



Common Industrial Cloud Interface – Uses Cases and Technical Requirements for Data Transfer

Stephen C. Briant
Rockwell Automation

Tom Whitehill
Schneider Electric

February 22, 2017

Agenda

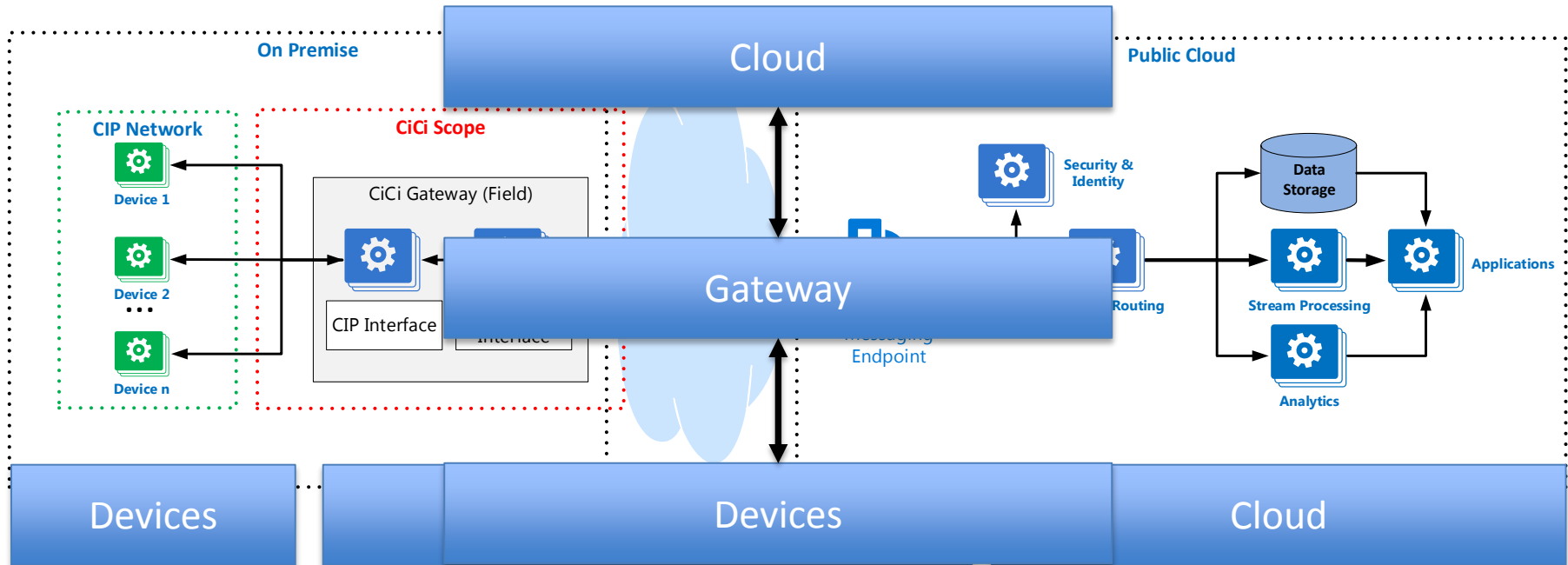
- Introduction
- Reference Architecture
- Cloud Technologies
- Guiding Principles
- Information Exchange Patterns
- Use Cases
- Proof-Of-Concept
- Conclusion – Next Steps

Introduction

- The Industrial Internet of Things is here, bringing new technologies, challenges and opportunities to industrial automation
- Companies are looking to the internet and cloud computing to provide new ways to improve operations and increase productivity as well as addressing concerns of an aging workforces
- Cloud computing offers many advantages that were previously unavailable, starting with the ability to connect to devices across an enterprise or a machine type across multiple enterprises. In addition, the ability to scale computing power and storage are enabling new possibilities for analyzing data streams.
- Acquiring data from devices is the primary focus in the market today, but there are definitely opportunities to do more
- In April, 2016, ODVA announced the formation of a new Special Interest Group (SIG) for the Common Industrial Cloud Interface (CICI) to address these opportunities.
- This new SIG intends to leverage technologies available in cloud platforms and “connect” them with the rich information defined in CIP Devices in a simple and secure manner.

Reference Architecture

Reference Architecture



Cloud Technologies

- **AMQP**

Advanced Message Queuing Protocol, is the open standard and has emerged as a very popular protocol for sending messages to and receiving messages from Cloud-based systems. In addition to being Open and Standard, AMQP was designed with these characteristics Security, Reliability, Interoperability.

- **MQTT**

Message Queue Telemetry Transport, is an ISO standard (ISO/IEC PRF 200922), publish/subscribe, lightweight messaging protocol used for Cloud-connectivity for limited network bandwidth, remote applications.

- **JSON**

JavaScript Object Notation, is a terse, readable, structured data format. It is very popular as a payload format for Device-to-Cloud and Cloud-to-Device messaging. A benefit to using JSON is that many stream processing applications are built to natively consume JSON structures efficiently and cost-effectively. Below is a very basic JSON message:

```
{  
  "name"="CICI",  
  "message"="Hello World!"  
}
```

Guiding Principles

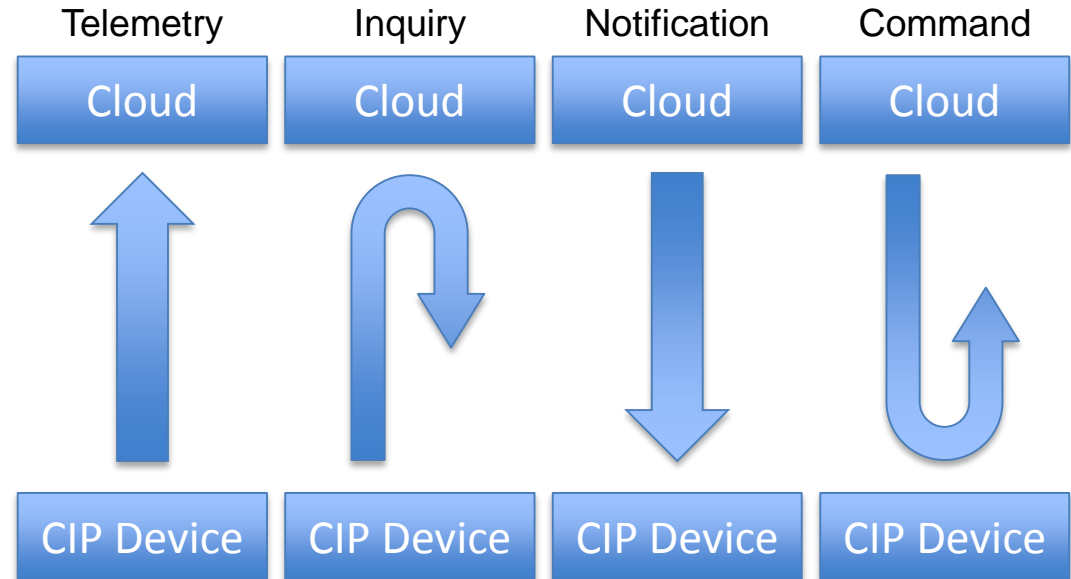
- There are fundamental differences in Cloud-based vs CIP-network based application development ... a few are summarized here:
 - Distribution: Cloud-based applications are intrinsically distributed, combining resources from multiple compute, storage and service platforms to achieve function.
 - Real-Time: due to its distributed nature and platform dependencies, the notion of real-time is an uncommon concept in public Cloud computing.
 - Protocols and Payloads: Message payloads that are easily programmatically digestible and extensible
 - Cloud communication protocols are generally widely used, open standards which are not industry specific and which may be replaced at any time
 - CIP communications, while an open standard, is industry specific and slow to change.
 - Application scope and lifecycle: Cloud-based applications can have very broad scope, virtually no limitations and can be modified or updated very rapidly

Guiding Principles

- Security
 - all communication must be initiated from the device or gateway.
- Performance
 - Communication must be performant and scalable across the Reference Architecture, therefore CIP must “stay home” or stay on premise
- Four V’s of Big Data
 - Volume (must handle the scale of data)
 - Variety (must handle different forms of data)
 - Velocity (must handle the speed required for analysis of the data)
 - Veracity (must handle the uncertainty or quality of the data)

Information Exchange Patterns

- **Telemetry**
 - Device > Cloud
 - One Way
- **Inquiry**
 - Device > Cloud w/Response
 - Two Way
- **Notification**
 - Cloud > Device
 - One Way
- **Command**
 - Cloud > Device w/Response
 - Two Way



Use Cases – Device Lifecycle

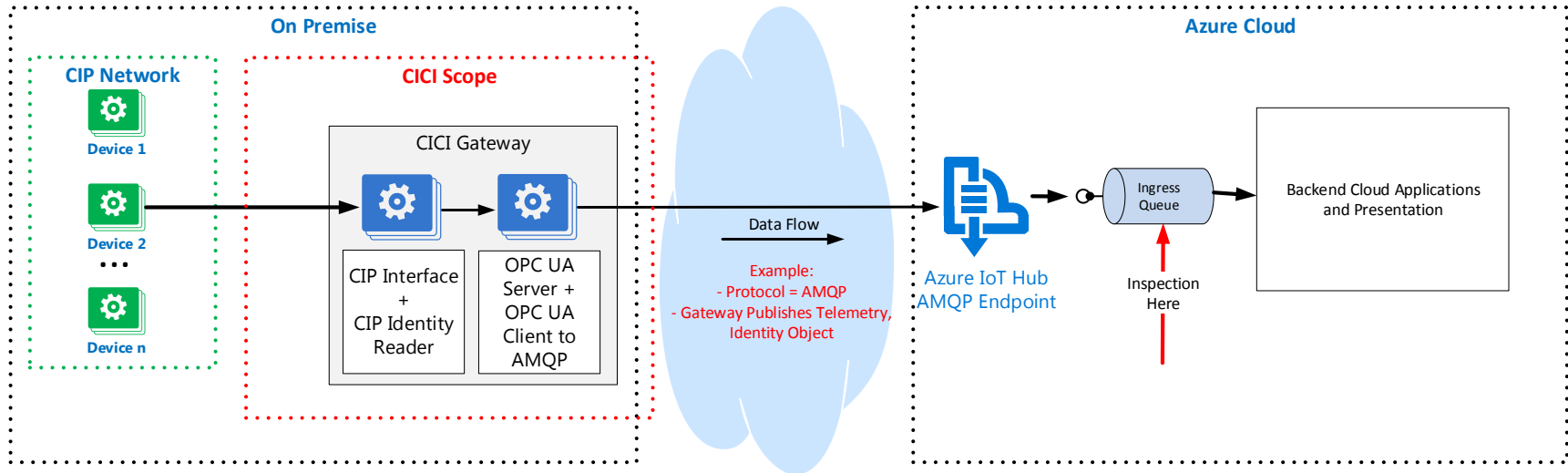
- Commissioning
 - Out-of-the-box definitions (Inquiry, Command)
 - Cloud Registration / backend business setup (Inquiry, Command)
 - On-boarding/Provisioning (Inquiry, Command)
 - Context of device in application (Inquiry, Command)
 - Control/Application loading (Inquiry, Notification, Command)
- Operating
 - Monitoring (Telemetry)
 - Maintenance (Telemetry, Command)
 - Calibration (Inquiry, Command)
 - Diagnosis (Telemetry)
 - Enable/Disable (Command)
 - Optimization / Changing Parameters / Programs (Telemetry, Command)
 - Software updates (Inquiry, Notification, Command)
 - Device Replacement (Inquiry, Command)
- Decommissioning
 - Removing a device (Telemetry, Command)

Proof-Of-Concept

- **Telemetry Use Case**
 - A cloud application needs a list of CIP Devices
 - A cloud application needs a list of product names of CIP Devices
 - A cloud application needs the firmware version of a CIP Devices
- **Goal**
 - Set up a simple application that accomplishes the Telemetry use case
 - Leverage open standards were possible
 - Leverage existing code and content to expedite efforts
- **Results**
 - Used a prototype CIP Stack and CIP Scanner in Node.JS
 - Used an open source OPC UA Server in Node.JS
 - Used Microsoft's open source C# application to read OPC UA Server, connect to Azure IoT Hub via AMQP
 - Defined an Azure IoT Device on Azure IoT Hub
 - Used Device explorer on Azure to view AMQP message stream

Proof-Of-Concept

Reference Architecture: Proof-Of-Concept



CIP Stack/CIP Scanner

- Load application into Node.JS

```
C:\temp\OPCUA\node-opcua>node MyServer.js
initialized
{"name":"cip_enip_udp","hostname":"NAUSSEW6MXRG12","pid":3800,"level":50,"msg":"Socket failed, Error: bind EADDRINU
0.0.0:44818","time":"2017-02-10T08:38:36.033Z","v":0}
Server is now listening ... < press CTRL+C to stop>
opc.tcp://NAUSSEW6MXRG12.RA-INT.COM:1234 NONE http://opcfoundation.org/UA/SecurityPolicy#None
opc.tcp://NAUSSEW6MXRG12.RA-INT.COM:1234 SIGN http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15
opc.tcp://NAUSSEW6MXRG12.RA-INT.COM:1234 SIGN http://opcfoundation.org/UA/SecurityPolicy#Basic256
opc.tcp://NAUSSEW6MXRG12.RA-INT.COM:1234 SIGNANDENCRYPT http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15
opc.tcp://NAUSSEW6MXRG12.RA-INT.COM:1234 SIGNANDENCRYPT http://opcfoundation.org/UA/SecurityPolicy#Basic256
disc: [ ]
```

- Print output of discovered CIP Devices are loaded into OPC UA Server

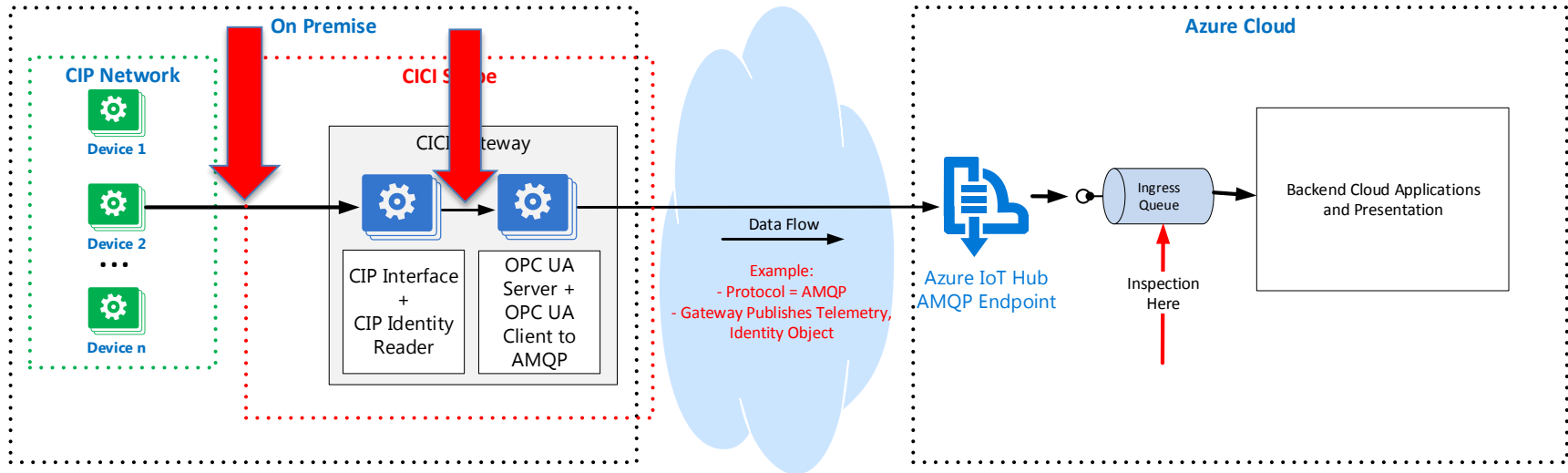
```
disc: [ { identity:
  { vendor_id: 1,
    device_type: 12,
    product_code: 166,
    major_revision: 10,
    minor_revision: 6,
    status: 48,
    serial_number: 10896684,
    product_name: '1756-EN2T/D',
    ip_address: '10.88.53.1' } } ],
```

```
{ identity:
  { vendor_id: 1,
    device_type: 14,
    product_code: 168,
    major_revision: 29,
    minor_revision: 11,
    status: 12384,
    serial_number: 12763990,
    product_name: '1756-L85E/B',
    ip_address: '10.88.53.3' } },
```

```
{ identity:
  { vendor_id: 1,
    device_type: 143,
    product_code: 2192,
    major_revision: 12,
    minor_revision: 2,
    status: 97,
    serial_number: 541012646,
    product_name: 'PowerFlex 755',
    ip_address: '10.88.53.9' } } ]
```

Proof-Of-Concept

Reference Architecture: Proof-Of-Concept



Setup OPC UA Client to OPC UA Server

- OPC UA Server address
- Node to start data transfer

```

Module.cs  Opc.Ua.Client.Sam... Module.Config.xml  #  X  gateway_config.json  main.c
164  and keep alive count are calculated using this value and the request publishing in
165  <MinSubscriptionLifetime>10000</MinSubscriptionLifetime>
166
167  </ClientConfiguration>
168
169  <Extensions>
170    <ua:XmlElement>
171      <ListOfPublishedNodes xmlns="http://opcfoundation.org/UA/SDK/Configuration.xsd">
172        <NodeLookup>
173          <ua:EndpointUrl>opc.tcp://NAUSMAY36CSR72.RA-INT.COM:1234</ua:EndpointUrl>
174          <ua:NodeId>
175            <!-- Current Server Time -->
176            <ua:Identifier>ns=1;s=Identity</ua:Identifier>
177          </ua:NodeId>
178        </NodeLookup>
179        <NodeLookup>
180          <ua:EndpointUrl>opc.tcp://NAUSMAY36CSR72.RA-INT.COM:1234</ua:EndpointUrl>
181          <ua:NodeId>
182            <!-- Current Server Time -->
183            <ua:Identifier>ns=1;s=Identity</ua:Identifier>
184          </ua:NodeId>
185        </NodeLookup>
186      </ListOfPublishedNodes>
187    </ua:XmlElement>
188  </Extensions>
  
```

Set up OPC UA Client to Azure IoT Hub

- Set up path to Azure
- Name of endpoint device
- Name for event hub
- Selection of transport

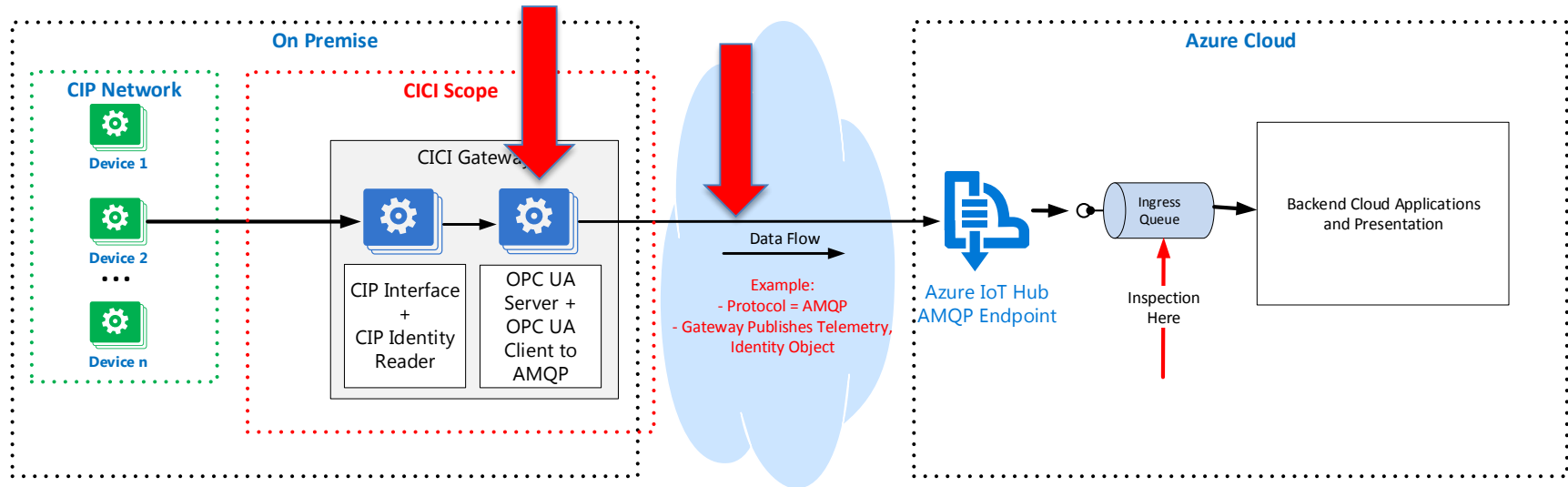


```
Module.cs      Opc.Ua.Client.SampleModule.Config.xml  gateway_config.json  main.c
Schema: <No Schema Selected>

1  {
2    "modules": [
3      {
4        "name": "opc_ua",
5        "loader": {
6          "name": "dotnet",
7          "entrypoint": {
8            "assembly.name": "Opc.Ua.Client.SampleModule",
9            "entry.type": "Opc.Ua.Client.SampleModule"
10         }
11       },
12       "args": "opcUA6777;ofSU0gnrEhjaXs8ea7E3gkB2jx/Utb3JL4S57YDSWY4="
13     },
14     {
15       "name": "IoTHub",
16       "loader": {
17         "name": "native",
18         "entrypoint": {
19           "module.path": "iothub.dll"
20         }
21       },
22       "args": {
23         "IoTHubName": "IOTtest6777",
24         "IoTHubSuffix": "azure-devices.net",
25         "Transport": "AMQP"
26       }
27     }
28   ]
29 }
```

Proof-Of-Concept

Reference Architecture: Proof-Of-Concept



Observe messages on Azure IoT Hub

Device Explorer

Configuration Management Data Messages To Device

Monitoring

Event Hub: IOTest6777

Device ID: opcUA6777

Start Time: ☒ 02/09/2017 11:25:37

Consumer Group: \$Default ☐ Enable

Monitor Cancel Clear

```
disc: [ { identity:
  { vendor_id: 1,
    device_type: 12,
    product_code: 166,
    major_revision: 10,
    minor_revision: 6,
    status: 48,
    serial_number: 10896684,
    product_name: '1756-EN2T/D',
    ip_address: '10.88.53.1' } },
```

Event Hub Data

```
'source': 'mapping'
'content-type': 'application/opcua+uajson'

2/9/2017 12:02:33 PM> Device: [opcUA6777], Data [{"HostName": "nausmay36csr72-ra-
int.com", "MonitoredItem": {"Id": "s=Identity", "Uri": "urn:NodeOPCUA-Server-default"}, "ClientHandle": 2, "Value":
{"Value": [{"identity": {"vendor_id": 1, "device_type": 12, "product_code": 166, "major_revision": 10,
"minor_revision": 6, "status": 48, "serial_number": 10896684, "product_name": "1756-EN2T/D", "ip_address":
"10.88.53.1"}}, {"identity": {"vendor_id": 1, "device_type": 14, "product_code": 168, "major_revision": 29,
"minor_revision": 11, "status": 12384, "serial_number": 12763990, "product_name": "1756-L85E/B",
"ip_address": "10.88.53.3"}}, {"identity": {"vendor_id": 1, "device_type": 143, "product_code": 2192,
"major_revision": 12, "minor_revision": 2, "status": 97, "serial_number": 541012646, "product_name":
"PowerFlex 755", "ip_address": "10.88.53.9"}}, {"SourceTimestamp": "2017-02-
09T17:02:29.184Z", "ServerTimestamp": "2017-02-09T17:02:29.184Z"}]] Properties:
'source': 'mapping'
'content-type': 'application/opcua+uajson'
```

Conclusion – Next Steps

- The Common Industrial Cloud Interface SIG was formed to enable cloud applications to have access to the valuable information available in CIP Devices
- There are many opportunities or use cases; however, the most obvious set of use cases are grouped around managing the lifecycle of CIP devices
- The Common Industrial Cloud Interface SIG will use guiding principles and Information Exchange Patterns to flush out functionality needed for these use cases
- The next step will be map needed functionality to available technologies and standards, including those shown in the Proof-Of-Concept
- If you would like more information or want to contribute, please consider joining the Common Industrial Cloud Interface SIG!