

# Getting Started With MTCConnect®: Writing Client Applications

## *A High-Level Architecture Overview and Tutorial For Software Application Developers*

By Dave Edstrom  
CEO/CTO [Virtual Photons Electrons](#)

This white paper was written for software developers who are interested in writing MTCConnect-enabled applications. The primary audience is software developers *who are not in the manufacturing arena* and are interested in creating MTCConnect client applications. The reason for this focus is to expand the number of software developers for MTCConnect and hopefully, the number of interesting applications that are created for MTCConnect enabled manufacturing equipment and devices.

If you are a software developer *and* in manufacturing, this will be a helpful primer to learn about MTCConnect. No matter what your experience in manufacturing, the next step after reading this white paper will be to become more familiar with MTCConnect by reading the MTCConnect specification – Parts 1, 2, 3 and 4. If you are neither a software developer, nor in manufacturing, the first part of this document will give you a high-level architecture overview, but there are other documents at [MTCConnect.org](#) you would find more appropriate if your interests are more business related.

This white paper is part of the “Getting Started With MTCConnect” series of documents. Please note this white paper will be receiving additional updates and clarifications. The current MTCConnect spec is version 1.2. The release date of this white paper is July 17th, 2013. Most of the graphics in this presentation come from slides created by me or Will Sobel, Chief Architect of MTCConnect and President of System Insights.

Any comments or suggestions on this document should  
be sent to:

[DavidAllenEdstrom@gmail.com](mailto:DavidAllenEdstrom@gmail.com)

## Table of Contents

MTConnect® Specifications or Materials.....	3
Purpose and Target Audience .....	5
What Problem Does MTConnect Solve?.....	<b>Error! Bookmark not defined.</b>
<b>Turner’s Five Laws of Manufacturing</b> .....	<b>11</b>
MTConnect Basics.....	12
<b>What is MTConnect?</b> .....	<b>12</b>
<b>What Isn’t MTConnect?</b> .....	<b>12</b>
<b>MTConnect Licensing</b> .....	<b>12</b>
<b>Selling MTConnect</b> .....	<b>13</b>
Three Key Pillars of MTConnect.....	13
<b>Adapter</b> .....	<b>13</b>
<b>Agent</b> .....	<b>15</b>
<b>Application</b> .....	<b>16</b>
Three Types of MTConnect Devices and Discovery.....	17
MTConnect Standard Overview.....	17
<b>Programmatic Conceptual Terms</b> .....	<b>19</b>
MTConnect’s Powerful Namespace – XML Schema Definition aka Data Dictionary..	19
The Simplest MTConnect Clients – Your Browser & Spreadsheet .....	20
Fault Tolerance and Persistence of Data .....	21
<b>MTConnect Sequence Number</b> .....	<b>21</b>
MTConnect Responses.....	23
<b>Devices</b> .....	<b>23</b>
<b>Streams</b> .....	<b>24</b>
<b>Assets</b> .....	<b>25</b>
<b>Error</b> .....	<b>25</b>
Your First MTConnect Hello World Application.....	29
MTConnect Requests and Responses.....	30
Mobile Devices – SAX and DOM Parsing .....	36
Hands-On Architecture, Agent and Hello World MTConnect Labs.....	37
MTConnect graphr.....	38
How To Test Your MTConnect Client.....	38
Security Concerns .....	38
github .....	39
Where To Go For Help .....	39

## MTConnect® Specifications or Materials

AMT - The Association For Manufacturing Technology (“AMT”) owns the copyright in this MTConnect® Specification or Material. AMT grants to you a non-exclusive, non-transferable, revocable, non-sublicensable, fully-paid-up copyright license to reproduce, copy and redistribute this MTConnect® Specification or Material, provided that you may only copy or redistribute the MTConnect® Specification or Material in the form in which you received it, without modifications, and with all copyright notices and other notices and disclaimers contained in the MTConnect® Specification or Material.

If you intend to adopt or implement an MTConnect® Specification or Material in a product, whether hardware, software or firmware which complies with an MTConnect® Specification, you SHALL agree to the MTConnect® Specification Implementer License Agreement (“Implementer License”) or to the MTConnect® Intellectual Property Policy and Agreement (“IP Policy”). The Implementer License and IP Policy each sets forth the license terms and other terms of use for MTConnect® Implementers to adopt or implement the MTConnect® Specifications, including certain license rights covering necessary patent claims for that purpose. These materials can be found at [www.MTConnect.org](http://www.MTConnect.org), or by contacting Paul Warndorf at [pwarndorf@mtconnect.hyperoffice.com](mailto:pwarndorf@mtconnect.hyperoffice.com).

MTConnect® Institute and AMT have no responsibility to identify patents, patent claims or patent applications which may relate to or be required to implement a Specification, or to determine the legal validity or scope of any such patent claims brought to their attention. Each MTConnect® Implementer is responsible for securing its own licenses or rights to any patent or other intellectual property rights that may be necessary for such use, and neither AMT nor MTConnect® Institute have any obligation to secure any such rights.

This Material and all MTConnect® Specifications and Materials are provided “as is” and MTConnect® Institute and AMT, and each of their respective members, officers, affiliates, sponsors and agents, make no representation or warranty of any kind relating to these materials or to any implementation of the MTConnect® Specifications or Materials in any product, including, without limitation, any expressed or implied warranty of noninfringement, merchantability, or fitness for particular purpose, or of the accuracy, reliability, or completeness of information contained herein. In no event shall MTConnect® Institute or AMT be liable to any user or implementer of MTConnect® Specifications or Materials for the cost of

procuring substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, indirect, special or punitive damages or other direct damages, whether under contract, tort, warranty or otherwise, arising in any way out of access, use or inability to use the MTConnect® Specification or other MTConnect® Materials, whether or not they had advance notice of the possibility of such damage.

## Purpose and Target Audience

As was stated in the document overview, this white paper was written primarily for software developers who are not in the manufacturing arena but who are interested in creating MTConnect client applications. This was written with the goal of expanding the number of software developers for MTConnect and hopefully the number of interesting applications that are created for MTConnect-enabled manufacturing equipment and devices. Specifically, what drove me to write this white paper from the point of view of a software developer who is *not* in manufacturing were the many conversations I had with my oldest son John, who is a gifted software developer, but knows nothing about manufacturing. We need to get the John Edstrom's of the world interested in manufacturing and developing MTConnect applications. Manufacturing is absolutely ripe for creative integration through new applications. MTConnect is becoming a huge enabler for manufacturing knowledge on the shop floor.

One of the most important events ever in the history of MTConnect is the MTConnect Challenge, which is comprised of two main goals (per the National Center for Defense Manufacturing and Machining website):

1. To engage and stimulate a broader base of software and system architects to develop advanced enterprise, facility, and machine control applications based on, and extensions to, the MTConnect standard to enable a more efficient and competitive domestic manufacturing infrastructure for the defense enterprise; and
2. To create valuable tools and applications that can be easily adopted by manufacturing enterprises, especially the lower tier producers, to enhance their manufacturing capabilities and support Department of Defense (DoD) supply chain management goals.

The main obstacle for the MTConnect Challenge to be successful is to make sure that the broadest group of experienced software developers can quickly understand the world of manufacturing so they can enter the challenge.

What is the definition of an experienced software developer? An individual who has multiple years of hands-on experience with software programming languages such as Java, C, C++, C# or other high-level programming languages, as well as extensive experience with web services, such as SAX, DOM, REST, XML and http. This description could also be described as the type of software background that most companies would expect when hiring a software developer.

What if the reader does not have expertise in those languages and protocols? The Internet is overflowing with sites where the reader can learn more about the languages and protocols listed above. The reader can also go to [MTConnectForum.com](http://MTConnectForum.com) for questions regarding MTConnect client development.

However, to be clear, without a software developer's background, this paper will have limited value to the reader, unless the goal is simply a high-level overview.

After a shop is MTConnect enabled, then shop floor monitoring is the obvious first step, but it is exactly that – a first step. The true value of MTConnect comes about when the data coming off the shop floor is not used for just shop floor monitoring, but also when it is integrated into the total manufacturing enterprise. The shop floor monitoring software companies have figured out how to use MTConnect by reading the standard. Some have attended MTConnect classes that Will Sobel, the Chief Architect for MTConnect and President of System Insights, has given. Some have attended hands-on courses that Joel Neidig and I have given on the basics of MTConnect at one of the [MC]2 MTConnect: Connecting Manufacturing Conferences.

While it is obvious that it would be very helpful to have deep manufacturing domain knowledge, it is not critical for the reader to have *deep* expertise of manufacturing. Deep expertise of manufacturing is needed if the reader intends to write an MTConnect agent or adapter. This is not to diminish the value of this domain knowledge, but you don't have to have been a machinist in order to write an application to get data off a machine tool. In the same way that a software developer doesn't need to be a neurosurgeon to write a medical application, the same is true of writing MTConnect client applications. However, the software developer does need to have a baseline knowledge of the nature of the business domain and this paper will provide high-level domain overview and pointers to additional documents and resources. This white paper will discuss agents and adapters from a high level, but is not a guide to write MTConnect agents or adapters.

If you are familiar with the shop floor, manufacturing equipment, and the connectivity challenges that exist, then you can probably skip the first three sections, "What Problem Does MTConnect Solve," "Turner's Five Laws," and "What You Need To Know About Manufacturing" sections. Those provide background and context for those who are not familiar with manufacturing.

Finally, a pet peeve of mine is when a white paper has so many references to other white papers that the pre-reading list and links are longer than the white paper itself. I tried to include enough about MTConnect so the reader is not constantly stopping and having to download yet another document. It is a delicate balance, so please let me know how that balance worked out and I will adjust in future revisions of this document. My email is listed at the end.

## Manufacturing's Fundamental Challenge

In September 2006, I made was attending the International Manufacturing Technology Show (IMTS) to get smart about manufacturing so I could give a keynote at AMT's annual meeting. I made two observations

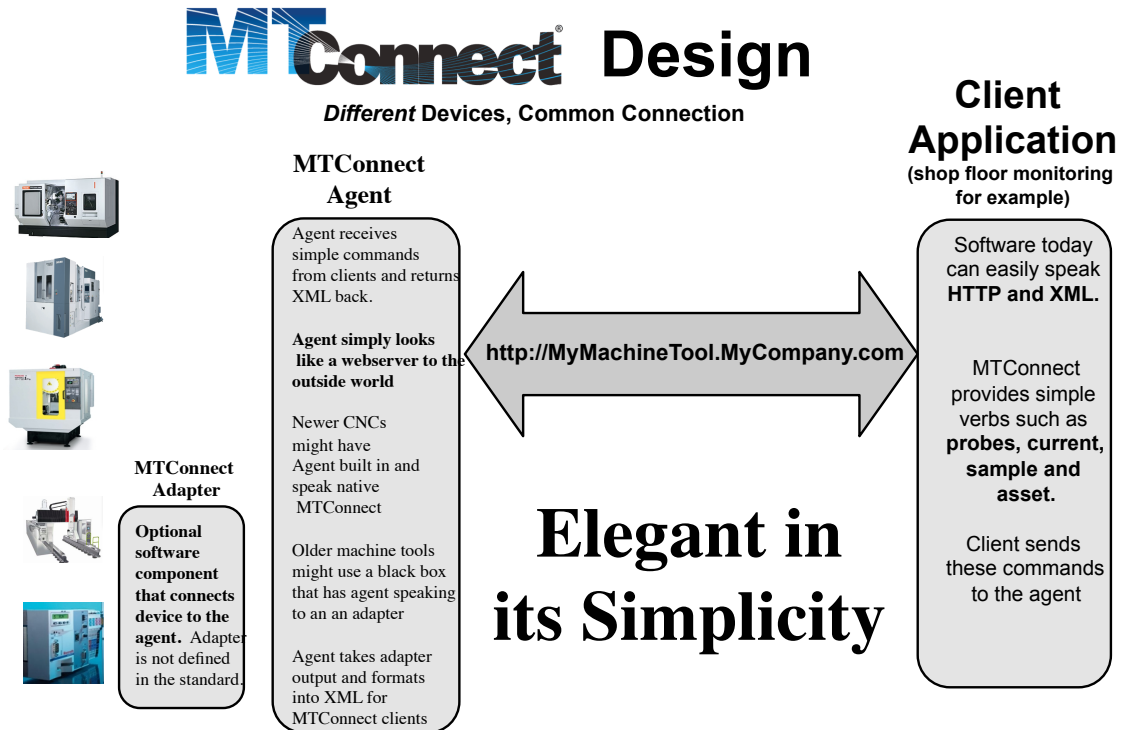
1. Manufacturing does not have a manufacturing problem. Manufacturing has a computer science problem. The manufacturing industry was like the computer industry back in the mid 1980s. There were too many network protocols and the fight was to own the winning protocol. Back then it was very expensive and you had to place a bet on which network protocol was going to win. It could easily be an additional \$700 to enable your PC to be networked in the enterprise. TCP/IP and Ethernet eventually won the network battle. When this happened the number of computers networked grew by many, many orders of magnitude, as did the software that would take advantage of the ubiquitous networking. It was the classic a rising tide lifts all ships.
2. Until you have an open and royalty-free way for these machine tools to speak to the rest of the world, nothing else really matters and manufacturing will just continue to struggle. The technologies are already out there today with XML, http and TCP/IP. Don't reinvent the wheel. Build on the de facto Internet platform that already exists today. Don't take the country club approach that has failed in manufacturing and other industries where you charge for the protocol and charge for each deployment.

MTConnect came out of this meeting based on keynotes given by Dr. Dave Patterson of UC-Berkeley and myself. Very smart and dedicated individuals built MTConnect. While it is not fair to recognize just two individuals for a standard such as MTConnect, it must be noted that Will Sobel, President and CEO of System Insights, was the chief architect and primary software developer for MTConnect, and Paul Warndorf, VP of Manufacturing Technology at AMT, was MTConnect's shepherd and guiding light.

MTConnect addresses the problem of having a common standard way to get information off manufacturing equipment. It does this using proven Internet protocols such as HTTP and XML, with a schema (data dictionary) and a common set of verbs.

MTConnect is elegant in its simplicity.

## The Three “A”s - Agents, Adapters and Applications



### What Do You Need To Know About Manufacturing?

A picture is worth a thousand words. Below is a before MTConnect picture.





## The Manufacturing World Prior To MTConnect



打開和免版稅將永遠贏得市場



ومنفتح وحر المملوك الفوز دائما في السوق



Offene und immer gewinnen den Marktplatz



Ανοικτή και θα κερδίσει πάντα την αγορά



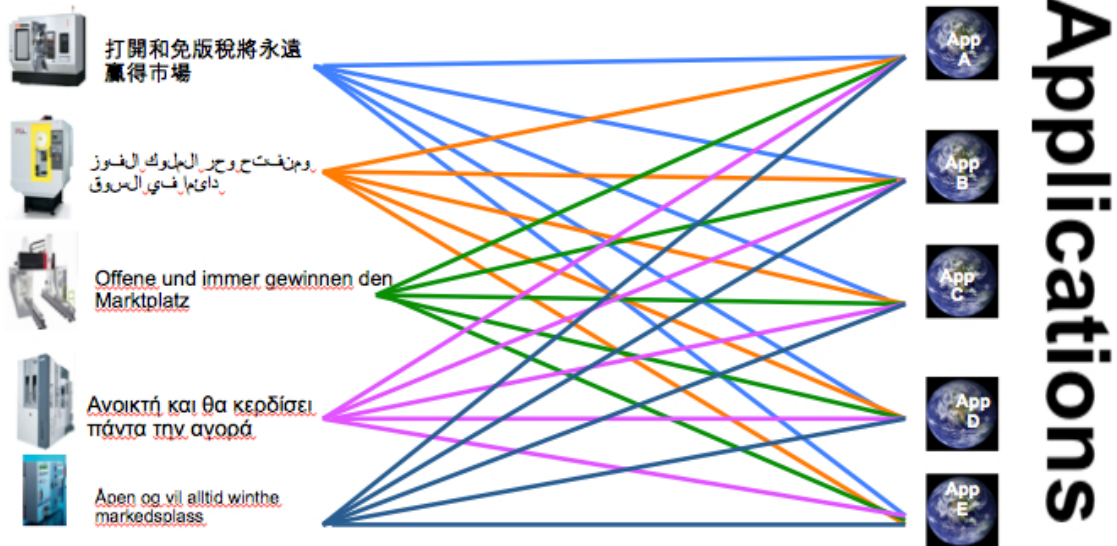
Åpen og vil alltid vinthe markedsplass

# A Connectivity and Integration Nightmare!

Notice in the diagram above that each machine tool speaks a different language. In the diagram below, you will notice that the number of two connections is the classic  $n$ -squared problem of connectivity. This was a nightmare for software application developers. The single point to remember here as a software developer of a client application is that prior to MTConnect, the onus was on you to have to write an adapter for each and every piece of manufacturing equipment you would be trying to get data from. MTConnect takes the burden from the application developer and moves it where it should be – closer to the device itself.
















# BEFORE MTConnect

The Classic  $N^2$  Problem



The diagram below shows (at a very high level) how MTConnect addresses this issue. From a developer standpoint, the problem of converting the proprietary formats has been moved to the adapter layer closer to the actual piece of manufacturing equipment. The advantage of this to the application developer is that now with MTConnect each client application only needs to concern itself with speaking MTConnect.

## Think of MTConnect the Bluetooth For Connecting Manufacturing Equipment to Applications

	打開和免版稅將 永遠贏得市場		What information would you like from me?	
	ومنفتح وحر الملوك الفوز دائما في السوق		What information would you like from me?	
	Offene und immer gewinnen den Marktplatz		What information would you like from me?	
	Ανοικτή και θα κερδίσει πάντα την αγορά		What information would you like from me?	
	Åpen og vil alltid vinthe markedsplass		What information would you like from me?	

Applications

### Turner's Five Laws of Manufacturing

Since this white paper is designed for software developers do not have a strong background in manufacturing, it is important provide context to the manufacturing domain. The most succinct way to provide context as it relates to writing an MTConnect client is to discuss Turner's Five Laws of Manufacturing. John Turner runs his own consulting company called FA & C Technology. John spent 30+ years at GE and GE Fanuc Automation specializing in manufacturing process intelligence and systems optimization as well as being a key member of the MTConnect Technical Advisory Board and MTConnect Steering Committee.

#### Turner's Five Laws of Manufacturing:

1. We Measure What Goes In To Production and What Comes Out. We Have Little Data on What Really Happens on the Production Floor
2. Anyone Who Says "I Know Exactly What Is Happening on My Plant Floor" – Don't Believe Them
3. We Don't Gather Data Because It Is Hard, and Someone Has to Look at It
4. No One Solution or Set of Data Works for Everyone
5. If You Don't Have an Avid Champion, Save Your Time and Money

If you want more specifics on each of these five laws, you can read the IMTS Insider article I wrote on this topic here – <http://tinyurl.com/TurnersLaws>

## MTConnect Basics

### What is MTConnect?

MTConnect is an open and royalty-free standard for manufacturing that is connecting manufacturing equipment with applications by using proven Internet protocols. Think of MTConnect as the “Bluetooth for manufacturing.” With Bluetooth, both devices must speak Bluetooth for anything useful to happen. Just as simply having an OBD-II port on your car does not provide you with any data unless you have an OBD-II scanner, the same “pairing” principle applies to MTConnect and software applications. You can have an MTConnect-enabled machine tool or piece of manufacturing equipment, but without the software to read and analyze the data you do not have both sides of the equation. The OBD-II scanner is really the application or the tool that you use to help you understand what is happening with your engine. In manufacturing, it is software applications, such as shop floor monitoring, that are the applications that speak to an MTConnect-enabled piece of manufacturing equipment on the shop floor.

### What Isn't MTConnect?

MTConnect is not an application. Let me say that again: **MTConnect is not an application.** The MTConnect Institute does not sell MTConnect. MTConnect is a standard based on open and royalty-free protocols. This is a common point of confusion with non-software developers, and that's why it is important to bring this point out.

### MTConnect Licensing

Please go to [MTConnect.org](http://MTConnect.org) for complete information. Below is a paragraph from the beginning of all MTConnect white papers that describes the *essence* of adopting or implementing MTConnect in a product: “If you intend to adopt or implement an MTConnect® Specification or Material in a product, whether hardware, software or firmware which complies with an MTConnect® Specification, you SHALL agree to the MTConnect® Specification Implementer License Agreement (“Implementer License”) or to the MTConnect® Intellectual Property Policy and Agreement (“IP Policy”). The Implementer License and IP Policy each sets forth the license terms and other terms of use for MTConnect® Implementers to adopt or implement the MTConnect® Specifications, including certain license rights covering necessary

patent claims for that purpose. These materials can be found at [www.MTConnect.org](http://www.MTConnect.org) , or by contacting Paul Warndorf at [pwarndorf@mtconnect.hyperoffice.com](mailto:pwarndorf@mtconnect.hyperoffice.com).”

There is a source code and binary FAQ at MTConnect.org for more information.

## Selling MTConnect

A frequently asked question is: “My company has agreed to the **MTConnect Intellectual Property Policy and Agreement** or the **MTConnect Specification Implementers License Agreement** and **intends to adopt or implement** an MTConnect® Specification or Material in a product, whether hardware, software or firmware which complies with an MTConnect® Specification. Does the MTConnect Institute recommend or determine pricing of my product?”

**The answer to the question above is no.** Companies are free to determine whatever price they choose, including making it free. The reason this question sometimes comes up is that there are no charges to join MTConnect and there are sometimes incorrect assumptions made regarding pricing.

We strongly encourage the reader to read and understand the **MTConnect Intellectual Property Policy and Agreement; MTConnect Specification Implementers License Agreement** as well as the **source code and binary FAQ** at MTConnect.org.

## Three Key Pillars of MTConnect

From a software developer perspective and at an 80,000-foot view, MTConnect comprises three key pillars:

1. Adapter
2. Agent
3. Application

Applications speak to an agent. The agent speaks to adapters (adapters are optional). Adapters or agents speak to manufacturing equipment.

### Adapter

The adapter is the optional piece of software that can be thought of as the bridge between the manufacturing equipment and the agent. The purpose of providing information on the adapter for software developers of client applications is to simply provide minimal information to understand what is happening between the device and the agent.

On one side, the adapter is speaking whatever protocol, as well as physical mechanism, that is needed to speak to a piece of manufacturing equipment (could be a machine tool, bar feeder, CMM, etc.), normalizing that data, putting into a modified version of the Simple Hierarchical Data Representation (SHDR) format and sending it to the agent. The SHDR protocol is a simple flat, fast and low latency human readable protocol delimited by pipes (|) that is in ISO 8601 date time format with optional decimal places. The reason the adapter is listed as being optional is that for those pieces of manufacturing equipment where MTConnect is native, there is no need for an adapter.

### **Below are some key points to remember about MTConnect adapters:**

- **Adapters are NOT in the MTConnect standard**
- Optional (for non-native MTConnect devices)
- Speaks to devices
- Adapters constantly stream output to agent(s)
- Applications typically do not speak to adapters directly, as this would involve bypassing the agent, which makes little logical sense.
- Adapters can be sending data to multiple agents or just one agent – design and implementation choice
- View of the world is MTConnect Agent(s)

### **Example of Reference Adapter Responsibilities**

1. The term “reference” agent is used loosely in this context because there is no standard for an adapter. This was done by design. While this is true, it is also true that many agent developers have used the SHDR protocol as the mechanism for the adapter to send data to the agent.
2. Provide a socket connection to which a process can connect and receive updates.
3. Only send data when it changes (**this is important**) and only sends data after handshake with agent. In other words, adapter will not just send out data with no known agent.
4. Make sure all data that must conform to a controlled vocabulary, fixed set of values, only communicates those values. Examples are ControllerMode and Execution.
5. Provide recovery when client disconnects and support multiple connections for testing
6. Send all initial values to new clients when they connect
7. Format the data according to the SHDR specification
8. There is a heartbeat between reference adapter and agent. See in upcoming slide.
9. Perform any connection to the manufacturing equipment interfaces and gather data.
  - NOTE: This connection to the machine tool interface and the gathering of data is where all of the hard and time-consuming work is done.

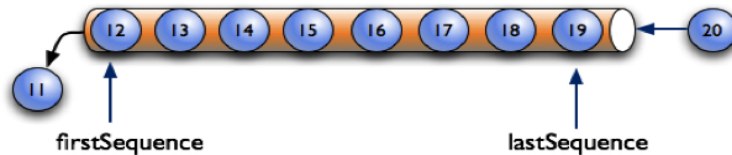
REMEMBER, the adapter is NOT part of the standard and this just provides background on the reference adapter.

### Agent

The agent can be thought of as the bridge between adapter(s) and application(s). As a developer of client applications, expertise in what the agent is doing and understanding the standard is critical.

Below is a picture of what the agent is doing:

## The Agent “Bucket” Buffer Model



- Think of Agent as having two *buckets or buffers* that controller deposits data into
  - SamplesEventsConditions Buffer default is 131,072 is  $2^{17}$  NOTE: This must be a factor of two for speed purposes
  - Assets Buffer default is 1,024
  - Both of these can be changed to different sizes based on need
- Each application “reaches into the bucket” at the rate it wants
- As bucket fills up, older measurements “leak out” the bottom
  - whether or not any applications have read them
  - Capacity of bucket depends on Agent implementation
- The ability to keep or persist the data can be done either at the application level or creating a custom agent – **developers design choice**

### Application

The application or client is the software that speaks directly to the agent, requesting information and keeping track of the sequence numbers so it can continue to request and process XML information coming from the agent. The most obvious application when connecting to a piece of manufacturing equipment is shop floor monitoring. A shop floor monitoring program typically requests information from the agent(s), processes that information and then displays graphs or charts on what the piece of manufacturing equipment is doing.

What is interesting for software developers is all the ways MTConnect data can be analyzed in the countless other non-shop floor applications.



## Three Types of MTConnect Devices and Discovery

# Three Types of MTConnect Device Connections

1. Native
2. Translation Dependent
3. Connection Dependent

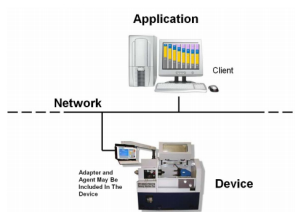


Figure 2 – MTConnect Native Device

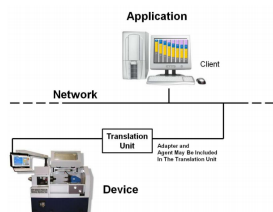


Figure 3 – MTConnect Translation Dependent Device

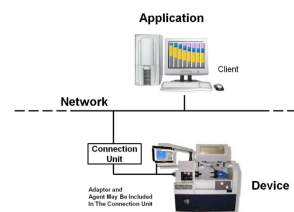


Figure 4 – MTConnect Connection Dependent Device

As a developer of an MTConnect client application, you will need to understand the high-level differences of the three types of device connections. The important difference between the native, translation dependent and connection dependent MTConnect devices is how the data becomes MTConnect enabled. As a software developer of client applications, you are much more concerned with *what* types of data are available, as opposed to *how* the data was obtained and transformed into MTConnect data.

How do you know what agents and manufacturing equipment is available? Hopefully, the plant's IT department (if they have one) is using DNS, Active Directory or some similar dynamic discovery mechanism so you are not using IP addresses directly.

## MTConnect Standard Overview

The MTConnect standard is broken down into four parts. What you see below are the four standard sections of MTConnect with the major sub-sections. Later, we will explore what the software developer needs to know in order get data from an MTConnect-enabled device. Before the developer is able to develop anything more than a very simple application, it will be necessary read and understand Parts 1, 2 and 3 of the standard. Part 4 is for assets and this becomes important for developers who are using those typically “mobile” assets of the manufacturing equipment.

#### MTConnect Standard Part 1

- Overview and Protocol
  - MTConnect Document Structure
  - Versioning
  - HTTP and XML – Brief Reminder
- Architecture Overview
  - Request Structure
  - Agent initialization
  - Application Communication
  - Agent Data and agent asset storage
- Reply XML Document Structure
  - Devices
  - Streams
  - Assets
  - Error
- Protocol (Commands to Agent) Overview
  - Probe
  - Sample
  - Current
  - Asset
  - MTConnect Agent and Adapters

#### MTConnect Standard Part 2

- Components
- Data Items

#### MTConnect Standard Part 3

- Streams
- Events
- Samples
- Conditions

#### MTConnect Standard Part 4

- Assets

An example of an asset is something that is associated with the manufacturing process that is NOT a component of the device AND can be removed. Assets are

in Part 4 of the MTConnect standard and will concern itself with the modeling of these assets and the management and communication of asset data using MTConnect.

**Concrete Examples of Assets:**

- Cutting Tools, Workholding Systems, and Fixtures.
- ISO 13399 – standard for describing product data regarding cutting tools, independent from any particular system

## Programmatic Conceptual Terms

The terms below are key conceptual terms that software developers will need to know and understand when developing application will speak to the MTConnect agent.

- **Header** - Protocol related information.
- **Components** - The building blocks of the device.
- **Data Items** - The description of the data available from the device.
- **Streams** - A set of samples or events for components and devices.
- **Samples** - A point-in-time measurement of a Data Item that is continuously changing.
- **Events** - Unexpected or discrete occurrence in a component. This includes state changes and conditions.
- **Conditions** - Normal, Warning or Fault. Warning is trending toward a fault and fault means the device has stopped and needs intervention to get it working.

## MTConnect’s Powerful Namespace – XML Schema Definition aka Data Dictionary

A very powerful component of the MTConnect standard (tons of hard work went into creating this) is the XML schema or the data dictionary. The file [http://www.mtconnect.org/schemas/MTConnectDevices\\_1.2.xsd](http://www.mtconnect.org/schemas/MTConnectDevices_1.2.xsd) is more than 600 lines long. It is this data dictionary that makes the job of the client application developer much easier. Applications that do not provide a schema place the burden on determining what each value means as well as the allowable values back on to the shoulders of the application developer. Below is a graphic that drives home the point.

How to extend the namespace is at [MTConnect.org](http://www.MTConnect.org)

## The Simplest MTConnect Clients – Your Browser & Spreadsheet

In order to make it easy to test an MTConnect client, the MTConnect Institute has paid for a simulator at [MTConnect.Agent.org](http://MTConnect.Agent.org). This document will use [MTConnect.Agent.org](http://MTConnect.Agent.org) to show how to use the MTConnect commands.

The first step in understanding how to write an MTConnect client application is to understand how to speak to the agent.

- You can **literally** type the name of your machine tool into your browser. To see how this might work, there is a machine tool simulator running at [agent.MTConnect.org](http://agent.MTConnect.org). Type the following into your browser window:
  - <http://agent.MTConnect.org/probe>
- You can **literally** type the name of your machine tool into Excel and start reading and charting MTConnect data. How? Excel speaks HTTP and XML like ALL applications that were not written in a cave.
- MTConnect **READS** manufacturing equipment data (from a machine tool for example, but could be any type of device on the shop floor) and converts it to XML.

Below are the very easy steps to get MTConnect data into Excel are as follows:

- Go into the **Data** selection on the upper tab
- Select **From Web**
- New Web Query will come up
- In the browser bar that pops up, put in your MTConnect http url (for example):

<http://agent.mtconnect.org/sample?count=1000>

- Hit **Import**
- **You will then hit Import** again after the data is loaded into the window.
- Select **Yes** to add to the question, Do You Want To Continue To Add This Schema To Your Notebook?
- When the pop-up window comes up for **Import Data**, select **Existing Worksheet**, which is the bottom selection when you are prompted on where to put your data
- At this point Excel will put in the proper headers and values in the correct locations
- Of course, you will likely want to use a more specific Sample Request in step #4, but I kept it simple to just to make the point how easy it is for applications to get data from an MTConnect enabled device.

There is a four-minute screencast on exactly how to do this. Notice that I did absolutely nothing in terms of laying out the titles of the columns, where the columns and rows should go, where the data should appear. The reason why is that since this is XML, Excel (just like a lot of software that reads XML data) knows how

to deal with the data and display it properly. Note that the video starts off with a gray screen for the first few seconds and then straightens out. You can view this at <http://tinyURL.com/ExcelMTConnect>

## Fault Tolerance and Persistence of Data

MTConnect was designed to be simple and scalable. Below are important points to remember as a software developer of client applications for MTConnect.

- MTConnect uses a **RESTful protocol** — meaning it is stateless on the server side — much like NFS (Network File System – created by Sun Microsystems)
- There is NO fault tolerance or automated recovery for MTConnect. It is up to the agent to keep track of and persist **any** data. The reference agent does not persist data. A circular buffer is defined where the adapter is writing data in.
- Developing clients, you should ask how the adapters (which are optional) and the agents are restarted in case of failure. Are they defined as services that are automatically restarted? While the adapters and agents are outside the scope of this white paper, it is still important to understand what happens with adapters and agents when they fail.
- Therefore, it is up to the **client to keep track of state**. For example, at a minimum, the client needs to remember BOTH the next sequence number and the instance ID in case the agent goes down.
- Agent failure is the more complex scenario and requires the use of the instanceId.
  - The instanceId was created to facilitate recovery when the Agent fails and the application is unaware. Since HTTP is a connectionless protocol, there is no way for the application to easily detect that the Agent has restarted, the buffer has been lost, and the sequence number has been reset.

## MTConnect Sequence Number

The MTConnect sequence number is between 1 and 18,446,744,073,709,551,615. MTConnect is a RESTful protocol and it is up to the application developer to track the sequence number.

Below is a snippet from running the following simple MTConnect command that runs a sample with a count of 10:

**`http://agent.mtconnect.org/sample?count=10`**

```
<Header creationTime="2012-12-26T13:55:15Z" sender="mtconnect"
instanceId="1340212647" version="1.2.0.10" bufferSize="131072"
nextSequence="2933744389"
firstSequence="2933744379"
lastSequence="2933875450"/>
```

```
<Streams><DeviceStream name="VMC-3Axis" uuid="000">
```

Notice the bold XML attributes highlighted above in the header. As a developer of applications, the instanceId is important. As was just pointed out in a previous paragraph, the instanceId was created to facilitate recovery when the Agent fails and the application is unaware. It is up to the developer to track sequence numbers. Version is important, as there have been changes in the standard. The MTConnect specification is the best place to review these changes.

## MTConnectDevices\_1.2.xsd

```
<xs:element name='MTConnectStreams' type='MTConnectStreamsType'>  
  <xs:annotation>  
    <xs:documentation>  
      The root node for MTConnect  
    </xs:documentation>  
  </xs:annotation>  
</xs:element>  
<xs:simpleType name='SenderType'>  
  <xs:annotation>  
    <xs:documentation>
```

The date and time the document was created

```
  </xs:documentation>  
</xs:annotation>  
<xs:restriction base='xs:dateTime'>  
</xs:simpleType>  
<xs:simpleType name='SequenceType'>  
  <xs:annotation>  
    <xs:documentation>  
      A sequence number  
    </xs:documentation>  
  </xs:annotation>  
  <xs:restriction base='xs:integer'>  
    <xs:minInclusive value='1' />  
    <xs:maxExclusive value='18446744073709551615' />
```

Notice that we are defining the MTConnect sequence number, which is an integer as well as what the minimum sequence maximum numbers can be.

## MTConnect Responses

# 4 kinds of XML Responses in MTConnect

- **Devices**
  - descriptive information about the configuration of the machine(s) and what data can be delivered
  - returned by *probe* command issued to Agent
- **Streams**
  - Data samples and events from the device(s)
  - returned by *sample* or *current* command issued to Agent
- **Assets**
  - Retrieve information on mobile assets
- **Error**
  - Returned when an error occurs that prevents further processing
  - Caveat: most things that don't work as you expect aren't necessarily errors

To be an MTConnect compliant the **device only needs to report on availability**

- `<DataItem category="EVENT" id="avail" type="AVAILABILITY"/>`
- **Everything** else is optional

It's very important to understand the four kinds of MTConnect responses.

## Devices

- An Agent represents one or more *devices*
- A device ("machine") is a collection of *components*
  - A component can have sub-components, allowing representation of arbitrarily complex devices
  - In fact, the Device itself is one type of Component
- There's a fixed set of Component types
  - Special components can be used to extend the specification – we reference a white paper later on this
  - Each component also has a *name*, and possibly some other attributes

## Anatomy of a Component (Examples)

- Attributes
  - uuid - A globally-unique id for the component (required for a Device; optional for other component types)
  - **name** - The name of the component
  - sampleRate - An optional sample rate
- Description
  - example: Manufacturer, serial number, free-form descriptive text
- DataItems
  - *What can be reported* by this component
- Components: subcomponents (recursively defined)
  - A component may have *its own* data
  - example: PATH\_FEEDRATE for Axes component, *in addition* to POSITION or ROTARY\_VELOCITY for each axis

## Data Items and their Types

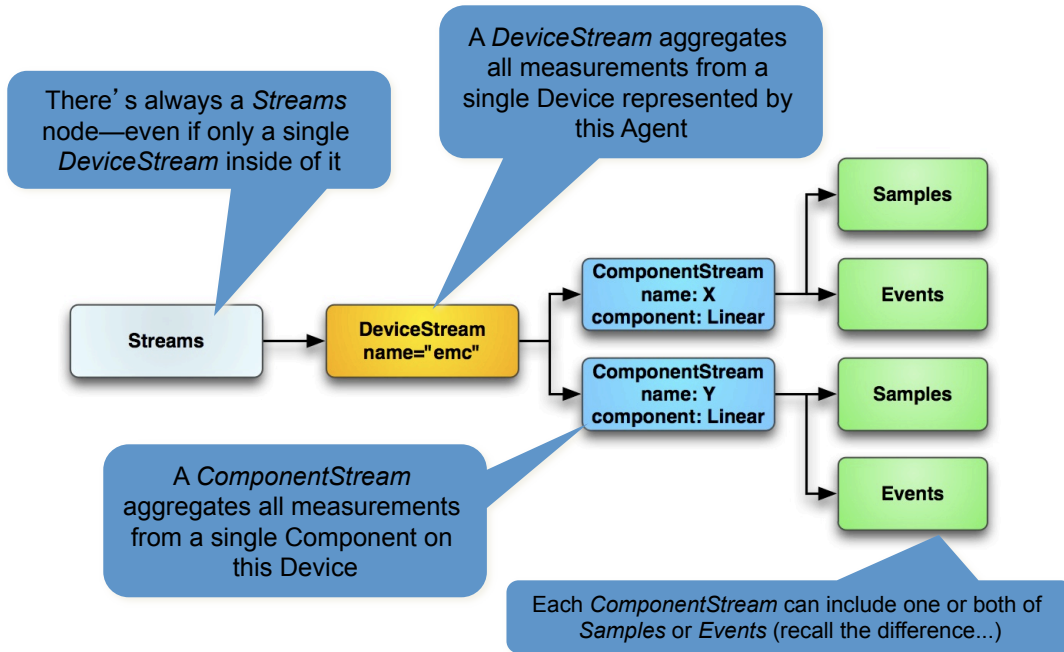
- The type of a specifies what (kinds of) values its corresponding Samples or Events may take
- POSITION is a numerical value
- EXECUTION is one of IDLE, PAUSED, EXECUTING
- DIRECTION is one of CLOCKWISE or COUNTER\_CLOCKWISE
- legal units are codified in the MTConnect Specification

## Streams

- What is a *stream*?
  - A collection of samples and events
  - organized by device & component
- Rationale: flat stream of data reduces duplication of static relationships and information. Can always reference back to devices.
- Provides simple structure for XML parsing (data is easily aggregated)
- The *probe* command returns *Devices* (metadata)
- The *current* and *sample* commands return *Streams* (data)
- There are two variants of *current* and one variant of *sample*



# Stream Structure



## Assets

An asset is a mobile asset which is not a component of a device and can be removed. Please see the diagram on the next page for specifics.

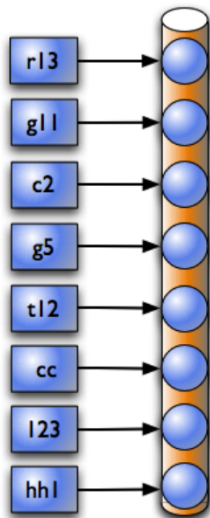
## Error

An error is returned when an error occurs that prevents further processing.

## **MTConnect *Agent* Asset Buffer**

- MTConnect stores assets in a separate “bucket” buffer.
- The Agent provides a limited number of assets that can be stored at one time and uses the same method of pushing out the oldest asset when the buffer is full. The buffer size for the asset storage is maintained separately from the sample, event, and condition storage.
- Assets behave like a key/value in memory database.
- In the case of the asset the key is the `assetId` and the value is the XML describing the asset.
- The key can be any string of letters, punctuation or digits and represent the domain specific coding scheme for their assets.
- Each asset type will have a recommended way to construct a unique `assetId`, for example, a Cutting Tool SHOULD be identified by the tool ID and serial number as a composed synthetic identifier.

# MTConnect Agent Asset Buffer



- When an asset is added or modified, an **AssetChanged** event will be sent to inform us that new asset data is available.
- The application can request the new asset data from the device
- The tool data **MUST** remain constant until the AssetChanged event is sent. Once it is sent, data **MUST** change to reflect the new content at that instant. The time reflect the time the last change was made to the asset data.
- **Every time an asset is modified or added it will be moved to the end of the buffer and become the newest asset.**
- As the buffer fills up, the oldest asset will be pushed out and its information will be removed.
- **The application is RESPONSIBLE for persistence.**

## Assets

- Request for all the assets in the *Agent*:
- *url*: <http://agent.mtconnect.org/assets>
- *Returns all available MTConnect assets in the Agent. MTConnect **MAY** return a limited set if there are too many asset records. The assets **MUST** be added to the beginning with the most recently modified assets.*
- **Example url**:
  - <http://DavesMachineTool.DE.com/asset/decut1;decut2>
  - decut1 and decut2 are mobile asset identifiers

**Below is from github and an example of what might be returned from an MTConnect asset probe.**

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2010 rel. 2 (http://www.altova.com)-->
<MTConnectAssets xsi:schemaLocation="urn:mtconnect.org:MTConnectAssets:1.2
../MTConnectAssets_1.2.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:mtconnect.org:MTConnectAssets:1.2"
xmlns:mt="urn:mtconnect.org:MTConnectAssets:1.2">
  <Header creationTime="2001-12-17T09:30:47Z" instanceId="1" sender="String" version="1.1"
bufferSize="112321"/>
  <Assets>
    <ToolAssembly name="MC20006L" serialNumber="112233">
      <Description>0.5000" Drill, HSK63A-ER32 Collet Chuck, 6.000" Proj</Description>
      <ToolState>FRESH</ToolState>
      <PotCount>1</PotCount>
      <Inserts>
        <Insert insertId="1">
          <Length maximum="6.5000" type="NOMINAL"
minimum="5.5000">6.0000</Length>
          <Diameter maximum="2.0005" type="NOMINAL"
minimum="1.9995">2.0000</Diameter>
          <TipAngle maximum="90" type="NOMINAL" minimum="88">90</TipAngle>
          <CornerRadius maximum="0.0012" type="NOMINAL"
minimum="0.0008">0.0010</CornerRadius>
        </Insert>
      </Inserts>
    </ToolAssembly>
    <ToolAssembly name="MC2000L" serialNumber="11233">
      <Description>0.5000" Drill, HSK63A-ER32 Collet Chuck, 6.000" Proj</Description>
      <ToolState>FRESH</ToolState>
      <PotCount>1</PotCount>
      <Inserts>
        <Insert insertId="1">
          <Length type="MEASURED">6.0375</Length>
          <Diameter type="MEASURED">2.0002</Diameter>
          <TipAngle type="MEASURED">89</TipAngle>
          <CornerRadius type="MEASURED">0.0011</CornerRadius>
        </Insert>
        <Insert insertId="3">
          <Length type="MEASURED">5.9764</Length>
          <Diameter type="MEASURED">1.9998</Diameter>
          <TipAngle type="MEASURED">88.5</TipAngle>
          <CornerRadius type="MEASURED">0.0009</CornerRadius>
        </Insert>
      </Inserts>
    </ToolAssembly>
  </MobileAssets>
</MTConnectAssets>
```

## Your First MTConnect Hello World Application

Below is the simplest of Java programs. The program uses Java's basic networking and IO classes to read agent.MTConnect.org with a probe command, print the XML out until there is nothing left to print. Below **Corvette:** is my prompt on my MacBook Pro in a terminal window.

**Corvette:** more MTConnectHelloWorld.java

```
import java.net.*;
import java.io.*;

public class MTConnectHelloWorld {

    public static void main(String[] args) throws Exception {

        System.out.println("Your First Hello World Program" + "\n");

        URL MTConnect = new URL("http://agent.MTConnect.org/probe");
        BufferedReader in = new BufferedReader(
            new InputStreamReader(MTConnect.openStream()));

        String inputLine;
        while ((inputLine = in.readLine()) != null)
            System.out.println(inputLine);
        in.close();

        System.out.println("\n" + "XML from probe command printed above");

    }
}
```

Compile:

**Corvette:** javac MTConnectHelloWorld.java

Execute:

**Corvette:** java MTConnectHelloWorld

At this point you will see a few pages of XML coming from the simulator that is running as a CNC vertical mill doing a very simple part in an endless loop.

If you are on a \*nix (Mac OS X is obviously in this category as well) box, then you can accomplish the same as above from a command line:

**Corvette:** curl <http://agent.MTConnect.org/probe>

If you are a PC user, curl comes with the powershell terminal or you could ftp curl as well. As pointed out earlier, you could also do this from a web browser by just putting in the URL above. You can put these in your browser to see the results.

## MTConnect Requests and Responses

As a software developer for MTConnect applications, sending the appropriate MTConnect commands to the agent, parsing the results, analyzing, displaying and likely storing the returned XML MTConnect responses are the high-level sections of an MTConnect application.

It is important to remember the MTConnect big picture:

1. Agent(s) and Adapter(s) {adapters are optional} start up
2. *Applications* start up and (optionally) query name server to discover Agent(s).
  - Application may already know where agent(s) are, e.g. in a relatively static deployment scenario
3. **Applications communicate *requests* to Agents (via HTTP) and receive *responses* (in XML).**

Below are a few examples:

Typically, the first command is a probe to the agent to retrieve the metadata manufacturing device is capable of providing.

**<http://agent.mtconnect.org/probe>**

Run the MTConnect probe command in a browser and notice the devices. We have already discussed the header earlier. Below is a snippet from the probe to show you Devices node, the Device with the id of “dev” and the DataItems with the category of “Event”.

```
<Devices>
  <Device id="dev" iso841Class="6" name="VMC3Axis" sampleRate="10" uuid="000">
    <Description manufacturer="SystemInsights"/>
    <DataItems>
      <DataItem category="EVENT" id="avail" type="AVAILABILITY"/>
      <DataItem category="EVENT" id="dev_asset_chg" type="ASSET_CHANGED"/>
    </DataItems>
  </Device>
</Devices>
```

Notice below the Components, the Axes and the DataItems.

```

<Components>
  <Axes id="ax" name="Axes">
    <Components>
      <Rotary id="c1" name="C">
        <DataItems>

<DataItem category="SAMPLE" id="c2" name="Sspeed" nativeUnits="REVOLUTION/MINUTE" subType="ACTUAL" type="SPINDLE_SPEED" units="REVOLUTION/MINUTE">

      <Source>spindle_speed</Source>
    </DataItem>

<DataItem category="SAMPLE" id="c3" name="Sovr" nativeUnits="PERCENT" subType="OVERRIDE" type="SPINDLE_SPEED" units="PERCENT">

      <Source>SspeedOvr</Source>
    </DataItem>
    <DataItem category="EVENT" id="cm" name="Cmode" type="ROTARY_MODE">
      <Constraints>
        <Value>SPINDLE</Value>
      </Constraints>
    </DataItem>
    <DataItem category="CONDITION" id="Cloadc" type="LOAD"/>
  </DataItems>
</Rotary>

```

**<http://agent.mtconnect.org/current>**

The current request must return most current component's data values unless the option at argument is used

- at – [optional] specifying the MTConnect protocol sequence number. MUST NOT be used with the interval command as this will just keep returning the same data repeatedly
- path – xpath expression the specifies components and/or data items

Run the MTConnect current command in a browser and notice the devices. We have already discussed the header earlier. Below is a snippet from the current request to show you Streams node, the DeviceStream, the ComponentStream with the id of "Rotary" and the various dataItemIds.

```

<Streams>
  <DeviceStream name="VMC-3Axis" uuid="000">
    <ComponentStream component="Rotary" name="C" componentId="c1">
      <Samples>
        <SpindleSpeed dataItemId="c2" timestamp="2013-01-
01T19:34:04.446894Z" name="Sspeed" sequence="2934360162" subType="ACTUAL">UNAVAILAB
LE
      </SpindleSpeed>

```

```

    <SpindleSpeed dataItemId="c3" timestamp="2013-01-
01T19:34:04.446894Z" name="Sovr" sequence="2934360163" subType="OVERRIDE">UNAVAILABL
E<
    /SpindleSpeed>
  </Samples>
  <Events>
    <RotaryMode dataItemId="cm" timestamp="2012-06
20T17:17:27.439360Z" name="Cmode" sequence="16">SPINDLE</RotaryMode>
  </Events>
  <Condition>
    <Unavailable dataItemId="Cloadc" timestamp="2013-01-
01T19:34:04.446894Z" sequence="2934360155" type="LOAD"/>
  </Condition>
</ComponentStream>
<ComponentStream component="Controller" name="controller" componentId="cn1">
  <Events>
    <EmergencyStop dataItemId="estop" timestamp="2013-01-
01T19:34:04.446894Z" sequence="2934360171">UNAVAILABLE</EmergencyStop>
    <Message dataItemId="msg" timestamp="2012-06-
20T17:17:27.439360Z" sequence="28">UNAVAILABLE</Message>
  </Events>
  <Condition>
    <Unavailable dataItemId="clp" timestamp="2013-01-
01T19:34:04.446894Z" sequence="2934360165" type="LOGIC_PROGRAM"/>
  </Condition>

```



## The sample Command

- Retrieves a series of data starting from a position and return up to the requested number of samples or events
- Allows application to retrieve all data without missing anything
- Can stream data as it arrives
- Can be thought of as a window into the stream of data

## Sample Request Parameters

### Sample request retrieves values for component's data items

- **path** – xpath expression the specifies components and/or data items
  - default is all components in device or devices if no device is specified
- **from** – starting sequence number for Events, Samples, Conditions
  - default is 0
- **interval** – time in milliseconds that the agent should pause between sending samples for Events, Samples, Conditions
  - Sample can be used with an interval in special cases and this should only be done judiciously. (App beware.) The interval used to be limited to a minimum of 10 ms, but with the device integration we allowed it to go down to 0.
  - **What does this mean?**
    - When you specify an interval of zero you're telling the agent that when a value changes, notify me IMMEDIATELY and don't try to collect any additional data items. This is great for processes that need to do real-time monitoring (< 10ms). It should always be used with a path since this will allow only specific events to trigger.
  - We usually use 1000 (or one second) intervals between samples.
- **count** – must return next sequence
  - default is 100

<http://agent.mtconnect.org/sample>

Run the MTConnect sample command in a browser. Below is a snippet from the sample request (with no additional parameters) to show you Streams node, the ComponentStream, the ComponentStream with the name of “VMC-3Axis” and the various Events, Conditions and data items.

```
<Streams>
  <DeviceStream name="VMC-3Axis" uuid="000">
    <ComponentStream component="Rotary" name="C" componentId="c1">
      <Samples>
```

```

<ComponentStream component="Controller" name="controller" componentId="cn1">
  <Events>
    <EmergencyStop dataItemId="estop" timestamp="2012-12-
31T04:12:51.242511Z" sequence="2934232704">UNAVAILABLE</EmergencyStop>
  </Events>
  <Condition>
    <Unavailable dataItemId="clp" timestamp="2012-12-
31T04:11:51.227979Z" sequence="2934232644" type="LOGIC_PROGRAM"/>
  </Condition>
</ComponentStream>

```

## Sampling and XPath

- Remember: the path parameter in the command refers to the *component* structure (not the stream structure)
- XPath allows arbitrarily complex expressions

Expertise in XPath is always important with web applications that use XML documents and it is certainly the case with MTConnect.

Let's try a simple XPath expression with the sample request:

**<http://agent.mtconnect.org/sample?count=1000>**

Everything after the ? is a parameter passed to the sample command. The above request passes the variable count of 1000. The time range of information received is a function of the sampling rate that the adapter passes to the agent. In the example above at agent.MTConnect.org, the sampling rate is 1,000 per second so it is one second of data. This can be verified by looking at nextSequence value of "**2934239884**" and the firstSequence being "**2934238884**".

```

<Header creationTime="2013-01-
01T22:35:07Z" sender="mtconnect" instanceId="1340212647" version="1.2.0.10" b
ufferSize="131072" nextSequence="2934239884" firstSequence="2934238884" la
stSequence="2934369955"/>

```

As with most web programming with XML documents, path and // are used as a wildcard and used quite a bit with complicated XML documents.

Run the MTConnect sample command with the wildcard for all **Axes** in a browser.

<http://agent.mtconnect.org/current?path=//Axes>

The MTConnect XML response will list Rotary and Linear axes.

```
<ComponentStream component="Rotary" name="C" componentId="c1">
```

```
<ComponentStream component="Linear" name="X" componentId="x1">  
<ComponentStream component="Linear" name="Y" componentId="y1">  
<ComponentStream component="Linear" name="Z" componentId="z1">
```

As would be expected, Rotary, X, Y and Z linear axes information would be returned.

Run the MTConnect sample command with the wildcard for Linear in a browser.

<http://agent.mtconnect.org/sample?&path=//Linear>

Below are two lines from the response where the component "Linear" was selected.

```
<ComponentStream component="Linear" name="X" componentId="x1"><Samples>  
<ComponentStream component="Linear" name="Y" componentId="y1"><Samples>  
<ComponentStream component="Linear" name="Z" componentId="z1"><Samples>
```

The @ with square brackets [] are used to access attributes. This is used to access XML attributes and is used in many MTConnect client applications.

Run the MTConnect sample command in a browser with the wildcard for Linear and look for "X" which would be the X axis. Note the use of single and not double quotes for the X axis.

**[http://agent.mtconnect.org/sample?&path=//Linear\[@name='X'\]](http://agent.mtconnect.org/sample?&path=//Linear[@name='X'])**

```
<ComponentStream component="Linear" name="X" componentId="x1">  
  <Samples>  
    <Position dataItemId="x2" timestamp="2012-12-  
31T08:05:24.880333Z" name="Xact" sequence="2934245264" subType="ACTUAL">UNAVAILABLE  
  </Position>  
  
  </Samples>  
  <Condition>  
    <Unavailable dataItemId="Xloadc" timestamp="2012-12-  
31T08:05:54.887601Z" sequence="2934245273" type="LOAD"/>  
  </Condition>  
</ComponentStream>
```

Run the MTConnect current command in a browser with the wildcard for Axes, Linear and DataItem and look for any DataItem that would have an attribute of type=ACTUAL

**[http://localhost:5000/current?path=//Axes//Linear//DataItem\[@subType='ACTUAL'\]](http://localhost:5000/current?path=//Axes//Linear//DataItem[@subType='ACTUAL'])**

```
ComponentStream component="Linear" name="X" componentId="x1">
  <Samples>
    <Position dataItemId="x2" timestamp="2013-01-02T01:27:50.765651" name="Xact"
sequence="323382" subType="ACTUAL">-1.3000440598
    </Position>
  </Samples>
</ComponentStream>
<ComponentStream component="Linear" name="Y" componentId="y1">
  <Samples>
    <Position dataItemId="y2" timestamp="2013-01-02T01:27:50.765651" name="Yact"
sequence="323383" subType="ACTUAL">0.1999527365
    </Position>
  </Samples>
</ComponentStream>

<ComponentStream component="Linear" name="Z" componentId="z1">
  <Samples>
    <Position dataItemId="z2" timestamp="2013-01-02T01:25:16.123716" name="Zact"
sequence="280371" subType="ACTUAL">-0.1000000015
    </Position>
  </Samples>
</ComponentStream>

</DeviceStream>
```

## Mobile Devices – SAX and DOM Parsing

Joel Neidig, who is a Systems Engineer at ITAMCO, wrote the first mobile app. I created a half-hour webinar under the umbrella of the Offices of Strategic Innovation Roundtable (OSI/R) where I interviewed Joel. This ITAMCO OSI/R is located here <http://tinyurl.com/ITAMCO-MTConnect>. In the webinar, we discussed the exact steps went through to write a mobile app for MTConnect. Joel discusses Apple's SDK and the steps a developer goes through to write and place an app in the iTunes store. Joel discusses [Objective-C](#) and the GUI that is part of the Apple SDK. Joel also talked about Android as well. For Android, Joel used Eclipse as the IDE. Google has lots of tutorials on how to get started, as does Apple.

When I asked Joel what he thought was the trickiest part of developing in iOS, he replied that the lack of an XML parsers with the SDK. ITAMCO created a Simple API for XML (SAX) parser for iOS. For the Android platform, ITAMCO used a Document

Object Model (DOM) parser. DOM on the Android takes more horsepower to run. As this is written for software developers, there are other XML parsing choices and it is not appropriate to discuss this here in further detail. ITAMCO has open sourced both of these apps as well. ITAMCO has since written an MTConnect Blackberry app as well.

At the [\[MC\]2 2011 MTConnect: Connecting Manufacturing Conference](#), Joel and I had a hands-on session on MTConnect architecture where we go through basic application development with Chris Tacke leading a hands-on MTConnect Hello World lab, as well as Nat Frampton, who led a hands-on lab for Adapters. The specifics are in the next section. At [MC]2 2013, Joel and I gave an updated version of this MTConnect Architecture Lab as well.

## **Hands-On Architecture, Agent and Hello World MTConnect Labs**

This section lists the six hands on labs for software developers and implementers interested in learning how MTConnect works. These labs are at MTConnect.org under the Developer section. You will need a login at MTConnect.org to view these three labs.

NOTE: If you want to have access to the actual MTConnect Standard, Schema, Development Documents and Instructions/Guidelines, then you will simply need to have a login here at MTConnect.org.

Below are six videos that are one hour and 45 minutes each that will answer software developer's basic questions on MTConnect. This starts out with a presentation and hands-on for the basic MTConnect protocol, shows how to build a reference adapter and how the agent works, as well as writing your first MTConnect application.

### **Hands-On MTConnect Software Development Webinars - [MC]2 2011**

#### **MTConnect 101: Fundamentals of MTConnect Workshop**

*Joel Neidig, Systems Engineer, ITAMCO and Dave Edstrom of Virtual Photons Electrons*

#### **MTConnect Architecture: Understanding and Building MTConnect Agents and Adapters Workshop**

*Nat Frampton, President, Real Time Development Corporation*

#### **MTConnect Hello World: Building Your First MTConnect Application Using the SDK Workshop**

*Chris Tacke, President, OpenNetCF Consulting*

## Hands-On MTConnect Software Development Webinars – [MC]2 2013

NOTE: Please go to [MTConnect.org](http://MTConnect.org) for the presentations which were being updated at the time of this document and are expected to be posted by September 1<sup>st</sup>, 2013.

### MTConnect Architecture Overview

Joel Neidig, Systems Engineer, ITAMCO and Dave Edstrom of Virtual Photons  
Electrons

### Adapters, Adapters, Adapters

Will Sobel, President, System Insights

### Building Adapters Hands-on Lab

Will Sobel, President, System Insights

## MTConnect graphr

There is a brilliant young intern at [System Insights](http://SystemInsights.com) named [Prince](#) who wrote an MTConnect monitoring application called [MTConnect graphr](#) that he has [open sourced at github](#). Here is the url to learn more and to see how Prince addressed this at <http://tinyurl.com/MTConnectGraphr>

## How To Test Your MTConnect Client

While the MTConnect Institute provides a simulator at <http://Agent.MTConnect.org>, you could also consider going to github and setting up the MTConnect Simulator yourself. Instead of including all of the information on how to set this up in this document, the reader can go to <http://tinyurl.com/MTConnectSimulator> to learn how to do this.

## Security Concerns

The MTConnect agent can be thought of as a simple web server using http. As https is not supported, there are a variety of ways to provide secure connections. Many users are familiar with setting up a VPN or using SSH to do port forwarding. Specifics of how to do these secure connections are outside the scope of this white paper, but are easy to find on the Internet.

The reason to provide a secure tunnel for an MTConnect agent is to ensure that data is not being intercepted. While machine tool data might not sound that exciting to steal, imagine if this is for a new medical device, and stealing the data could allow for it to be reverse engineered.

## github

The MTConnect Institute stores its source code at <https://github.com/mtconnect>. There is a great deal of source code and binaries. Most of the source and binaries are for the agent and adapters, but you will see the cstream source code at <https://github.com/mtconnect/cstream> as an example of an MTConnect application written in C.

The screenshot shows the GitHub profile for the organization 'mtconnect (MTConnect Institute)'. The header includes the GitHub logo, a search bar, and navigation links for 'Explore', 'Gist', 'Blog', and 'Help'. The organization's name and logo are prominently displayed. Below this, the URL 'http://www.mtconnect.org/' and the member since date 'Sep 18, 2009' are shown. The profile statistics indicate 15 Public Repos and 2 Members. The 'Public Repositories (15)' section features a search bar and a list of repositories. Three repositories are visible: 'cppagent' (C++ Agent toolkit, last updated a day ago), 'schema' (Schema source and generator script, last updated 14 days ago), and 'adapter' (C++ adapter, last updated 7 days ago). The 'Organization Members (2)' section lists 'johnmichaloski (John Michaloski)' with 5 Public Repositories and 0 followers, and 'wsobel (William Sobel)' with 1 Public Repository and 1 follower. Each repository entry includes a commit history chart showing 'all commits' and 'commits by owner' over a 52-week period.

## Where To Go For Help

The MTConnect Forum, at <http://MTConnectForum.com>, is the right place to post your MTConnect questions on any MTConnect topic. Specifically for software development, you can find the Applications section at: <http://mtconnectforum.com/forum/Forum6.aspx>

If you do not get an answer there, drop me a note and I will either respond with the answer, or research the question and then reply.

**If you have any comments or suggestions on this document, those should be sent to me: [DavidAllenEdstrom@gmail.com](mailto:DavidAllenEdstrom@gmail.com)**