Metadata for CIP Devices

Greg Majcher
Principal Application Engineer
Rockwell Automation

Michael Schaffner
Embedded Systems Engineer, Networks and Security
RA Technologies, Inc.

Presented at the ODVA
2025 Industry Conference & 23rd Annual Meeting
March 19, 2025
Clearwater Beach, Florida, USA

Abstract

One of the conclusions emerging from the xDS project is the recognition of the importance of metadata. The addition of metadata to the CIP object model and its inclusion in offline device descriptors would make it possible for tools to provide contextualized presentation of devices both off-line and on-line. Similarly, an intermediate conclusion of the Common Industrial Cloud Infrastructure (CICI) SIG has been that embedded metadata in devices enables information discovery and understanding in data science use cases.

In this paper, we will revisit the use cases for metadata in devices, present concepts for implementing metadata support and illustrations of how this can result in enhanced representation of devices.

Keywords

Metadata, Full Parameter, Stub Parameter, xDS, CIP

Definition of terms

Term	Definition
Common Industrial Protocol (CIP)	The upper layers of the communication protocol used by all CIP Networks
Electronic Data Sheet (EDS)	A file that contains device description and configuration information for a CIP-based device
Edge Device or Edge Gateway	A device or application that connects devices on a local network (control network) with an external network or the cloud
EPATH	An array of bytes whose contents convey a path to information or the relationship between objects

Full Parameter Object	A parameter object instance within a device that contains the data value, prompt strings, data conversion factors, and other information associated with the parameter
Metadata	Data that describes other data
Stub Parameter	A shorthand form of a parameter object instance that supports a limited number of attributes (i.e., the parameter’s value), but does not include things like descriptive text or data limits
STC File	A file used to document the supported CIP functionality that is to be tested for conformance to the CIP specifications
Tag Name	A human-readable name given to a piece of data. This name originated from the practice of affixing a label (tag) to signal wires to identify the signals.
xDS	An ODVA project that was established to make improvements to the definition and use of EDS files

Introduction

There is a spectrum of support for metadata in CIP products ranging from opaque blobs to full descriptions of every interesting piece of information in a device. Many product implementations leave customers struggling or don’t deliver potential value. There is no good way (today) to document where most products fall on that spectrum, but an educated guess is that most lean towards the disappointing, opaque end of the spectrum. Many implementations have dismissed the value of providing rich context to their data or actively chose not to provide it due to resource constraints. A rich set of metadata has become more valuable today as end users attempt to use our devices in new and innovative ways to improve their products and processes. New software and workflows have entered the industrial automation space and are becoming increasingly important. We should be able to justify metadata additions to our products based on the value they will bring to our users.

metadata /mĕt′ə-dā″tə, -dăt″ə, -dä″tə/
noun plural

1. Data that describes other data, as in describing the origin, structure, or characteristics of computer files, webpages, databases, or other digital resources.

noun

1. Data that describes other data, serving as an informative label.
2. Data about data. [3]

Types of Metadata

Metadata can describe the origin, structure, or any characteristics of the main subject data. It can be fixed or variable (settable). Any discussion of metadata first requires agreement on the context because the same item can be data in one context and metadata in another context. Following is a discussion of different types of metadata.

Fixed in the Product Design
Some metadata is fixed during design and will never change during the life of a product.

In CIP products we usually think of object attributes as data. Each attribute was given an attribute ID, name, description, datatype, access rule, and whether the value is stored in non-volatile memory. These specification-defined items are all metadata specified for each attribute in a CIP implementation.

However, since CIP is an object-oriented technology, related information is grouped into objects. Some object attributes are metadata for other attributes. Consider the Analog Input Point Object. The Value attribute is the center point of the object, but most of the other attributes (e.g., Status, Owner Vendor ID, Owner Serial Number, and Input Range) could be considered metadata for the Value attribute. These related pieces of information are grouped together in each instance of the object, but without intimate knowledge of the object definition, clients would not be able to distinguish the data from metadata (i.e., attribute 7, Input Range is metadata for attribute 3, Value).

Created at Application Time
Some metadata is assigned after the data has been given context (i.e., used in an application or changed in some way).

When a product is used in an application, variable or tag names are assigned to the product data to provide easy association between the product data and its use within the application. These human readable names are essential for design and support personnel to understand that the products are configured and programmed correctly.

Entered by User
Some products support the ability for users to enter installation-specific information directly into the product. Metadata like the description of a program variable, or the physical location within the plant can be stored on the product and used to support maintenance or asset management activities.

Additional Data
Our products all have data directly associated with their main function. Drives convert electrical energy to motion and therefore have attributes indicating input and output voltages, torque or speed references. IO modules have attributes representing the value of physical signals. However, most products also have additional data that can be used to troubleshoot issues with the product or the application. Things like CPU Utilization, internal or ambient temperature, hours of operation, etc. are all examples of data that is in addition to the data associated with the product's core function. This additional data can be used in the context of the product, the machine or application it is part of, or even the physical location where the product is operating. In these broader contexts this additional data could be considered metadata if it is used to describe or qualify some other information.

All these forms of metadata have value in some context. Let's examine how, or if, we are delivering that value today.

Present State of Metadata in our Products

Offline Issues
Many implementations that do not provide rich context for their data online rely on an offline mechanism to describe their information models. Within the CIP ecosystem Electronic Data Sheets, or EDS files, are a primary, standardized method for offline description of devices. Some products deliver "stub" EDS files that only contain the bare minimum of required information. EDS files that do have content are used almost exclusively to document the configurable parameters of a device, but they don't need to be limited to that. Parameter entries can describe any interesting or useful piece of information that the product has to offer. Additionally, The CIP Networks Library, Volume 1, Chapter 7-3.6 describes ways to provide context to connection data using mechanisms such as Assem and Param entries [1].

Even if vendors provided the best possible EDS files, that would not always be enough. There are use cases where EDS files are missing and not easy to obtain. A machine that was programmed by a machine builder might be delivered without matching EDS files. In many plants access to the internet from the production network is blocked for security reasons making retrieval of the files inconvenient. We specify the ability to retrieve EDS files from the File object, but that is not required.

The Trouble with Optional

Many valuable CIP features are not implemented because they are optional. Even when they are implemented, if they are not identified in the product's EDS file, they go unnoticed because users or client tools do not know they exist. In other words, we have both a problem of low adoption and of poor discoverability leaving users without useful and interesting information. We could know which optional features are implemented; this information is included in STC files submitted for conformance test. However, STC files are not part of an end-user's workflow.

It Takes a Clever Client

Param entries in an EDS file may point to an instance of the Parameter Object within the device, but they might also point directly to data in any other object. The link path determines what the Parameter is associated with. Figure 1 shows a typical parameter entry from an EDS file.

Figure 1 - Typical Parameter Entry

```
Param111 =
        0,
        7, "20 0F 25 CB 00 30 01",
        0x02,
        2, 2,
        "Scaling",
        "",
        "",
        0, 1,
        0,
        1, 1, 1, 0,
        0, 0, 0, 0,
        0;
```

This Param entry points to a Parameter Instance (0xCB). That instance may or may not provide a full description of the data. This Param entry certainly could be more complete. There is no way to know from this entry what this data is or how it may relate to other Param entries in the EDS file. In some cases, a Parameter Group could be used to group associated data (e.g., a group for each I/O channel that contains all attributes related to the channel), but you still may not understand the data-metadata relationship between those Parameters. Additionally, not all parameter object instances need to have a corresponding entry in an EDS. Therefore, unless a device supports Full Parameter Objects, a tool always needs to have the EDS file and online access to the device before it can fully assess the device's information. Even with parameters defined in the EDS file, the tool needs to follow the Link Paths to the real objects that contain the data. Once this information is retrieved, the tool can start the task of figuring out the relationships between different pieces of information that essentially exist in a flat list. Even if those associations are made, for vendor-specific objects the tool has no way of knowing what is considered data, and what is metadata.

CIP would benefit from a simpler mechanism to associate bits of information beyond simply grouping them into the same object. We have examples of object associations in CIP. Electrical Energy Objects must be associated with an instance of the Base Energy Object. The Electrical Statistics Object must be associated with an instance of the Electrical Energy Object. The Process Device Profiles have fixed relationships between different Process Measurement Objects and Process Device Diagnostic instances. These are all examples of where we have specified associations between objects. That solution works for publicly defined objects, but even then, tools need to understand the object definitions. It is not effective for multivendor interoperability with vendor specific objects.

Fixed but Hidden
Most specification-defined metadata (e.g., name, data type, etc.) is not exposed in any way. A client or tool could have knowledge of publicly defined objects or even its own vendor-specific objects and provide the associated metadata for them. However, there is no way for clients to understand vendor-specific objects defined by other vendors. Parameter instances could be used in this case, but there are some limitations that would need to be addressed (e.g., parameter names being limited to 16 characters) and not all specification-defined metadata is currently represented by Parameter Object instance attributes (e.g., access rule).

Most Meaningful and Least Supported
Arguably the most valuable metadata to users is data they enter based on their applications. This category of data is probably neglected the most.

It is mainly only controller products that provide a linkage between device signal names and variable names in the user's program. Very few products provide any mechanism to store this information or any other information the user may want to enter. Many users employ design software to document electronic or mechanical designs. In most cases they then need to manually duplicate that effort in their control application code. And even after that, if they access devices directly, there is no record of the name used in the control program. An edge device could not easily construct an application-specific information model for such devices without having EDS files and also access to controllers connected to the device.

Metadata Use in Software

Software exists today that can browse devices and present users with all the interesting data that products possess. This data can be transferred to cloud applications where it can be used for tasks like asset management, preventative maintenance, or optimizing a process. These use cases only become possible when products supply meaningful metadata to help discover, identify, and provide context for the product's data.

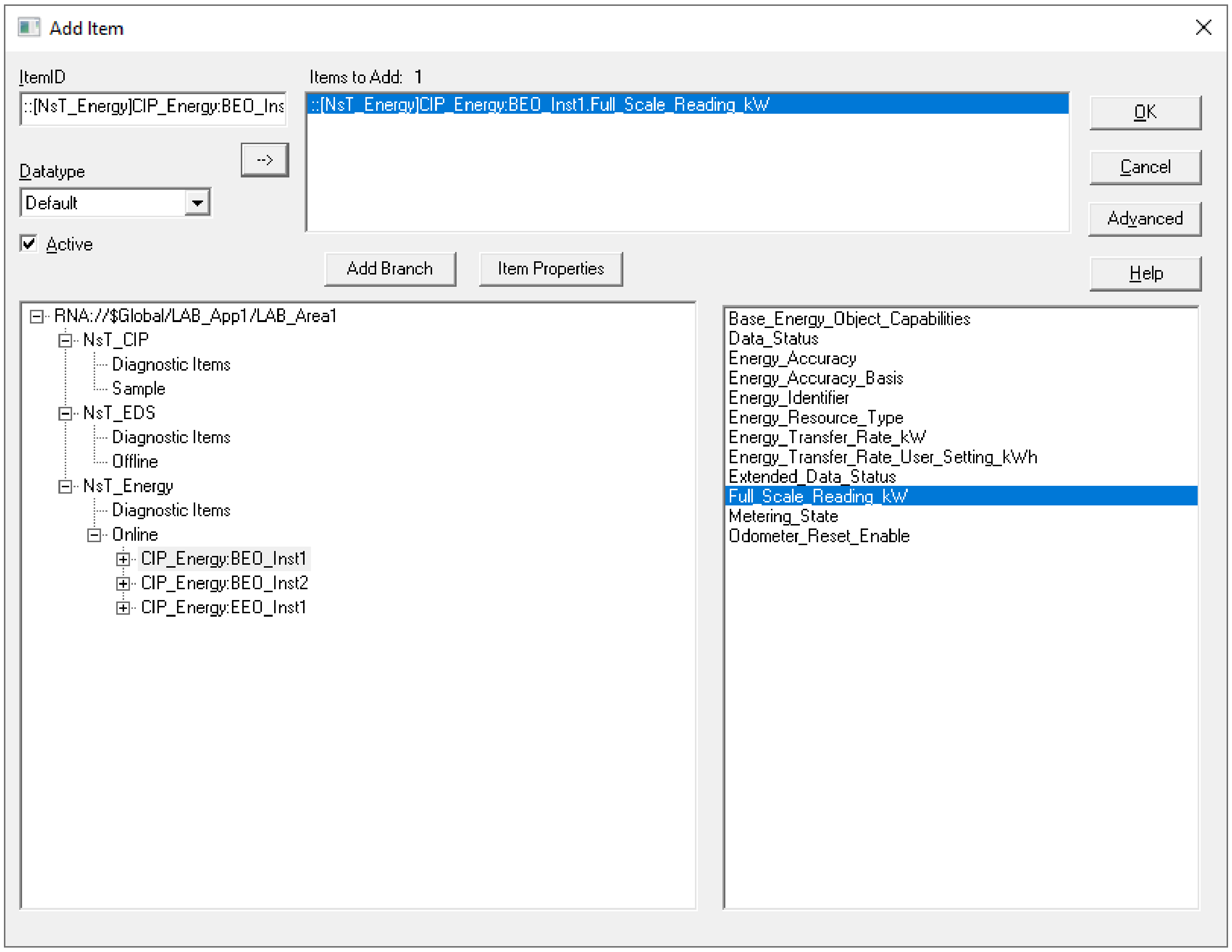
An example can be shown using Rockwell Automation's FactoryTalk Linx^{TM} and FactoryTalk LiveData^{TM}. These software applications can browse devices on an EtherNet/IP network and collect data that they discover. The amount that is known about the data can make a big difference in how it is presented.

Today these tools allow users to reference data using several different "shortcuts":

- CIP EPATH to the data of interest (e.g., Attribute)
- Parameter from the device's EDS file
- Element from a preformatted list (e.g., CIP Energy objects)

The left-hand pane of Figure 2 demonstrates this. Using NsT_CIP a user can enter any valid EPATH to a piece of information within the device. This list is empty until the user provides input. NsT_EDS will present the user with all the Parameters listed in the device's EDS file. The NsT_Energy shortcut shows all the CIP Energy Attributes of the Base Energy and Electrical Energy Objects. Users add items to scan from these lists. The figure shows the Full_Scale_Reading_kW item being added to the list of items to scan.

Figure 2 - Shortcuts to Sources of LiveData



ItemID: ::[NsT_Energy]CIP_Energy:BEO_Ins

Items to Add: 1

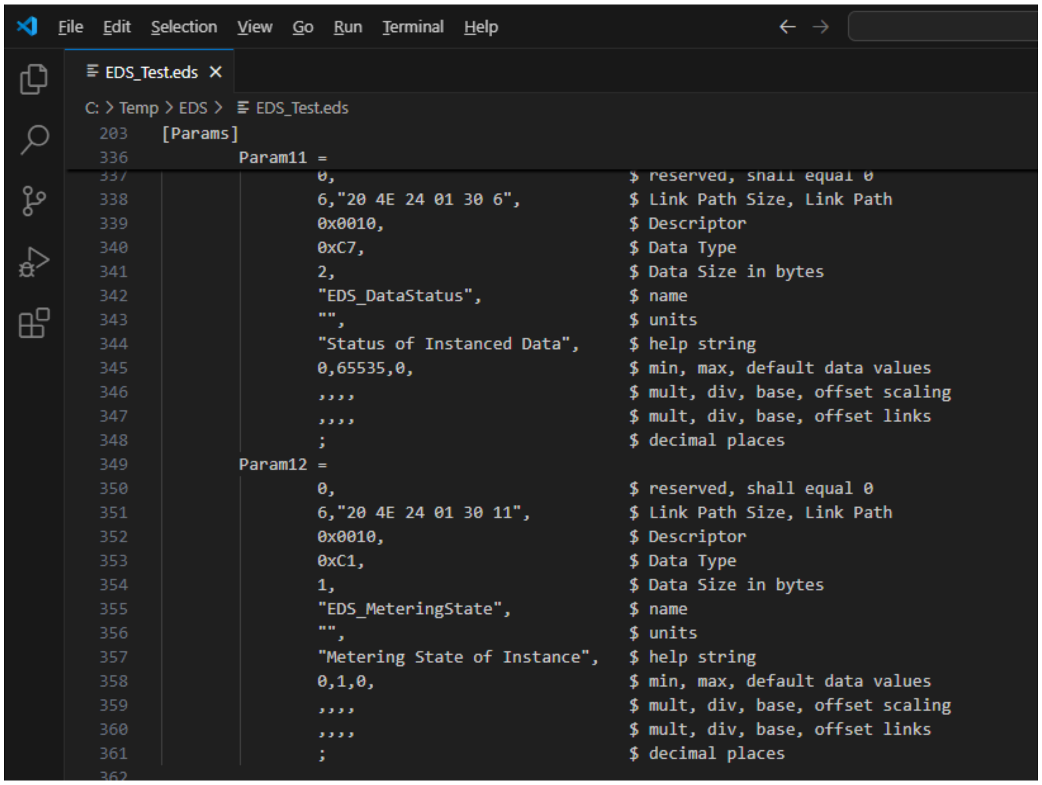
::[NsT_Energy]CIP_Energy:BEO_Inst1.Full_Scale_Reading_kW

```
RNA://$Global/LAB_App1/LAB_Area1
  NsT_CIP
    Diagnostic Items
    Sample
  NsT_EDS
    Diagnostic Items
    Offline
  NsT_Energy
    Diagnostic Items
    Online
      CIP_Energy:BEO_Inst1
      CIP_Energy:BEO_Inst2
      CIP_Energy:EEO_Inst1
```

```
Base_Energy_Object_Capabilities
Data_Status
Energy_Accuracy
Energy_Accuracy_Basis
Energy_Identifier
Energy_Resource_Type
Energy_Transfer_Rate_kW
Energy_Transfer_Rate_User_Setting_kWh
Extended_Data_Status
Full_Scale_Reading_kW
Metering_State
Odometer_Reset_Enable
```

The three shortcut types that are used in this example have access to different amounts of metadata and that leads to different presentations. To demonstrate this, the same two data items were scanned using all three shortcut types. Figure 3 shows the Param entries for the data items used in this example.

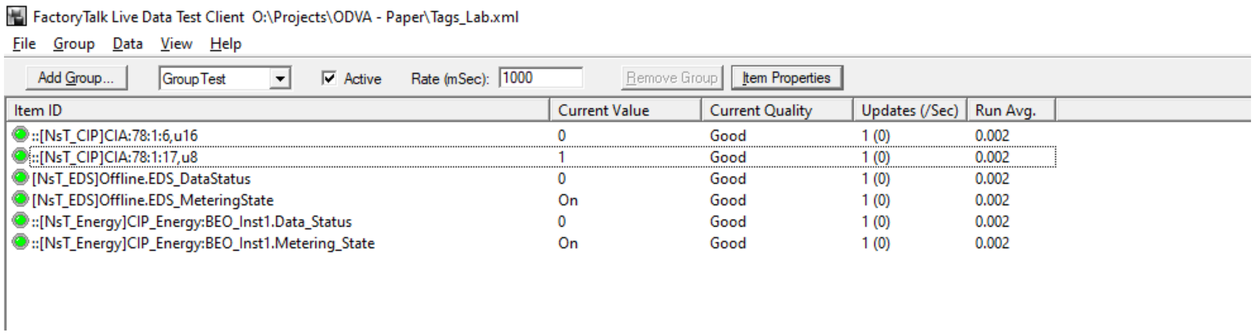
Figure 3 - Param Entries for the Data Items Used


File Edit Selection View Go Run Terminal Help
EDS_Test.eds
C: > Temp > EDS > EDS_Test.eds

```
[Params]
        Param11 =
                0,                              $ reserved, shall equal 0
                6,"20 4E 24 01 30 6",           $ Link Path Size, Link Path
                0x0010,                         $ Descriptor
                0xC7,                           $ Data Type
                2,                              $ Data Size in bytes
                "EDS_DataStatus",               $ name
                "",                             $ units
                "Status of Instanced Data",     $ help string
                0,65535,0,                      $ min, max, default data values
                ,,,,                            $ mult, div, base, offset scaling
                ,,,,                            $ mult, div, base, offset links
                ;                               $ decimal places
        Param12 =
                0,                              $ reserved, shall equal 0
                6,"20 4E 24 01 30 11",          $ Link Path Size, Link Path
                0x0010,                         $ Descriptor
                0xC1,                           $ Data Type
                1,                              $ Data Size in bytes
                "EDS_MeteringState",            $ name
                "",                             $ units
                "Metering State of Instance",   $ help string
                0,1,0,                          $ min, max, default data values
                ,,,,                            $ mult, div, base, offset scaling
                ,,,,                            $ mult, div, base, offset links
                ;                               $ decimal places
```

Figure 4 shows the Base Energy Object's Data Status and Metering State Attributes as represented by the preformatted model, Param entries in the EDS, and manually entered using a CIP EPATH.

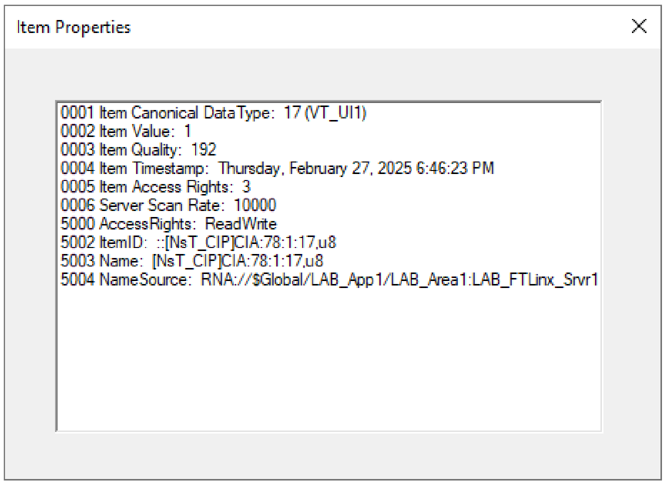
Figure 4 - Varying Levels of Metadata


FactoryTalk Live Data Test Client O:\Projects\ODVA - Paper\Tags_Lab.xml
File Group Data View Help
Add Group...
Group Test
Active
Rate (mSec): 1000
Remove Group
Item Properties

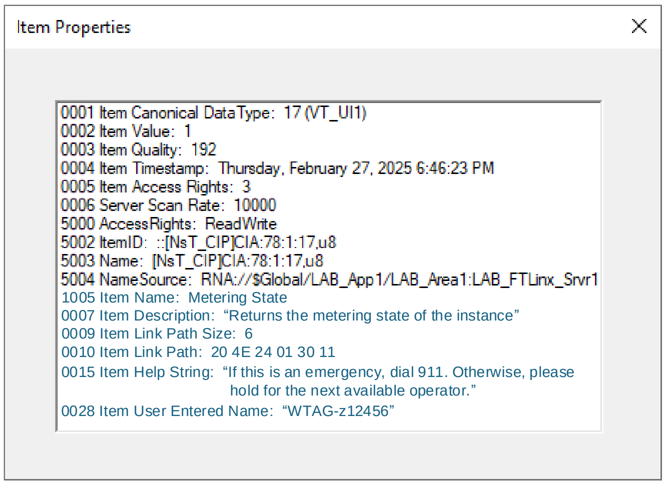
Item ID	Current Value	Current Quality	Updates (/Sec)	Run Avg.
::[NsT_CIP]CIA:78:1:6,u16	0	Good	1 (0)	0.002
::[NsT_CIP]CIA:78:1:17,u8	1	Good	1 (0)	0.002
[NsT_EDS]Offline.EDS_DataStatus	0	Good	1 (0)	0.002
[NsT_EDS]Offline.EDS_MeteringState	On	Good	1 (0)	0.002
::[NsT_Energy]CIP_Energy:BEO_Inst1.Data_Status	0	Good	1 (0)	0.002
::[NsT_Energy]CIP_Energy:BEO_Inst1.Metering_State	On	Good	1 (0)	0.002

The software can provide additional information for each data item by clicking on the Item Properties button as shown in Figure 5. The information on this properties page is not standardized in any CIP definition; it is only presented for the purposes of example. The left side of the figure displays what the tool showed for a specific data item. An extended list of standardized properties, if developed, could be presented in a dialog such as this, providing users with rich context for the data. The right side of the figure shows an enhanced set of properties.

Figure 5 - Data Item Properties Displays Metadata



```
0001 Item Canonical DataType:  17 (VT_UI1)
0002 Item Value:  1
0003 Item Quality:  192
0004 Item Timestamp:  Thursday, February 27, 2025 6:46:23 PM
0005 Item Access Rights:  3
0006 Server Scan Rate:  10000
5000 AccessRights:  ReadWrite
5002 ItemID:  ::[NsT_CIP]CIA:78:1:17,u8
5003 Name:  [NsT_CIP]CIA:78:1:17,u8
5004 NameSource:  RNA://$Global/LAB_App1/LAB_Area1:LAB_FTLinx_Srvr1
```



```
0001 Item Canonical DataType:  17 (VT_UI1)
0002 Item Value:  1
0003 Item Quality:  192
0004 Item Timestamp:  Thursday, February 27, 2025 6:46:23 PM
0005 Item Access Rights:  3
0006 Server Scan Rate:  10000
5000 AccessRights:  ReadWrite
5002 ItemID:  ::[NsT_CIP]CIA:78:1:17,u8
5003 Name:  [NsT_CIP]CIA:78:1:17,u8
5004 NameSource:  RNA://$Global/LAB_App1/LAB_Area1:LAB_FTLinx_Srvr1
1005 Item Name:  Metering State
0007 Item Description:  "Returns the metering state of the instance"
0009 Item Link Path Size:  6
0010 Item Link Path:  20 4E 24 01 30 11
0015 Item Help String:  "If this is an emergency, dial 911. Otherwise, please
                        hold for the next available operator."
0028 Item User Entered Name:  "WTAG-z12456"
```

Obviously, the method that utilizes the CIP EPATH is the least desirable. With this method, you only get the value of the item that is referenced.

The presentation based on Param entries is already better. A name is provided, and the data value was enumerated (i.e., instead of seeing 1, the user sees "On"). This solution is data-driven, but relies on accurate, complete, and consistent Param entries coded by different developers from different companies. In other words, the same CIP Attribute referenced in two different EDS files could be presented differently to an end user causing frustration and confusion. You can see slight differences in naming for identical items shown in Figure 4 (i.e., EDS_DataStatus vs. Data_Status).

Of course, the preformatted model can provide the richest presentation. The software can be instrumented (read hard-coded) with all the details presented in the CIP specifications. The obvious downside to this is the amount of work that needs to be done to support each object and the endless maintenance that will result as objects are updated.

What Needs to Change

For some of the issues presented we currently do not have a specified solution. For others, we have solutions, but they are not widely supported. In all cases there is room for improvement. EDS files can be better utilized to expose more information about products. Additionally, we need to specify mechanisms that allow products to report metadata online.

Expose ALL Metadata in EDS Files

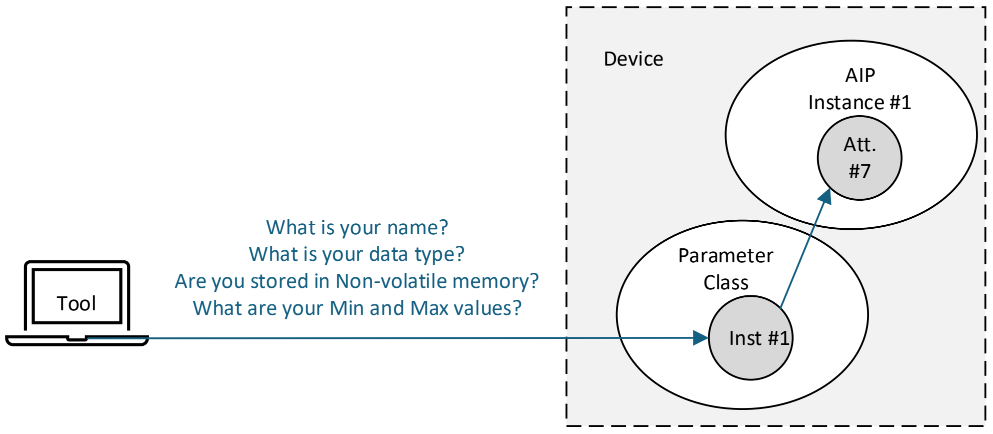
Too much of the EDS file content is optional. ODVA must move towards making more fields mandatory. EDS file creation, testing, and maintenance is much less expensive than product firmware. We should start there to get the most benefit from a relatively low amount of effort. Products can and should be required to describe more of the contents of their implementation to support end-users' needs and multi-vendor interoperability.

Provide a Central Repository

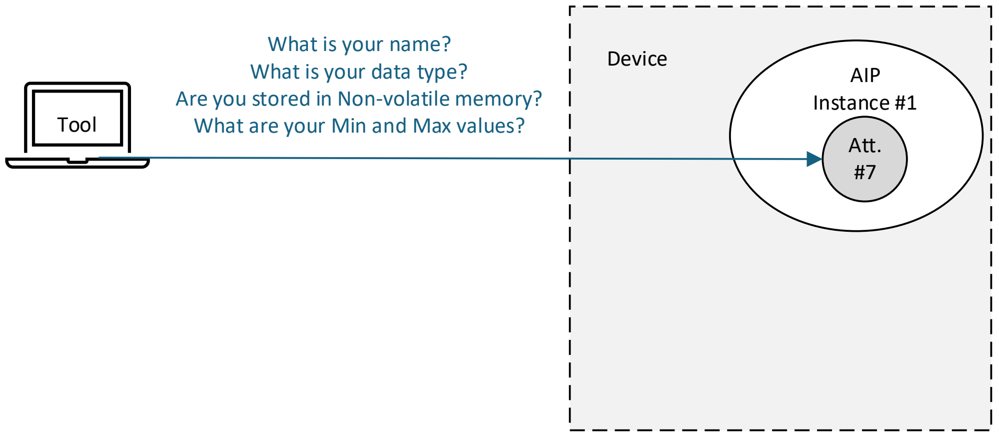
We need to make it more convenient for users to locate EDS files. Of course, the best solution for access to EDS files is for products to ship with their EDS embedded, but as we add more content to the files that may present a challenge for constrained devices. As a backup, and for cases where a different version of the file is needed, a central repository maintained by ODVA would provide the best experience for our end users.

Expose More Metadata in Firmware

Using an existing mechanism, products could support Parameter instances for every attribute in a product, thereby providing a comprehensive set of metadata. The downside to this technique is that it leads to bloat in the implementation due to the indirect nature of the Parameter object. You are effectively going to one object to learn the details of other objects, and in the process, you lose the object-oriented nature of CIP and end up with that flat list of parameters that is difficult for client tools to navigate.



An alternative to this would be for objects to possess and report their own metadata.


What is your name?
What is your data type?
Are you stored in Non-volatile memory?
What are your Min and Max values?
Tool
Device
AIP
Instance #1
Att.
#7

At ODVA’s 2023 Industry Conference I presented a simple, efficient extension that would enable CIP devices to report metadata for any addressable item in a device’s address space (e.g., class instances, attributes, or services). [2]

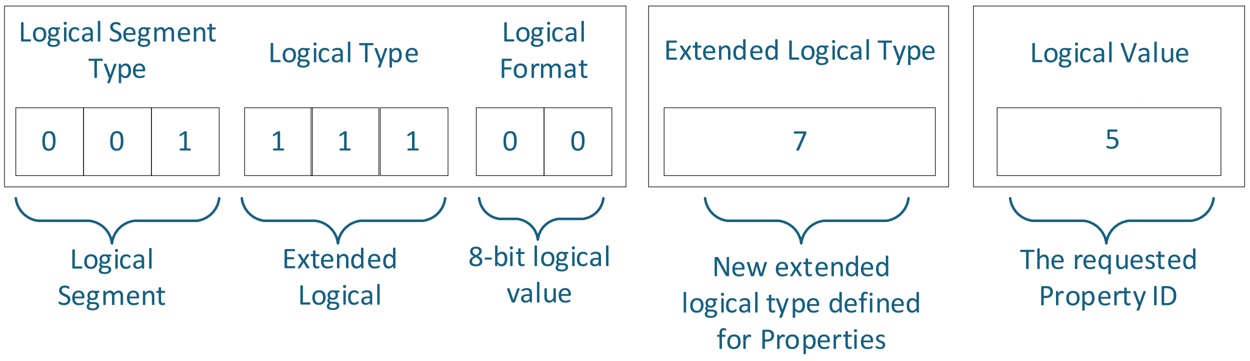
The CIP object model presents class and instance attribute data using tables as shown in Table 1 below. Each column specifies some property of the attribute. Columns 2-7 could be standardized metadata properties for all attributes. In other words, the name of any attribute could be addressed by referring to property 5 of that attribute or the data type as property 6. Some similar information could be addressed for class instances or services.

Table 1 - Attribute Metadata Properties

Attribute ID	Need in Implementation	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Values
1	2	3	4	5	6	7	8

A new Extended Logical segment could be defined to address these new metadata properties. Figure 6 shows how the new segment could be defined. See the CIP Networks Library Volume 1, Appendix C, Section C-1.4.2 for a complete definition of Logical segment types.

Figure 6 - New Extended Logical Segment to Address Properties



An example using this new segment type is shown in Table 2 below addressing the Name Property of the Identity Object's Product Code attribute.

Table 2 - Example Using Logical Segments for Attribute Metadata Properties

Segment Contents	Notes
[20][01][24][01][30][03][3D 07][05 00]	Segment Type = Logical Segment. 20 01 indicates Class 1 (Identity Object) 24 01 indicates Instance 1 30 03 indicates Attribute 3 (Product Code) 3D 07 05 00 indicates metadata property 5 (Name)

If a Get_Attribute_Single request was sent with this path, the response should be a string equal to "Product Code". Note that if a product supported Symbolic addressing, once obtained by a client tool, users could potentially use names instead of PATHs to reference data items in a product.

The list of properties could be extended to include information included in the Parameter object as well as new information that is currently not defined in CIP objects. Work would still need to be done to agree upon the final list of standardized properties. Table 3 presents a list of properties that could be standardized for Attributes.

Table 3 - Proposed Standard Properties

Property ID	Property Name	Need in Implementation for Attributes	Comment
1	Attribute ID	Required	Useful if the attribute was accessed symbolically or by a mechanism other than a CIP message (e.g., over a different communication channel)
2	Need in Implementation	Optional	
3	Access Rule	Required	
4	Non-Volatile	Required	
5	Name	Required	
6	Data Type	Required	
7	Description	Optional	
8	Semantics	Optional	Fantasy? Perhaps.
9	Link Path Size	Required	Useful if the attribute was accessed symbolically or by a mechanism other than a CIP message (e.g., over a different communication channel)
10	Link Path	Required	Useful if the attribute was accessed symbolically or by a mechanism other than a CIP message (e.g., over a different communication channel)
11	Descriptor	Optional	Much like its use in the Parameter Object, the Descriptor can be used to define how other Properties are used
12	Data Size	Optional	Useful for STRUCTs or ARRAYs
13	Units	Optional	
14	Units String	Optional	
15	Help String	Optional	
16	Minimum Value	Optional	
17	Maximum Value	Optional	
18	Default Value	Optional	
19	Scaling Multiplier	Optional	
20	Scaling Divisor	Optional	
21	Scaling Base	Optional	
22	Scaling Offset	Optional	
23	Decimal Precision	Optional	
24	Time Domain	Optional	How often this attribute is expected to change value
25	Raw Value	Optional	
26	Enumerated Value	Optional	
27	Synonym	Optional	Could be an array of strings with synonyms for the Name property
28	User Entered Name	Optional	The user-entered name for this attribute. Provides linkage between the user's program and the device.
29	Data Tags	Optional	An array of "#tags" that would enable sorting and filtering data items
30	Pending Value	Optional	Report the pending value of an attribute prior to execution of an Apply Attributes service
31-255	Reserved		
256 – 511	Object Specific Range		
512-768	Vendor Specific Range		
769-65,535	Reserved		

A range of implementations are possible. The most capable products could provide all metadata online. Constrained devices could choose to only provide support for properties offline via their EDS files. An in-between solution could provide online metadata for vendor specific information and rely on a published database of metadata for publicly defined CIP objects to limit the amount of memory consumed. This database could be maintained by ODVA and provided to tool vendors.

Encourage Broad Support

There are many optional attributes defined in CIP objects that few products implement. The CIP ecosystem has a chicken-egg problem with support; device vendors point to tool vendors saying that they do not support feature X because tools do not provide support, and tool vendors point to devices with similar comments. In some cases, we should probably change these items to be mandatory, but in many other cases making these features mandatory is not the right solution. A possible resolution to this is for some pioneering implementations to show the value associated with support for these optional features. Once users are exposed to them, they may start requesting support and thereby create demand.

Future View

There has been a lot of talk in industrial automation about information models. Think of all the commercial hype about the Process Automation Device Information Model (PA-DIM). But what benefit does this model actually bring to users? Industry groups spend long hours defining a standard way to represent a product or a process. Many times, that standard representation results in a “lowest common denominator” implementation that everyone can agree to. There is some small amount of benefit to standardized representations, but that benefit comes with the constraint of conformity to that least common denominator. Wouldn’t it be much more useful for device vendors to be able to describe their devices completely and consistently without being constrained by someone else’s thoughts about what it should look like? Now imagine a headless edge gateway that doesn’t have access to EDS files. How could it create an information model of interesting information from our products? It could if metadata was provided.

Summary/What’s it Good For?

Context-Aware Operation:
Devices can use metadata to adapt their behavior based on their environment or user preferences.

Seamless Integration:
Enriched and complete metadata enable devices to create an efficient ecosystem.

Simplified Discovery:
Metadata allows users to quickly understand a device's features and status.

Searchability:
Metadata improves how devices are represented in larger systems, such as IoT ecosystems or enterprise networks. This makes them easier to locate and manage.

Diagnostics Information:
Devices often generate metadata related to their health, performance, and error logs. This metadata is invaluable for diagnosing issues and optimizing performance.

Lifecycle Tracking:
Metadata can include usage history, firmware versions, and update records, aiding in proper maintenance and timely upgrades.

In summary:
Metadata enriches a device's representation by providing contextual, descriptive, and functional data that improves its usability, integration, and value to users and systems. The time has come to augment our products with this metadata to enable new and interesting workflows.

[1] ODVA, The CIP Networks Library, Volume 1: Common Industrial Protocol, Ann Arbor: ODVA, Inc., 2001-2024.

[2] Author, Gregory Majcher (2023). Enabling Data Scientist Use Cases with Discoverability and Metadata. https://www.odva.org/library_proceedings/enabling-data-scientist-use-cases-with-discoverability-and-metadata/

[3] The American Heritage® Dictionary of the English Language, 5th Edition