Adapters – Part 1
Before you start writing code…

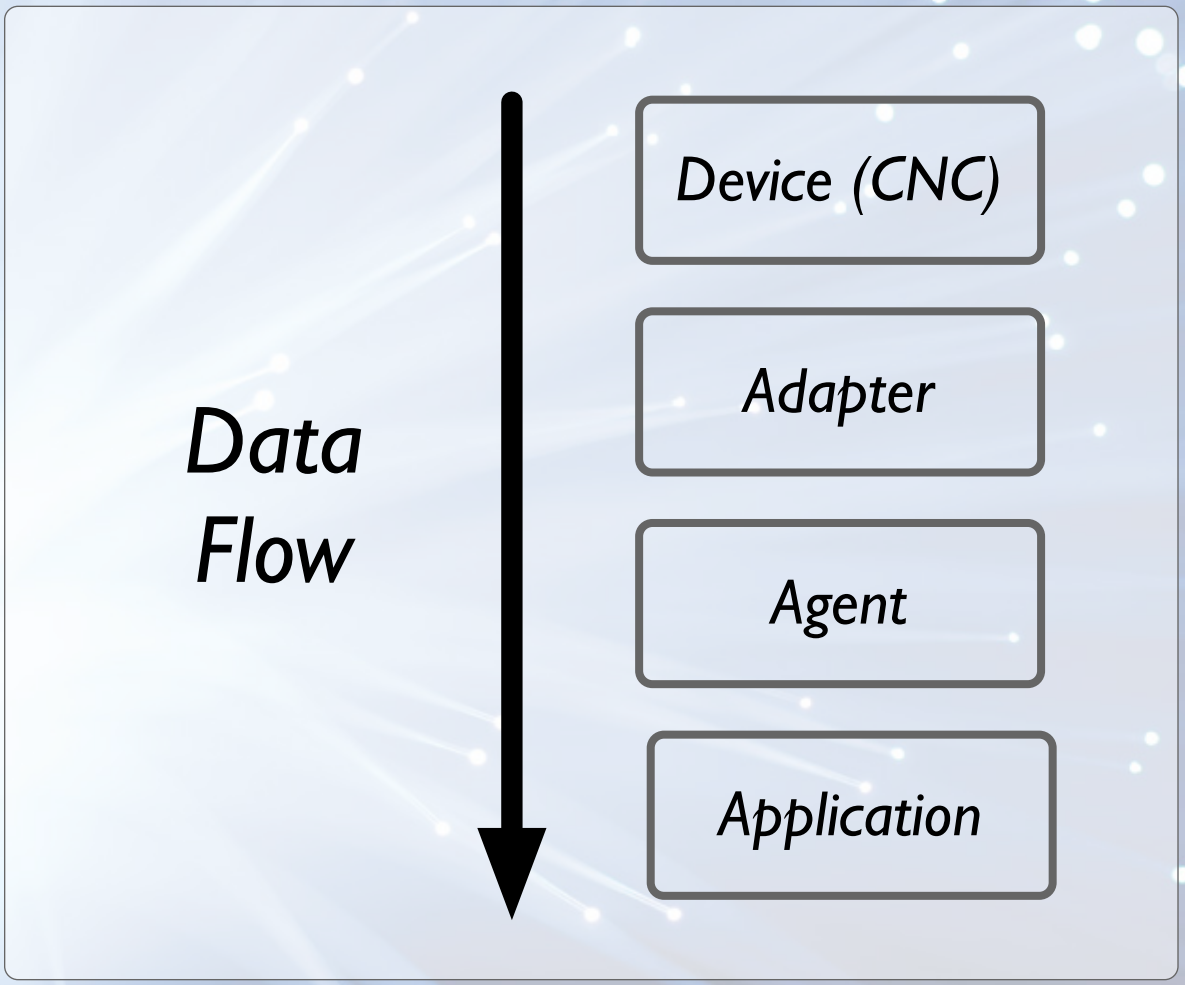
Will Sobel
System Insights

MTConnect®: Connecting Manufacturing Conference

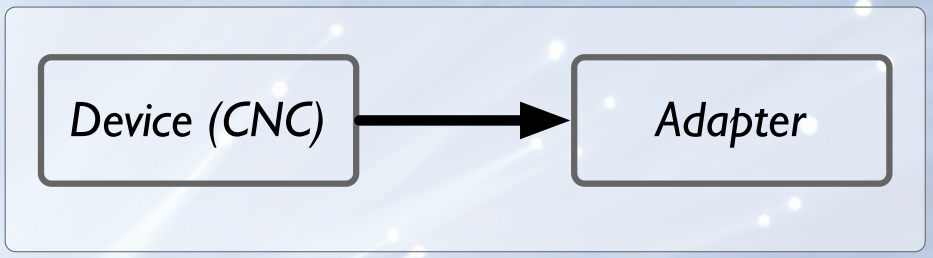
Agenda

- MTConnect Adapter Overview
- Why do we need adapters?
- Methodology
- Data Formats
- Conditions
- Time series

MTConnect Overview

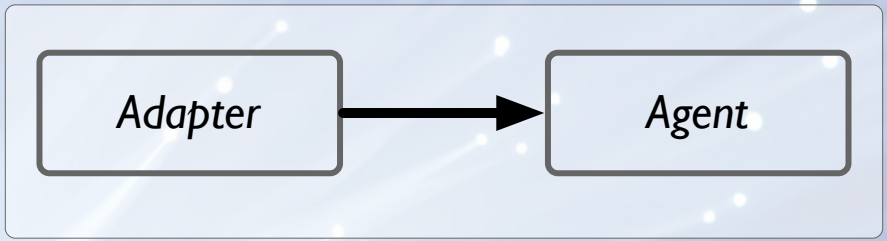


First Steps

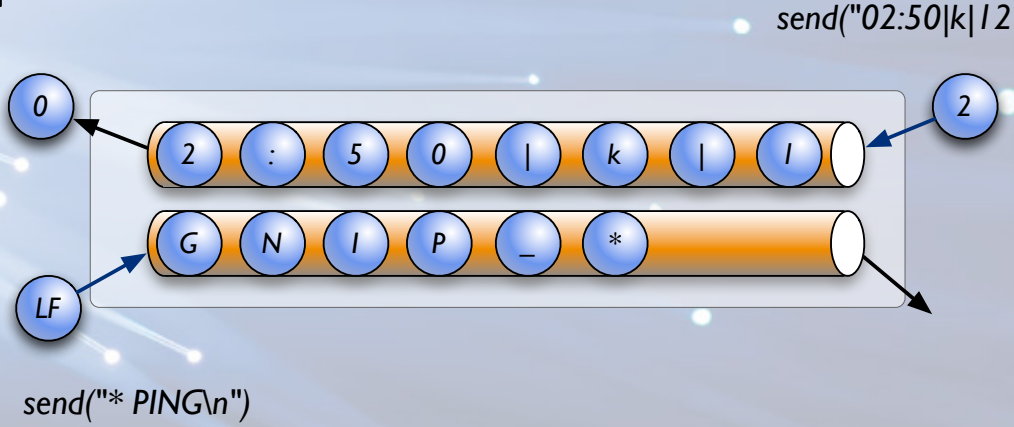


- Machine Tools, PLCs, and Sensors expose various interfaces
- Connection can be within the OS, over TCP/IP, or Serial RS-232, RS-485, etc…
- Different for every controller and vintage
- You'll spend 90% of your time getting data

Socket Connection



- Adapters use sockets to communicate
- Sockets are the generic term for an inter-process connection
- We use TCP/IP connections

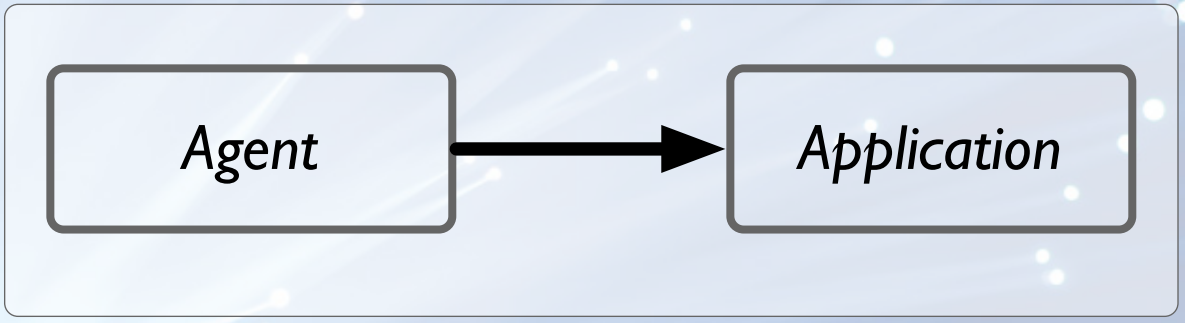


The Data

```
2009-06-15T00:00:00.000000|power|ON|execution|ACTIVE|line|412|
Xact|-1.1761875153|Yact|0.1766618937|Zact|-0.1000000015|
Xcom|-1.1750614363|Ycom|0.1837732914|Zcom|-0.1000000000|
spindle_speed|3400.0000000000|path_feedrate|0.4000000000|program|
flange_cam.ngc|mode|AUTOMATIC|block|x-1.168526 y0.225046|feed_ovr|
100.0000000000|SspeedOvr|100.0000000000|estop|ARMED|avail|
AVAILABLE
2009-06-15T00:02:19.575164|htemp|WARNING|HTEMP|1|HIGH|Oil
Temperature High
```

- The data is simple
  - Timestamp|name|value|name|value
  - Timestamp|name|level|code|native severity|qualifier|text
  - We'll handle time series and assets later…

And Finally



- HTTP is built on Sockets as well and uses TCP/IP as the protocol to communicate
- The agent receives request in HTTP format and replies using XML – that's another session…

Why Do We Need Adapters?

Adapters are not part of the MTConnect Standard!

Ok, then…

- Adapters provide more deployment options
- We can use a generic Agent for all devices
- Makes them easy to write
- Clear separation of responsibilities:
  - Adapter: Collect data from one device and write as text – filter duplicates
  - Agent: Collect data from adapters and implement HTTP requests – format XML to standard

Prerequisites

- 5 Skills you will need:
  - Format Time in UTC
    - YYYY-MM-DDTHH:MM:SS.FFF
  - Create a TCP Server Socket and Listen for a Client
  - Read and Write to the Socket
  - I assume you can get data from a device…
  - There is no fifth skill…

Even Simpler

- You will not need to even master those skills
- There are frameworks in C++, C#, Ruby, and Python
- Tomorrows Adapter Lab will use C# Framework

Framework Functionality

- Frameworks take care of the following
  - All communications and protocols
  - All data formatting
  - Checking if data has changed
  - Support for Events, Samples, Conditions, Time Series, and Assets

Your Responsibility

- Determine you controller configuration
- Create data items for each thing you want to measure
- Gather data periodically or on an event callback
- Send changed values

Simple Part is Over!

- Review – collect data and send changes
- Gathering data from your favorite device…
- I've done dozens of these, so here's my approach:
  - Controller & Paths
  - Axes
  - Systems
  - Cutting Tools

Make it Dynamic!

- Get the list of axes and paths from the controller first
- Configure the adapter to dynamically get data from these components
- Now we have a dynamic adapter – match the agents configuration to what we discover and it works across machine tool configurations
- The open source FANUC adapter is an example of self configuring multipath and axis

Controller & Paths

- Priorities for a CNC
  1. Controller Mode
  2. Execution
  3. Alarms → Conditions & Messages
  4. Program Name
  5. Overrides - Path Feedrate and Spindle Speed
  6. Part Count
  7. Path Positions
  8. Path Feedrate
  9. Line & Block

Axes & Spindles

- Priorities
  1. Positions & Angle
  2. Spindle Speed
  3. Loads
  4. Alarms
  5. Temperature

Systems

- These are usually a few conditions associated with a few special alarms
  - Coolant
  - Hydraulics
  - Pneumatics
  - Electrical

The Hard Part

- Treasure hunt… let's play find the data
- Let's use FOCAS since we have an open source version of the adapter:
  - Controller Mode
- We will use the `ODBST status` structure and the `cnc_statinfo` function

Now to Map…

- The aut byte has the following meanings:
  - status.aut == 5 or 6 means we're in manual mode
  - status.aut == 0 or 3 means were in MDI or EDIT
  - Otherwise we're in AUTOMATIC
- We now have some basic Controller Modes mapped for fanuc

Let's try another…

- Heidenhain
- We'll use the LSV2RUNINFO runInfo structure and the LSV2ReceiveRunInfo function call
- We can now map from the runInfo.ri.ExecutionMode
  - LSV2_EXEC_MDI → Manual Data Input
  - LSV2_EXEC_SINGLESTEP → Semi Automatic
  - LSV2_EXEC_AUTOMATIC & SMART → Automatic
  - All others → Manual

Community

- Wiki for implementation and behavior
- Example:
  - Current best practice for EDIT mode is MANUAL (will have explicate EDIT and BACKGROUND in future versions)
  - When motion is HOLD or Wait, then Execution is INTERRUPTED
  - Etc…

Alarms and Conditions

- Special handling required
- A few things to understand
  - Multiple conditions can be active for the same type at the same time
  - Conditions are unique by type and native code
    - Implementation decision
  - One or more conditions can be cleared at the same time

Representation

- Conditions are placed on one line
  - So are messages and time series data
- The have the following fields
  - Name
  - Level – NORMAL, WARNING, or FAULT
  - Native Code
  - Native Severity
  - Qualifier – HIGH or LOW
  - Text

How to Handle Alarm Lists

- Tools implement Mark-and-Sweep to collect alarms that are no longer active
- For every alarm in the list add it to the condition of that type
- If an alarm is not added, it is cleared
  - An add marks active alarms, the sweep find all alarms that are not marked and removes

FOCAS 2 Example

```
for (int i = 0; i < 31; i++)
{
  if (aAlarm & (0x1 << i))
  {
    ODBALMMSG2 alarms[MAX_AXIS];
    short num = MAX_AXIS;

    short ret = cnc_rdalmmsg2(aFlibhndl, i, &num, alarms);
    if (ret != EW_OK)
      continue;

    for (int j = 0; j < num; j++)
    {
      ODBALMMSG2 &alarm = alarms[j];
      char code[16];

      Condition *cond = translateAlarmNo(i, alarm.axis);
      if (cond == NULL)
        continue;

      sprintf(code, "%d", alarm.alm_no);
      cond->add(Condition::eFAULT, alarm.alm_msg, code);
    }
  }
}
```

Native Code	Message
416	Gen Fault 1
912	Gen Fault 2
649	Gen Fault 3

Delta

Native Code	Message
416	Gen Fault 1
912	Gen Fault 2
649	Gen Fault 3

Native Code	Message
416	Gen Fault 1
649	Gen Fault 3
214	Gen Fault 4

- Each time we evaluate which code are still active, newly active and no longer active
- In this example, 912 is removed and 214 is added

- …|system|NORMAL|912|||
- …|system|FAULT|214|||Gen Fault 4

Continued

Native Code	Message
416	Gen Fault 1
649	Gen Fault 3
214	Gen Fault 4

Native Code	Message

- When all the alarms are cleared, a NORMAL is sent to clear all
- …|system|NORMAL||||

Simple Conditions

- If the data source sends an event when the alarms starts and stops you can use a simple condition
- A simple condition requires an explicate clear when the condition is no longer active

Time Series

- Real time data collected at a fixed frequency
- Data is represented as a list of numbers followed by a space
- Also placed on a single line like conditions
- Fields
  - Name
  - Count
  - Rate
  - Values – ex. 9325..166 54321.13555 23.09123

Time Series Handling

- Time stamp is always set to the time the **LAST** sample was taken
- To compute the time of the first, multiple rate time count and subtract from the the timestamp
- Rate is given in Hertz (samples / second)

Example

time	name	count	rate					
T13:00:12.10	ia	5	10	16	24	12	66	18

- This is a time series with 5 items at 5 Hz.
- The sample was taken at 13:00:12.10, so since we have 5 at 10/second, the duration is ½ second
- The series started at 13:00:11.60
- The data will be represented like this:
  - `…T13:00:12.10|ia|5|10|16.0 24.0 12.0 66.0 18.0`
- The rate is options if it is fixed and has been provided in the DataItem defined in the Devices.xml

Relative Time

- New feature to handle sensors without a wall-clock time
- Instead of giving a timestamp provide a relative clock tick in milliseconds
- Agent will use it's own time and then compute the offsets based on the relative time
- Maintains consistent spacing between samples and allows for analysis

Relative Time Example

- 1456|m|5||1 2 2 4 5
- 1556|m|5||4 1 4 2 3
- 1656|m|5||2 2 1 5 4

- Each sample is 100ms apart, the Agent will create timestamps with an exact 100ms spacing

Assets

- Cutting Tool is currently the only asset we support, but others can be handled by the agent as well
- The agent now supports a multi-line document for assets
- Send XML document for the asset as the content (can be multiline)

Asset Representation

- Use the special name of @ASSET@ to signify an asset is to follow
- Next the Asset ID
  - All assets have a unique id
- Specify the type: “CuttingTool”
- And the data…

Multiline

- Asset example:

```
…|@ASSET@|AAA123|CuttingTool|--multiline--ABCD
<CuttingTool serialNumber="1" toolId="KSSP300R4SD43L240"
timestamp="2011-05-11T13:55:22"
assetId="KSSP300R4SD43L240.1" manufacturers="KMT,Parlec">
 <CuttingToolLifeCycle>
    <CutterStatus><Status>NEW</Status></CutterStatus>
    <Measurements>
       <BodyDiameterMax code="BDX">73.25</BodyDiameterMax>
       <OverallToolLength nominal="323.85"
minimum="323.596" maximum="324.104" code="OAL">323.86</
OverallToolLength>
    </Measurements>
 </CuttingToolLifeCycle>
</CuttingTool>
--multiline--ABCD
```

Alternative

- All data can be written to one line
- Example:

```
…|@ASSET@|AAA123|CuttingTool|<CuttingTool serialNumber="1"
toolId="KSSP300R4SD43L240" timestamp="2011-05-11T13:55:22"
assetId="KSSP300R4SD43L240.1" manufacturers="KMT,Parlec">…
```

- This format is used for both adding and updating assets

Adding Other Assets

- Specify another asset type and provide the full asset document
- Read Part 4 for full details on XML formats
- Cutting Tool is parsed and reformatted by the Agent to ensure proper representation

Protocol

- Heartbeats
  - * PING responded to with * PONG <frequency>
- Makes sure the connection stays open
- If adapter or machine becomes unresponsive, agent can disconnect
- If agent becomes unresponsive, adapter can disconnect
- Gracefully handles network issues
- Heartbeats are optional

For Tomorrow…

- Please make sure you have the latest download of the AdapterLabMaterial.zip
- Checkout the latest instructions at:
  - https://github.com/mtconnect/mc2-adapter-lab/wiki

[MC]²

Text

MTConnect®: Connecting Manufacturing Conference

MTConnect Institute