MTConnect Application Development

Leveraging Web Technologies

enabling digital manufacturing

MC²

Conference · Workshop · Expo
Background

• Benjamin Kiefer
  – University of Waterloo B. Asc – Mechatronics
  – Pratt & Whitney – Controls Software

• Jared Evans
  – University of Waterloo B. Asc
  – Apple Inc – HW Engineering Manager

• MAJiK Systems
  – Dynamic Real-Time Web Applications for Manufacturers
Why Use Web Applications in Manufacturing

• Available to everyone within your organization

• Operating System Independent

• Reliable, real-time information
Why MTConnect is a Good Fit

• Based off of HTTP Protocol

• RESTful Interface

• XML document schema to represent data
Structure of a Web Application

- Clients
- Application Server
- Data/API Server
- Database
- Other Data Sources
Data Sources

• Traditional Web Applications:
  – People are the source of all data

• Emerging Trends:
  – Web-Enabled Devices, Servers, and ‘Agents’ provide data as well as people
  – API Design becomes a huge factor for anyone creating a scalable web app
HTTP Protocol

• Foundation of the World Wide Web
• Application Level Protocol
• Uses TCP for its transport layer protocol
• Request-Response Based Protocol
• Uses a Client-Server Model
RESTful Interface

Representational State Transfer

• Stateless – Client’s responsibility to track its own state
• Uniform Interface – Resources and Information always accessed the same way
• Scalability and Performance – Separation of data and User Interface
HTTP REST Example

https://www.facebook.com/search/bwkiefer/friends

https://www.google.ca/search?q=mtconnect+institute
MTConnect Example

http://agent.mtconnect.org/current?path=//Controller
Developing with REST

- Language Independent
- Distributes Computing Load
- Standardizes API and Data Representation for faster development
Developing Using MTConnect

• Your application is the ‘Client’, the MTConnect Agent is the ‘Server’
• Your Application requests data from the MTConnect Agent based on your needs
• Agent responds with requested data
• Your application parses data and completes necessary actions
MTConnect Overview

MTConnect Standard

Agent

Adapte

CNC Machine

Adapte

CNC Machine

Adapte

Sensor

Adapte

HMI

Data Flow
MTConnect Basics

- **Probe** – Describes Agent’s Devices, Components, and Data Items
- **Assets** – Things associated with a device that are not a component
- **Sample** – Retrieves values for components’ Data Items
- **Current** – Retrieves current values for components’ Data Items
Data Hierarchy within the Agent
Configuring an Agent

- Source Code - https://github.com/mtconnect/cppagent

- Agent.cfg – Boost C++ File Format that tells the agent where the adapters it is connecting to should be located

- Devices.xml – Same format as an MTConnectDevices document. Data served by adapter should match a DataItem tag in Devices.xml
Components and Data Items

Adapter Data Output:

2013-05-13T16:00:05.0000Z|mode|AUTOMATIC

Agent DataItem:

<DataItem type="CONTROLLER_MODE" category="EVENT" id="p2" name="mode"/>

Agent tries to map Adapter data to a DataItem that has an **ID**, **Name**, or **Source** that matches Adapter Key
Running an Agent from your Command Line

• Build MTConnect Agent

• Agent.cfg

• Devices.xml

• Run Mtconnect Agent

• Try http://127.0.0.1:5000/probe
What Data Are You Interested In?

• Sample – Values read from the Device at a certain time

• Event – State or Message from the Device

• Condition – Device’s Health/Ability to Function
<MTConnectDevices>
  <Header></Header>
  <Devices>
    <Device>
      <Axes>
        <Rotary>...</Rotary>
        <Linear>...</Linear>
      </Axes>
    </Device>
  </Devices>
</MTConnectDevices>
Probing The Agent

GET Request:
http://agent.mtconnect.org/probe

200 OK

<MTConnectDevices ...
  <Header>...</Header>
  <Devices>
    <Device name="mill">...</Device>
    <Device name="lathe">...</Device>
  </Devices>
</MTConnectDevices>

Your Application

MTConnect Agent

Store Devices the Agent Returns in Device Queue
Probing Individual Devices

For Each Device in Device Queue:
GET Request:
http://agent.mtconnect.org/<deviceName>/probe

200 OK

<MTConnectDevices ...
<Header>...</Header>
<Devices>
  <Device name="deviceName">...<Device>
</Devices>
</MTConnectDevices>
GET Request:
http://agent.mtconnect.org/<deviceName>/current

200 OK

<MTConnectStreams …>
  <Header nextSequence="123">…</Header>
  <Streams>
    <DeviceStream name="device">…</DeviceStream>
  </Streams>
</MTConnectStreams>

Device Queue

Your Application

Store ‘nextSequence’ For Each Device in Queue
Sample

GET Request:
http://agent.mtconnect.org/<deviceName>/sample?from='124'&count=100

200 OK

<MTConnectStreams ...>
  <Header nextSequence="159">...</Header>
  <Streams>
    <DeviceStream name="device">...<DeviceStream>
  </Streams>
</MTConnectStreams>

Device Queue

Your Application

Store ‘nextSequence’ For Each Device in Queue
Sample

GET Request:
http://agent.mtconnect.org/<deviceName>/sample?from='160'&count=100

<MTConnectStreams ...>
  <Header nextSequence="201">...</Header>
  <Streams>
    <DeviceStream name="device">...<DeviceStream>
  </Streams>
</MTConnectStreams>
Streaming ‘Real-Time’ with MTConnect

http://agent.mtconnect.org/sample?interval=0&heartbeat=1000

- Interval – Send data every 0 ms (if data is available)

- Heartbeat – If there is no data available, the agent must send out a heartbeat to maintain contact with the client every 1000 ms (10 seconds is default)
Fault Tolerance

- Applications can be made fault tolerant by persisting ‘nextSequence’ number.

- If application is disconnected from Agent, it can pick up from where it left off after re-establishing connection.

- Agent’s Buffer has finite memory. Like other application level protocols, if Agent’s Buffer is exceeded during period that application is disconnected, information is permanently lost.
Where to Send Data

Real Time Data can be sent directly to Clients through Web Sockets or other technology.

Historical Data can be archived to your database to be retrieved and rendered when requested by clients.
Node-Red Demo