
4       <Events>
5         <Availability dataItemId="avail" sequence="156" timestamp="
      2021-10-01T14:26:38.668505Z">AVAILABLE</Availability>
```

```
3708  6          <AssetChanged assetType="CuttingTool" dataItemId="d1\
3709                textunderscore␣asset\textunderscore␣chg" sequence="75570"
3710                 timestamp="2021-10-07T05:08:53.870206Z">M8010W4194N1
3711                .120</AssetChanged>
3712  7          <AssetRemoved assetType="CuttingTool" dataItemId="d1\
3713                textunderscore␣asset\textunderscore␣rem" sequence="140"
3714                timestamp="2021-10-01T11:40:08.101461Z">UNAVAILABLE</
3715                AssetRemoved>
3716  8        </Events>
3717  9       </ComponentStream>
3718  10      <ComponentStream component="Controller" name="controller"
3719             componentId="cont">
3720  11       <Events>
3721  12         <EmergencyStop dataItemId="estop" sequence="159" timestamp="
3722           2021-10-01T14:26:38.66869Z">ARMED</EmergencyStop>
3723  13       </Events>
3724  14       <Samples>
3725  15         <AccumulatedTime dataItemId="cut\textunderscore␣time"
3726           sequence="75437" timestamp="2021-10-07T05:08:28.221704Z">
3727           1763070.0</AccumulatedTime>
3728  16       </Samples>
3729  17       <Condition>
3730  18         <Unavailable dataItemId="cont\textunderscore␣system"
3731           sequence="72" timestamp="2021-10-11T21:04:03.251999Z" type="
3732           SYSTEM"/>
3733  19         <Warning dataItemId="cont\textunderscore␣system" nativecode=
3734                "313" nativeSeverity="50" sequence="75573" timestamp="
3735                2021-10-07T05:08:58.518317Z" type="LOGIC\textunderscore␣
3736                PROGRAM">PALLET ARM DOWN RS. MALF.</Warning>
3737  20       </Condition>
3738  21      </ComponentStream>
3739  22      <ComponentStream component="Path" name="path" componentId="path1
3740             ">
3741  23       <Events>
3742  24         <Execution dataItemId="execution" name="execution" sequence=
3743           "222623" timestamp="2021-10-12T06:04:32.761198Z">INTERRUPTED</
3744           Execution>
3745  25         <VariableDataSet count="2" dataItemId="cvars" sequence="
3746                126513" timestamp="2021-10-12T03:57:31.106559Z">
3747  26           <Entry key="100">66.3314</Entry>
3748  27           <Entry key="101">167.2</Entry>
3749  28         </VariableDataSet>
3750  29         <WorkOffsetTable count="2" dataItemId="woffset" sequence="
3751                222101" timestamp="2021-10-12T06:04:11.990531Z">
3752  30           <Entry key="G53.1">
3753  31             <Cell key="X">1</Cell>
3754  32             <Cell key="Y">2</Cell>
3755  33             <Cell key="Z">3</Cell>
3756  34           </Entry>
3757  35           <Entry key="G53.2">
3758  36             <Cell key="X">4</Cell>
```

```
3759 37             <Cell key="Y">5</Cell>
3760 38             <Cell key="Z">6</Cell>
3761 39           </Entry>
3762 40         </WorkOffsetTable>
3763 41       </Events>
3764 42       <Samples>
3765 43         <CuttingSpeed dataItemId="cspeed" sequence="112" timestamp="
3766       2021-10-07T05:08:28.221704Z" subType="ACTUAL">UNAVAILABLE</
3767       CuttingSpeed>
3768 44       </Samples>
3769 45       <Condition>
3770 46         <Normal dataItemId="path\textunderscore␣system" sequence="
3771       153" timestamp="2021-10-11T21:04:03.262845Z" type="SYSTEM"/>
3772 47       </Condition>
3773 48     </ComponentStream>
3774 49   </DeviceStream>
3775 50 </Streams>
```

3776 C.2 Observations made for DataItem Example

Example 2: MTConnectDevices Response Document

```
3777  1 <Components>
3778  2   <Controller id="cont" name="controller">
3779  3       <DataItems>
3780  4       <DataItem category="EVENT" id="estop" name="estop" type="
3781       EMERGENCY\textunderscore␣STOP"/>
3782  5     </DataItems>
3783  6   </Controller>
3784  7 </Components>
```

Example 3: MTConnectStreams Response Document

```
3785  1 <ComponentStream component="Controller" name="controller"
3786       componentId="cont">
3787  2   <Events>
3788  3     <EmergencyStop dataItemId="estop" sequence="159" timestamp="
3789       2021-10-01T14:26:38.66869Z">ARMED</EmergencyStop>
3790  4   </Events>
3791  5 </ComponentStream>
```

3792 C.3 Sample Example

Example 4: Sample Example

```
3793  1 <Samples>
```

```
    <AccumulatedTime dataItemId="cut\textunderscore␣time" sequence="
     75437" timestamp="2021-10-07T05:08:28.221704Z">1763070.0</
     AccumulatedTime>
    <CuttingSpeed dataItemId="cspeed" sequence="112" timestamp="
       2021-10-07T05:08:28.221704Z" subType="ACTUAL">UNAVAILABLE</
       CuttingSpeed>
</Samples>
```

C.4 Event Example

Example 5: Event Example

```
<Events>
  <Availability dataItemId="avail" sequence="156" timestamp="
   2021-10-01T14:26:38.668505Z">AVAILABLE</Availability>
  <AssetRemoved assetType="CuttingTool" dataItemId="d1\
     textunderscore␣asset\textunderscore␣rem" sequence="140"
     timestamp="2021-10-01T11:40:08.101461Z">UNAVAILABLE</
     AssetRemoved>
</Events>
```

C.5 Condition Example

`Condition` types are represented differently in XML when compared to `Sample` and `Event` types. The element name is the condition state of the `Condition` type in Pascal Case. The name of the `Condition` type is represented by the attribute `type`.

If the condition state is unavailable then the element name is represented by `Unavailable`.

Example 6: Condition Example

```
<Condition>
  <Unavailable dataItemId="cont\textunderscore␣system" sequence="72"
    timestamp="2021-10-11T21:04:03.251999Z" type="SYSTEM"/>
  <Normal dataItemId="path\textunderscore␣system" sequence="153"
     timestamp="2021-10-11T21:04:03.262845Z" type="SYSTEM"/>
  <Warning dataItemId="cont\textunderscore␣system" nativecode="313"
     nativeSeverity="50" sequence="75573" timestamp="2021-10-07T05
     :08:58.518317Z" type="LOGIC\textunderscore␣PROGRAM">PALLET ARM
     DOWN RS. MALF.</Warning>
</Condition>
```

C.6 DataSet Example

Example 7: DataSet Example

```
<Events>
  <VariableDataSet count="2" dataItemId="cvars" sequence="126513"
   timestamp="2021-10-12T03:57:31.106559Z">
    <Entry key="100">66.3314</Entry>
    <Entry key="101">167.2</Entry>
  </VariableDataSet>
</Events>
```

C.7 Table Example

Example 8: Table Example

```
<Events>
  <WorkOffsetTable count="2" dataItemId="woffset" sequence="222101"
   timestamp="2021-10-12T06:04:11.990531Z">
    <Entry key="G53.1">
      <Cell key="X">1</Cell>
      <Cell key="Y">2</Cell>
      <Cell key="Z">3</Cell>
    </Entry>
    <Entry key="G53.2">
      <Cell key="X">4</Cell>
      <Cell key="Y">5</Cell>
      <Cell key="Z">6</Cell>
    </Entry>
  </WorkOffsetTable>
</Events>
```