Enabling Data Scientist Use Cases with Discoverability and Metadata

Greg Majcher
Rockwell Automation
“Data is the new oil”
Data Science Introduces New Use Cases

Data Scientists

• What are they looking for?  It depends
• How do they want to access data?  Don’t care
• How much data?  It depends
• How often?  It depends
Data Science Workflow

1. Define/Scope the Problem
2. Identify Needed Data
3. Collect Data
4. Clean/Organize Data
5. Analyze Data
6. Create Conclusions

- Discover
- Understand
- Deliver
Example User Stories

• Identify areas for energy savings
• Predict when a component will fail
• Increase the efficiency of a process
• Identify deteriorating quality in a process or a product being produced
• Diagnose performance or quality differences between similar production lines or facilities
How Can We Help

• Discover
  – Mechanisms to discover devices and the data they possess

• Understand
  – Metadata to help understand the data

• Deliver
  – Efficient delivery mechanisms
Discovering Devices
Discover Devices

- **LLDP**

- **ListIdentity**

  - Is From **Vendor** A, Has **Product Type** B & **Product Code** C
  - Is at **Revision** D
  - Has Current **Status** of E
  - Has a **Serial Number** F and **Product Name** “G”
  - And is currently in **State** F
What could we do better?

Learn about device capabilities during discovery

CIP Security Profiles are delimited in the ListIdentity response
We can expand this idea to include other features/capabilities
Discovering Data Online
Discovering Data Online

Using a Brute Force Technique

- Send request to all possible classes (65,535)
- Send request to all instances of classes present (4,294,967,295)
- Send request to all possible attributes (65,535)

- $65,535 + (4,294,967,295 \times C) + (65,535 \times I) = \text{MANY, many messages}$
Message Router Instance Attribute 1 eliminates 65,534 messages

<table>
<thead>
<tr>
<th>Number</th>
<th>Need in implementation</th>
<th>Access Rule</th>
<th>Name</th>
<th>Data Type</th>
<th>Description of Attribute</th>
<th>Semantics of Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Optional</td>
<td>Get</td>
<td>Object_list</td>
<td>STRUCT of</td>
<td>A list of supported objects</td>
<td>Structure with an array of object class codes supported by the device</td>
</tr>
<tr>
<td>Number</td>
<td></td>
<td></td>
<td>Number</td>
<td>UINT</td>
<td>Number of supported classes in the classes array</td>
<td>The number of class codes in the classes array</td>
</tr>
<tr>
<td>Classes</td>
<td></td>
<td></td>
<td>Classes</td>
<td>ARRAY of UINT</td>
<td>List of supported class codes</td>
<td>The class codes supported by the device</td>
</tr>
</tbody>
</table>

\[
1 \times 65,535 + (4,294,967,295 \times C) + (65,535 \times I) = \text{MANY messages}
\]
Discovering Data Online with Better Implementations

Class Attributes 2 and 3 *could* eliminate over 4 billion messages

<table>
<thead>
<tr>
<th>Number</th>
<th>Need in implementation</th>
<th>Access Rule</th>
<th>Name</th>
<th>Data Type</th>
<th>Description of Attribute</th>
<th>Semantics of Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Conditional(^2)</td>
<td>Get</td>
<td>Max Instance</td>
<td>UINT</td>
<td>Maximum instance number of an object currently created in this class level of the device.</td>
<td>The largest instance number of a created object at this class hierarchy level.</td>
</tr>
<tr>
<td>3</td>
<td>Conditional(^2)</td>
<td>Get</td>
<td>Number of Instances</td>
<td>UINT</td>
<td>Number of object instances currently created at this class level of the device.</td>
<td>The number of object instances at this class hierarchy level.</td>
</tr>
</tbody>
</table>

\[1 \times 65,535 + (2 \times 4,294,967,295 \times C) + (65,535 \times I) = \text{thousands of messages}\]
Discovering Data Online with Better Implementations

Find_Next_Object_Instance* could be used for sparsely populated lists

*Note, this service cannot be used for UDINT instances
# Discovering Data Online with Better Implementations

<table>
<thead>
<tr>
<th>Number</th>
<th>Need in implementation</th>
<th>Access Rule</th>
<th>Name</th>
<th>Data Type</th>
<th>Description of Attribute</th>
<th>Semantics of Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Optional</td>
<td>Get</td>
<td>Optional attribute list</td>
<td>STRUCT</td>
<td>List of optional instance attributes utilized in an object class implementation.</td>
<td>A list of attribute numbers specifying the optional attributes implemented in the device for this class.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>number of attributes</td>
<td>UINT</td>
<td>Number of attributes in the optional attribute list.</td>
<td>The number of attribute numbers in the list.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>optional attributes</td>
<td>ARRAY</td>
<td>List of optional attribute numbers.</td>
<td>The optional attribute numbers.</td>
</tr>
<tr>
<td>6</td>
<td>Optional</td>
<td>Get</td>
<td>Maximum ID Number Class Attributes</td>
<td>UINT</td>
<td>The attribute ID number of the last class attribute of the class definition implemented in the device.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Optional</td>
<td>Get</td>
<td>Maximum ID Number Instance Attributes</td>
<td>UINT</td>
<td>The attribute ID number of the last instance attribute of the class definition implemented in the device.</td>
<td></td>
</tr>
</tbody>
</table>
What could we do better?

New class attributes for supported instances and attributes modeled after the Message Router’s Object_List attribute
Understanding Data Online
“Metadata” is documented in the specifications, but it is not exposed online.

<table>
<thead>
<tr>
<th>Attribute ID</th>
<th>Need in Implementation</th>
<th>Access Rule</th>
<th>NV</th>
<th>Name</th>
<th>Data Type</th>
<th>Description of Attribute</th>
<th>Semantics of Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

What if these columns were reported as standardized attribute properties? We could define logical segments to retrieve them.

<table>
<thead>
<tr>
<th>Segment Contents</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>[20][01][24][01][30][03][3C 07][05]</td>
<td>Segment Type = Logical Segment. 20 01 indicates class 1 (Identity Object) 24 01 indicates instance 1 30 03 indicates attribute 3 (Product Code) 3C 07 05 indicates metadata property 5 (Name)</td>
</tr>
</tbody>
</table>
Discovering Data Offline
Discovering Data Offline

Yes, this is a valid EDS

[File]
DescText = “NotSoSmart LazyBoy Widget EDS File”;
CreateDate = 04-01-1999;  $ Create date
CreateTime = 17:51:44;
Revision = 1.1;  $ Revision of EDS

[Device]
VendCode = 65535;
VendName = “NotSoSmart, Inc.”;
ProdType = 43;
ProdTypeStr = “Generic Device”;
ProdCode = 42;
MajRev = 1;  $ Device Major Revision
MinRev = 1;  $ Device Minor Revision
ProdName = "LazyBoy Smart-Widget";

[Device Classification]
Class1 = EtherNetIP;
[Discrete Input Class]

Revision = 2; $ Revision 2 of the object is implemented
MaxInst = 8; $ The highest instance number that exists in the product is 8
Number_Of_Static_Instances = 8; $ There are 8 static instances present
Number_Of_Dynamic_Instances = 0; $ There are no dynamic instances
Class_Attributes = 1; $ Class attribute 1 is supported
Instance_Attributes = 3, 4, 5, 6; $ Instance attributes 3, 4, 5, and 6 are supported
Class_Services = 0x14; $ Get is supported for class attributes
Instance_Services = 0x14, 0x10; $ Get and Set are supported for inst attributes
Object_Name = “Discrete Input Point Object”
Object_Class_Code = 0x08;
Understanding Data Offline
Understanding Data Offline

[Params]
Param1 = 0,
6,"20 06 24 01 30 F0", $ Size, Link Path to Conn Mgr Object
0x0002, $ Descriptor
0xC7, 2, $ UINT Data Type, Data Size
"Percent I/O Utilization", $ Name
"%", $ Units
"Indicates what percentage of the I/O comms resources are in use in this device in units of 0.1%", $ Help string
0,1000,0, $ Min/Max/Default
,,,$ Unused fields
;
Delivering Data
Delivering Data

- Request / Response
- Small amounts of varied data
- Larger amounts of data
- Lots of data often

Unconnected & Class 3
Multiple_Service_Packet
Send_Receive_Fragment
New transports
Summary and Call to Action

- There is a lot we can do immediately through vendor adoption
- We can improve some of the existing definitions
- We can define better alternatives
Summary Plan

Near Term (Today)
• Fully-defined EDS files

Medium Term (Next Releases)
• Support for Message Router’s Object_List
• Support for Max Instance and Number of Instances
• Support for Param entries to define data

Long Term (Future Publication Cycles)
• Enhance existing definitions
• Define & Support online metadata
• Define & Support new transport protocols where needed
Thank You!
Questions?