# Mapping CIP to OPC UA

Gregory Majcher
Principal Engineer
Rockwell Automation, Inc.

Presented at the ODVA
2022 Industry Conference and 21st Annual Meeting
March 9, 2022
San Diego, California

## Abstract

The world is abuzz with talk of the next industrial revolution, Industry 4.0. Digitization, smart technologies, machine-to-machine communications, Internet of Things deployments, the list goes on, and on. All these innovative concepts will benefit from the standardized exchange of information between all levels of an industrial system. New products will evolve over time, but existing technologies can be adapted now to leverage the massive installed base of CIP™-enabled products around the world. One such technology that is gaining acceptance is OPC UA. Many vendors, end users, and industry consortia have converged on OPC UA as a vendor-neutral mechanism to exchange data. ODVA and OPC Foundation are jointly developing a CIP Companion Specification to support this data exchange.

This paper documents the approach taken by that joint working group to map the CIP information model to the OPC UA information model. With this mapping, applications utilizing OPC UA and its associated companion specifications can obtain data from CIP devices without the need to understand CIP. This solution will easily bring value to end users seeking to extract more data from existing installations for their evolving application needs.

## Keywords

OPC UA, Companion Specification, Gateway, Information Model, PA-DIM

## Definition of terms

There are some significant terms that collide between CIP and OPC UA.  In this section the overloaded terms provide both the CIP and the OPC UA definitions so that the differences are made clear.

**AddressSpace**
The information that an OPC UA server exposes to clients.  This information would be analogous to the collection of Objects that a CIP device exposes.  The AddressSpace of many OPC UA servers also contains complete type information of all types used in instances.

**Attribute (OPC UA)**
Characteristics that are used to describe OPC UA Nodes.  The set of Attributes is fixed and defined in a core OPC UA specification.  Each NodeClass has a defined subset of the Attributes that it is required to support.  Some Attributes are required by all NodeClasses (e.g., NodeId).

**Attribute (CIP)**
Characteristics of an Object that provide status or govern the operation of an Object.  Each CIP Class defines its own Attributes.

**BrowseName**
The symbolic name for a Node. It is comprised of an index that points to a namespace and a symbolic name (text). A BrowseName may not be unique and cannot be used to uniquely identify a Node – only the NodeId is unique.

**BrowsePath**
A sequence of BrowseNames constructed by starting from a known point in the AddressSpace and following references to subsequent Nodes. It can be thought of much like a path in a file system that leads to a file. As with a BrowseName, a BrowsePath may not be unique.

**Class/Object/Instance (CIP)**
A Class is a set of Objects that all represent the same kind of system component. An Object Instance is the actual representation of a particular Object within a class. Each Instance of a Class has the same set of Attribute and service definitions.

**ObjectType/Object Instance (OPC UA)**
An ObjectType provides the type definition for a physical or abstract element of a system. An Object Instance is the actual representation of a particular Object. For example, a Pump may be an ObjectType, and a server can have many instances of it in its address space.

**DataType (OPC UA)**
A DataType is used to describe the format of a value transmitted on-the-wire. It may be a simple integer or a complex, structured value. Only Variables and VariableTypes have the DataType attribute.

**Interface**
Functionality that is grouped so that it can be easily added into an information model. Interfaces allow you to build your information model using “has a” relationships. For example, the information model for a car could reference via Interface the model for a radio. This allows the radio to be defined separately and included where needed. The car must expose mandatory elements of the radio model and may optionally support optional elements.

**Node**
The fundamental building block of OPC UA information modeling. Everything in an OPC UA information model is a Node. Objects, Variables, Methods, etc., are all referred to as Nodes.

**NodeClass**
A classification for the different kinds of Nodes that defines the set of metadata (e.g., Attributes) that each must maintain. For example, there is a NodeClass to represent Objects and a different NodeClass to represent Variables.

**NodeID**
A unique identifier for a piece of information within an OPC UA server. It is comprised of a namespace, which identifies a naming authority, and an identifier assigned by the naming authority, which is unique within the namespace.

**Reference**
Defines the relationship between two Nodes. For example, an Object will use a “HasComponent” Reference to describe the relationship it has to one of its Objects or Variables. References can point in a forward or reverse direction. The reverse of “HasComponent” is “ComponentOf”. This allows the server to model sophisticated relationships between Nodes, such as a doubly-linked list.

**Variable/DataVariable/Property (OPC UA)**
Variables are used to represent values much like Attributes in CIP. However, OPC UA makes a distinction between DataVariables and Properties. A DataVariable can have a Property that further describes it. For example, CIP models an analog channel as an Object with Attributes representing the

value and the engineering units.  OPC UA might also model the channel as an Object, but the value would be a DataVariable and the engineering units would be a Property of the value.

## Introduction

The original OPC standard was born out of a need to “abstract PLC-specific protocols into a standard interface”.  That need still exists today.  End users choose field level protocols based on application specific needs, but there is an increasing desire to collect and use data from devices on all those disparate fieldbuses.  The users of that data may not understand the details of each field level protocol and they wish to get data consistently across all of them.  Many are choosing OPC UA to achieve that. ODVA’s Common Industrial Cloud Infrastructure Special Interest Group (CICI SIG) was formed to enable interactions between cloud applications and ODVA CONFORMANT™ devices that support EtherNet/IP™.  At ODVA’s 2020 Industry Conference, members of the CICI SIG presented a paper documenting use cases for a CIP Companion Specification for OPC UA [1].  This paper will outline the progress towards meeting some of those use cases.

## What is OPC UA?

OPC UA is an open, royalty free set of standards designed as a universal communication protocol.  It makes use of standard internet technologies, like TCP/IP, HTTP, and Web Sockets.  It defines sets of core services and a basic information model framework. That framework allows vendor-specific information to be exposed in a standard way using metadata.  This enables OPC UA Clients to discover and use that vendor-specific information without prior knowledge of it.

### Information Modeling in OPC UA

The elemental modeling concept in OPC UA is the “Node”.  Everything is implemented using Nodes: objects, methods, variables, references, etc.  All Nodes are described by a set of required attributes such as the NodeId, NodeClass, and BrowseName.  Different node classes have additional attributes – for example, Variable nodes have a Value, DataType, and other attributes which describe the contents of the data.  These attributes are the metadata that allows clients to discover the information that a server has to offer.  OPC UA uses inheritance, and each new, derived type can add required and optional attributes, or require an optional attribute from a base class be mandatory in the derived class, but cannot remove support for inherited attributes.  This technique is used by all information modeling elements used in OPC UA.

Much like CIP, OPC UA uses Objects to represent complex information.  The information modeling framework defines a textual representation and a graphical representation that is similar to UML.  Figure 1 shows a simple information model for how a drill press might be modeled in OPC UA.

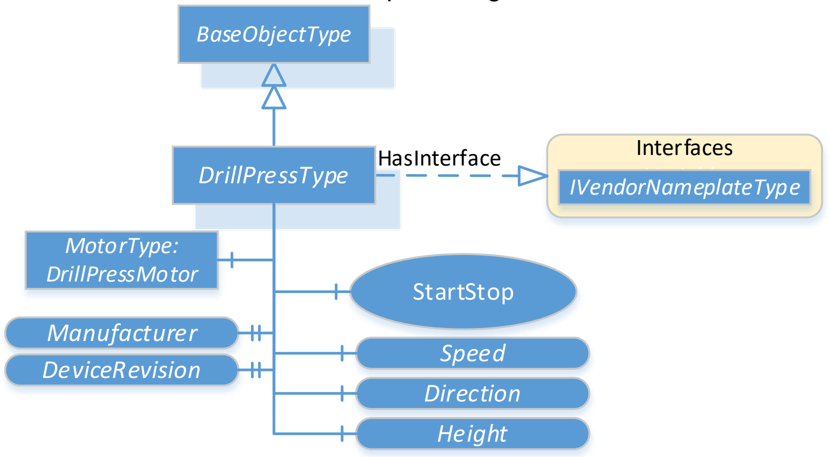

*Figure 1 - OPC UA Model for a Drill Press*

You can see from the figure that the DrillPressType inherits from the BaseObjectType. The BaseObjectType is defined by OPC UA 10000 – 5 Information Model. OPC UA's Part 5 is a core specification that defines the required attributes that all Object Types must support for consistency.

The newly defined DrillPressType object has components (an object representing the motor, a method to control operation, Speed, Direction, and Height) and properties (Manufacturer and DeviceRevision). The object also includes an OPC UA "Interface". Two of the optional properties from the interface (Manufacturer and DeviceRevision) are defined as mandatory for this object so they are specifically shown in the drawing.

The textual representation for this model is shown in Table 1. Each element of the object definition is further defined in this table. References indicate the relationship between the object and each element. The Node Class tells you which fundamental modeling construct (Node Type) is used. DataTypes can either be fundamental data types defined in the core specifications, or derived types. To help readers identify where something is defined, namespaces are included in the definition. If a number and a colon precede an element, it means that it is defined elsewhere. Specifications are required to supply a namespace table containing an index and a URI of any referenced type definitions. The "Other" column documents the need in implementation. Mandatory (M) and Optional (O) are shown here, but OPC UA defines additional classifications not demonstrated in this simple example.

| **Attribute** | **Value** | | | | |
|---|---|---|---|---|---|
| BrowseName | DrillPressType | | | | |
| IsAbstract | False | | | | |
| **References** | **Node Class** | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| Subtype of the BaseObjectType defined in OPC 10000-5 | | | | | |
| 0:HasComponent | Object | DrillPressMotor | | 2:MotorType | M |
| 0:HasComponent | Variable | Speed | 0:Int32 | 0:PropertyType | O |
| 0:HasComponent | Variable | Direction | 0:Boolean | 0:PropertyType | M |
| 0:HasComponent | Variable | Height | 0:Int32 | 0:PropertyType | M |
| 0:HasComponent | Method | StartStop | Defined in x.y.z | | M |
| | | | | | |
| 0:HasInterface | ObjectType | 1:IVendorNameplateType | | | |
| **Applied from IVendorNameplateType (defined in OPC 10000-100)** | | | | | |
| 0:HasProperty | Variable | 1:Manufacturer | 0:LocalizedText | 0:PropertyType | M |
| 0:HasProperty | Variable | 1:DeviceRevision | 0:String | 0:PropertyType | M |

*Table 1 - Drill Press Object Type Definition*

Finally, you will recognize that all the elements have a BrowseName. The BrowseName is a required attribute of all elements in OPC UA. The BrowseName is human-readable, but its primary use is for clients to build paths of BrowseNames. Clients gain access to a server's address space through some known entry point and then start navigating using BrowseNames and references between the elements. Once the client finds the specific information it needs, it invokes a service on the server to translate a BrowsePath into a NodeId. Once the client has the NodeId, it will use that to access the information in the future. This whole model is very similar to the directory structure on a computer.

## What is a Companion Specification

A companion specification uses OPC UA's information modeling framework to describe common information found in an industry vertical (e.g., Plastics and Rubber), or standard types of machines (e.g., Pumps and Vacuum Pumps). These specifications attempt to harmonize the representation of the information in that vertical or machine by defining objects, variables, data types, methods and references specific to their operation.  Simply put, companion specifications define a common interface to information represented by the model that could be supported in any instantiation of that model.

**The CIP Companion Specification**
CIP does not represent any one industry vertical or specific machine.  It is used everywhere throughout industry.  The purpose of the CIP Companion Specification is to provide a translation that would allow information found in CIP devices to be represented in the form needed by existing and future OPC UA client applications regardless of what companion specifications those applications use.  An example of the translation that is needed is presented in this paper.

The translation rules defined by the CIP companion specification can be implemented in any device that has an OPC UA server, however the most efficient and expedient way to enable this capability in existing installations would be for it to exist in a gateway appliance, or gateway function in a product already on the network.

Figure 2 depicts an industrial network that is accessed by an OPC UA Client Application.  OPC UA clients can discover OPC UA Servers and browse their address spaces.  In this figure either the controller or the gateway appliance could contain the gateway functionality that would gather data from devices and translate it for the OPC UA client.  This could be done for EtherNet/IP devices as well as non-CIP field devices integrated through CIP's Integration Volumes 7A-C.  The client could see each proxied device as an individual OPC UA server as if it were communicating directly with it.

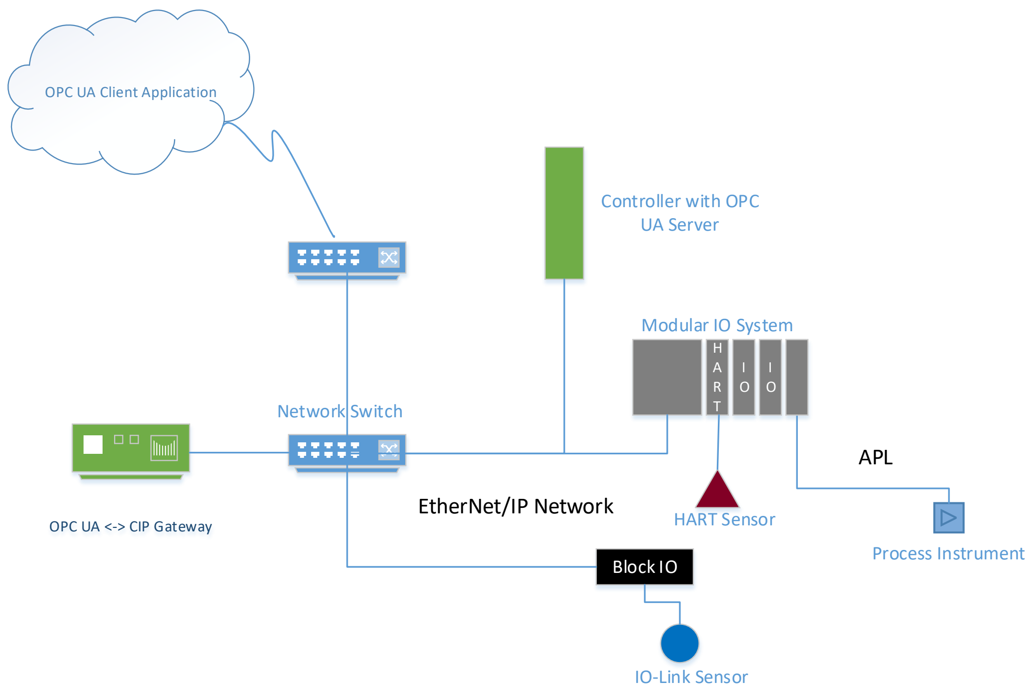

*Figure 2 - Industrial Network Accessed by OPC UA Client Application*

**CIP Device Information Model**
To build our model we considered the elements that are common to all CIP devices: an identity, a network interface, and a collection of device-specific application objects.  We created a standard representation for the Identity Object and CIP network interfaces by defining several OPC UA object and

interface types.  For all other device-specific application objects we plan to create OPC UA types that could model any CIP object generically.

Many of the use cases outlined in [1] deal with asset management.  For OPC UA this kind of information is defined in OPC 10000-100 (Devices).  While there is considerable overlap between the information in OPC's Devices specification and CIP's Identity Object, we decided to model both.  We wanted to preserve all the attributes of the Identity Object in a lossless way and be able to represent a CIP device as modeled in OPC's Part 100: Devices.

## CIP Identity Model

To model the CIP Identity information, we defined a CipIdentityType Object.  The object inherits from OPC UA's BaseObjectType; the root for all object definitions.  Each required CIP attribute is modeled as a property of the CipIdentityType.  In addition to this ObjectType, we also defined an identical InterfaceType so that existing models could expose the CIP Identity Object attributes simply by including the interface.  Interfaces are an OPC UA modeling construct that enables you to build object functionality by referencing the interface (i.e., composition).  When referencing an interface, the source object must include mandatory elements in the interface and can optionally include any optional elements.

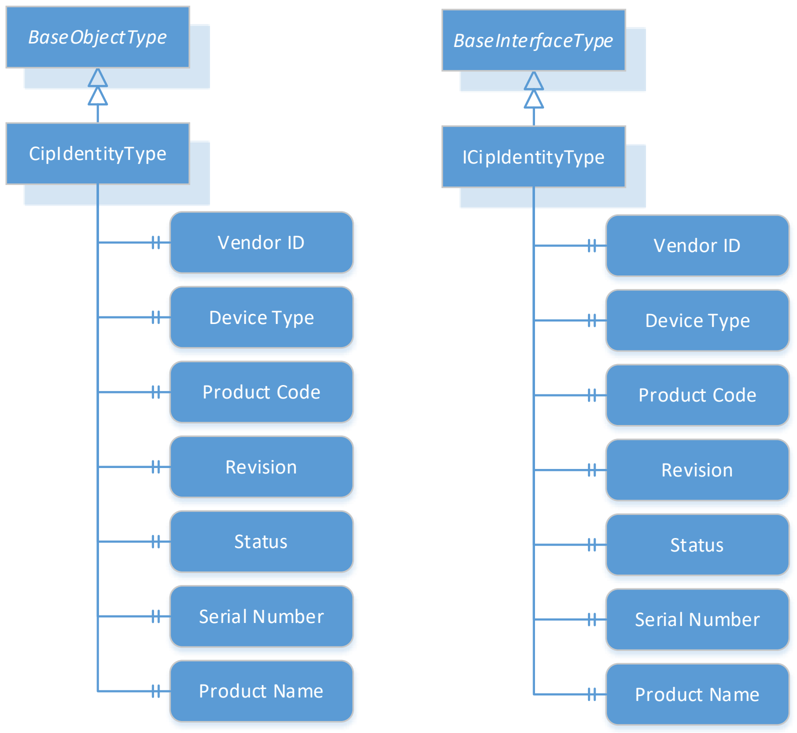

*Figure 3 – Modeling the Identity Object*

The optional attributes of the Identity Object were also added to OPC UA Interfaces.  We grouped optional attributes into five interfaces: common, uncommon, HART, IO Link, and ModBus.  This was a tradeoff between defining many interfaces and one very large interface that contained everything the Identity Object defines.  This grouping should allow for more sensible implementations.  For example, CIP can model the Identity of HART, IOLink, or ModBus devices.  The optional interfaces needed for these attributes can be applied only where needed to represent devices on those networks.

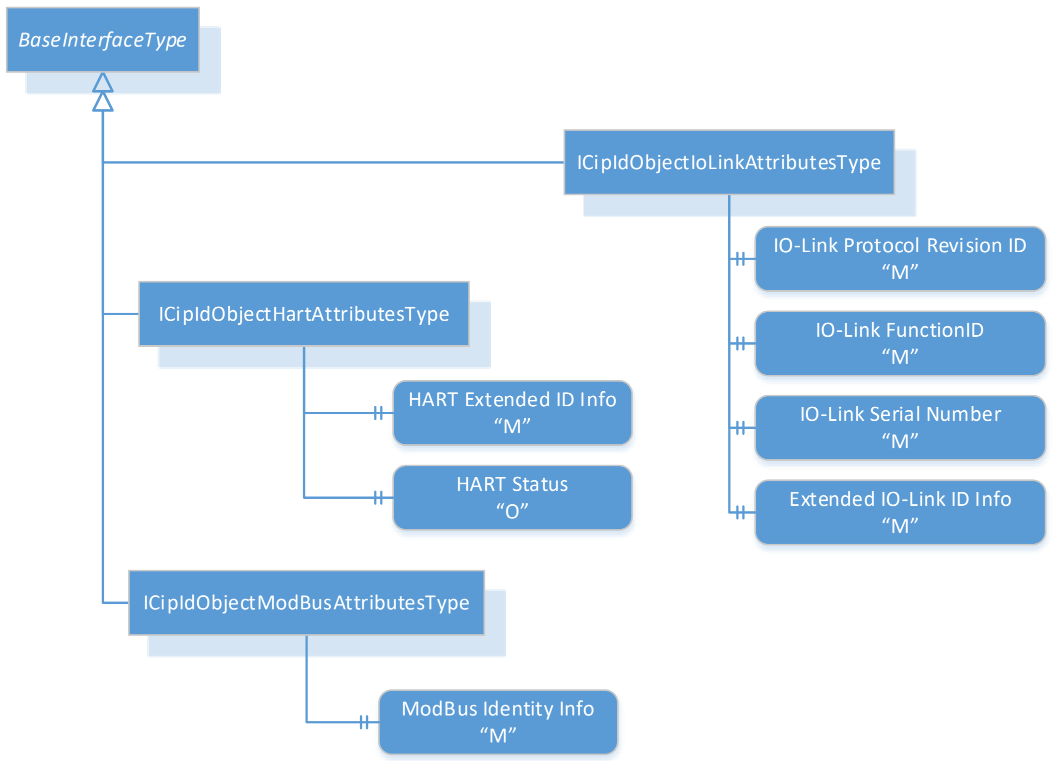
BaseInterfaceType
ICipIdObjectIoLinkAttributesType
IO-Link Protocol Revision ID “M”
IO-Link FunctionID “M”
IO-Link Serial Number “M”
Extended IO-Link ID Info “M”
ICipIdObjectHartAttributesType
HART Extended ID Info “M”
HART Status “O”
ICipIdObjectModBusAttributesType
ModBus Identity Info “M”

*Figure 4 - Optional Functionality Modeled using Interfaces*

OPC UA’s Part 100: Devices specification defines information for the identification of devices. Those properties are contained in two interfaces, IVendorNamplateType and ITagNameplateType. Figure 5 shows the CIPDeviceType inheriting from the ComponentType which is defined in Part 100. The CIPDeviceType further mandates several optional properties from the Part 100 interfaces to align with many existing companion specifications (e.g., PA-DIM). This collection of OPC UA properties and CIP Identity Object Attributes represents all the identification functionality from both specifications.

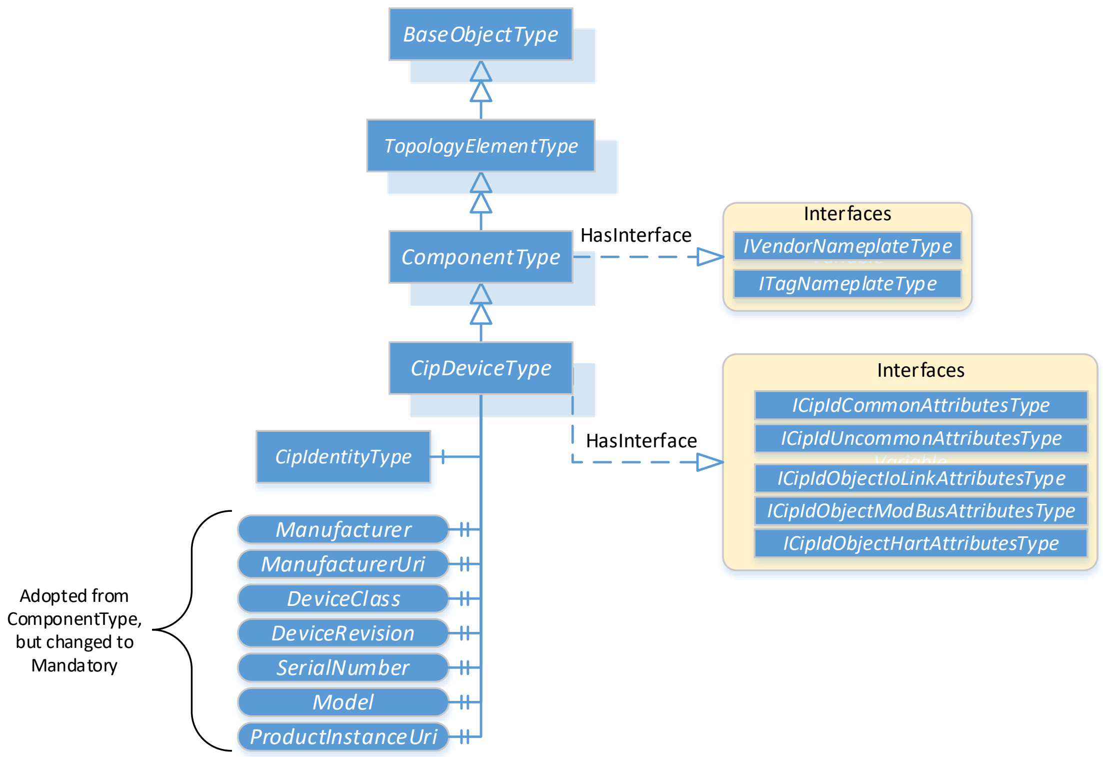

*Figure 5 - CipDeviceType Derived from Part 100 Types*

Some CIP attributes have a close equivalent property in an OPC UA object, while others require significant translation. For example, OPC UA makes extensive use of string data types in many properties that have numeric equivalents in CIP attributes. Table 2 demonstrates examples of the translation rules being proposed for the CIP companion specification for the identity information inherited from Part 100.

*Table 2 - Translation for Identity Information*

| OPC UA ComponentType | | CIP Identity Object | | Mapping |
|---|---|---|---|---|
| Property | DataType | Attribute | Datatype | |
| Manufacturer | Localized Text[1] | Vendor ID | UINT | ODVA maintains a list of manufacturer names alongside the Vendor ID. The numeric Vendor ID must be mapped to the manufacturer name string by the gateway function. |
| ManufacturerUri | String | Vendor URI | STRING | If the Vendor URI is not supported, a client or gateway shall report a value of “www.odva.org/vendor_id_NNNNN”, where NNNNN is the value of the Vendor ID attribute in decimal with leading zeroes. |
| Model | Localized Text[1] | International Product Name | STRINGI or SHORT_STRING if Product | If the device does not support International Product Name, the Product Name attribute should be used, but it will not be localized. |

| | | | | |
|---|---|---|---|---|
| | | | Name is returned | |
| ProductCode | String | Catalog Number | SHORT_STRING | If Catalog Number is not supported, return empty string. |
| HardwareRevision | String | Hardware Revision | Struct of: USINTs | Convert the USINTs to “Major.Minor“ |
| SoftwareRevision | String | Revision | Struct of: USINTs | Convert the USINTs to “Major.Minor“ |
| DeviceRevision | String | Revision | Struct of: USINTs | Convert the USINTs to “Major.Minor“ |
| DeviceClass | String | Device Type | UINT | If the Device Type maps to a publicly defined device profile, the Gateway will respond with a string for that type. Otherwise respond with “Vendor Specific Device Type XX“, where XX is the Device Type. |
| SerialNumber | String | Serial Number | UDINT | Convert the UDINT to a String |
| ProductInstanceUri | String | | | Gateway will retrieve the ManufacturerUri and Serial Number from the device. If ManufacturerUri is not supported, it will retrieve the Vendor ID and map the Vendor ID to the ManufacturerURI. The ManufacturerUri will be concatenated with the Serial Number. |
| AssetId | String | Assigned_Name | STRINGI | If the device does not support the Assigned Name attribute, return a Null string |
| ComponentName | Localized Text[1] | Assigned_Description | STRINGI | If the device does not support the Assigned Description attribute, return a Null string |
| 1: If a device does not support localization, OPC UA specifications indicate that a default language should be used. It would be acceptable for devices or the gateway to always respond with English. | | | | |

## CIP Network Interface Model

OPC UA 10000 - 22 Base Network Model defines a very basic InterfaceType to be used by all networks. We defined object types that referenced that InterfaceType and new InterfaceTypes that model the information for the specific CIP network of the device. Part 22 defines a well-known location in the server’s address space for network interfaces. Our new types will be located there so that any client can learn about a device’s interface to its network. Figure 6 shows the model for DeviceNet and EtherNet/IP.

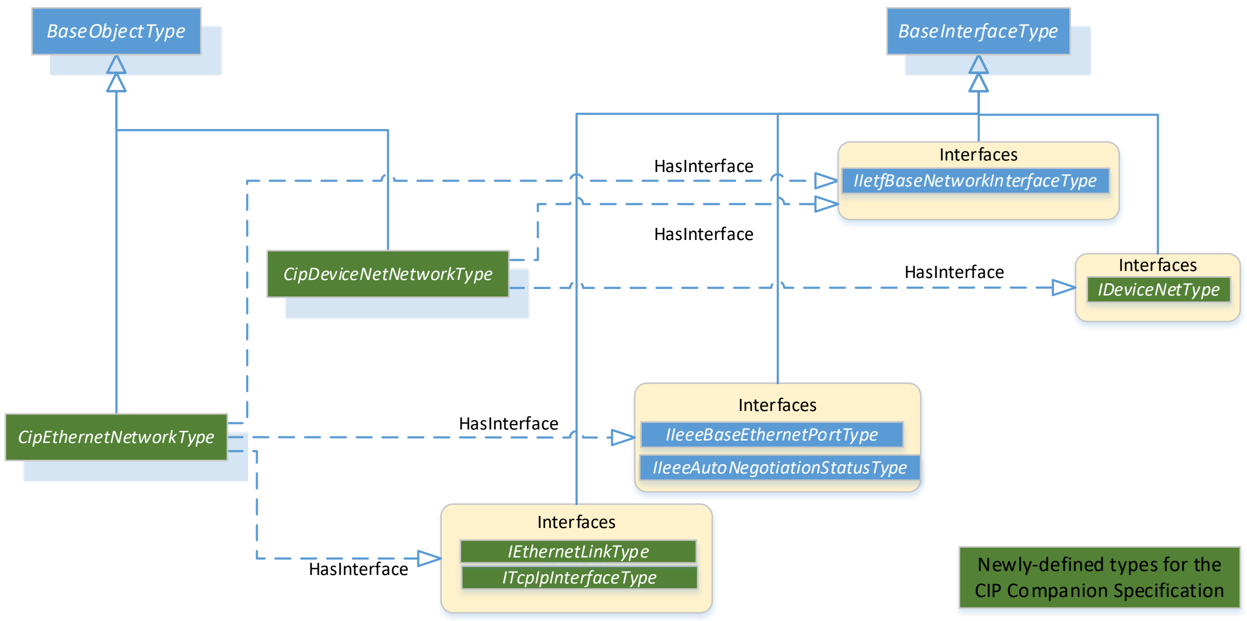

Figure 6 - CIP Network Types derived from Part 22

## Full CIP Device Model

Figure 7 shows the full information model for an EtherNet/IP device including the hierarchy of type inheritance.

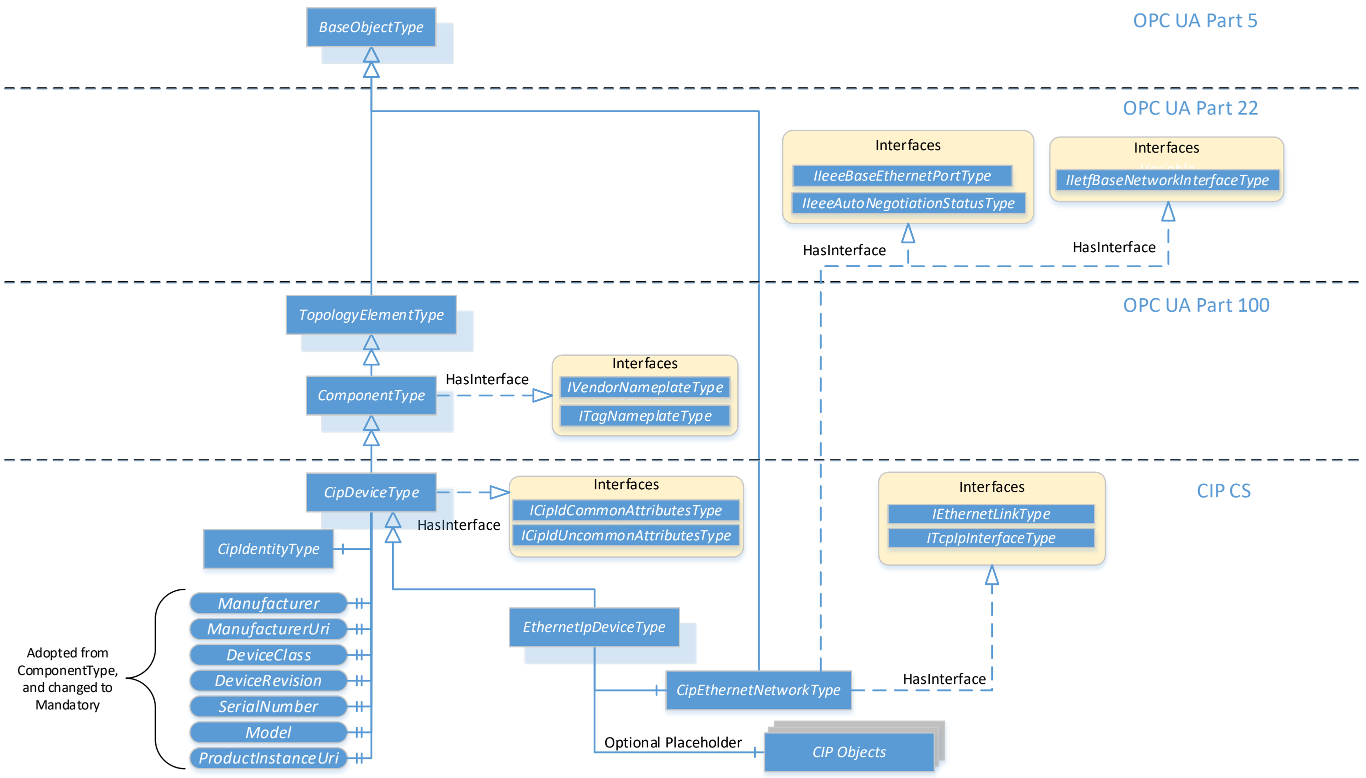

Figure 7 - CIP EtherNet/IP Device Information Model

## Generic CIP Object Model

The CIP specifications define dozens of application objects and vendors are free to define more of their own.  There are several alternatives to modeling these for OPC UA.  We could provide a translation for only the most common publicly defined objects.  That would provide some benefit but would neglect some applications due to the missing translations and it does not address vendor specific objects.  We could provide a translation for all publicly defined objects.  That would take a long time and still would not address vendor-specific objects.  The most comprehensive solution is to provide some type of generic mapping of the CIP object model to OPC UA.

Fig 7 shows that the EthernetIpDeviceType has an "Optional Placeholder" for CIP Objects.  OPC UA uses the Optional Placeholder modeling rule for open-ended situations.  It allows you to place any number of objects with an unspecified BrowseName at that point in the model.  We will use this to represent all other CIP objects in a device.

The details behind the GenericCipObjectType are still being developed.  Figure 8 demonstrates how this generic type might be used.  It shows a device that supports two instances of the Discrete Input Point Object.  It supports Class Attribute 1, and Instance Attributes 3 and 4.  The gateway would use some algorithm to generate unique BrowseNames for each Node in the model.  In this case we used the pattern "CIP Class X Instance Y Attribute Z".  For brevity the pattern could have been "CXIYAZ".  The important part is that the name is unique within the path.  Using this generic object, we believe we can represent any CIP object in OPC UA.

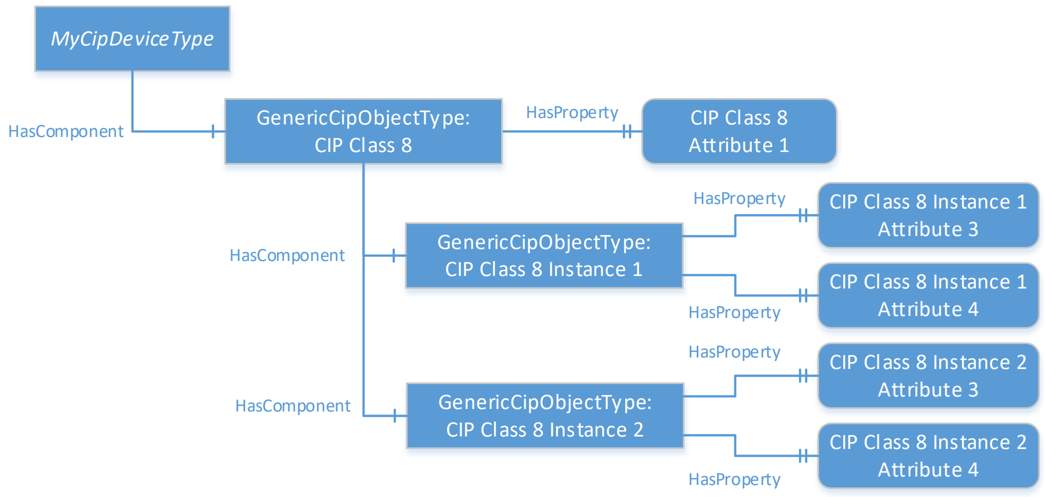
MyCipDeviceType
HasComponent
GenericCipObjectType: CIP Class 8
HasProperty
CIP Class 8 Attribute 1
HasComponent
GenericCipObjectType: CIP Class 8 Instance 1
HasProperty
CIP Class 8 Instance 1 Attribute 3
HasProperty
CIP Class 8 Instance 1 Attribute 4
HasComponent
GenericCipObjectType: CIP Class 8 Instance 2
HasProperty
CIP Class 8 Instance 2 Attribute 3
HasProperty
CIP Class 8 Instance 2 Attribute 4

*Figure 8 - Generic CIP Object Information Model*

**Example Integration with Other Information Models**

Let's finish with an example to understand how this model might be used.  The Downright Delicious Doughnut Corporation is a multinational company with production facilities around the world.  They buy machines and control equipment locally where their production facilities exist.  As such they need to manage equipment from many different suppliers.  They want to harmonize their asset management function so that it can be performed and accessed from any location.  In Figure 9 we see a model of their operations.

The high-level processes at DDD's locations are similar: dough is formed, cooked, drilled (likely the best way to make the holes in donuts), dressed, and packed in continuous batches.  Some equipment used at both factories is identical, while other equipment is different.  The Fried Division has standardized on EtherNet/IP, while the Baked Division uses *Inferior Netz*.  The two network technologies use different information to represent the identity of products even for identical pieces of equipment.  EtherNet/IP uses the numerical attribute values defined in the Identity Object.  Inferior Netz uses all string datatypes with a heavy use of animal names to differentiate products.  For both networks there are vendors that produce OPC UA gateway products.

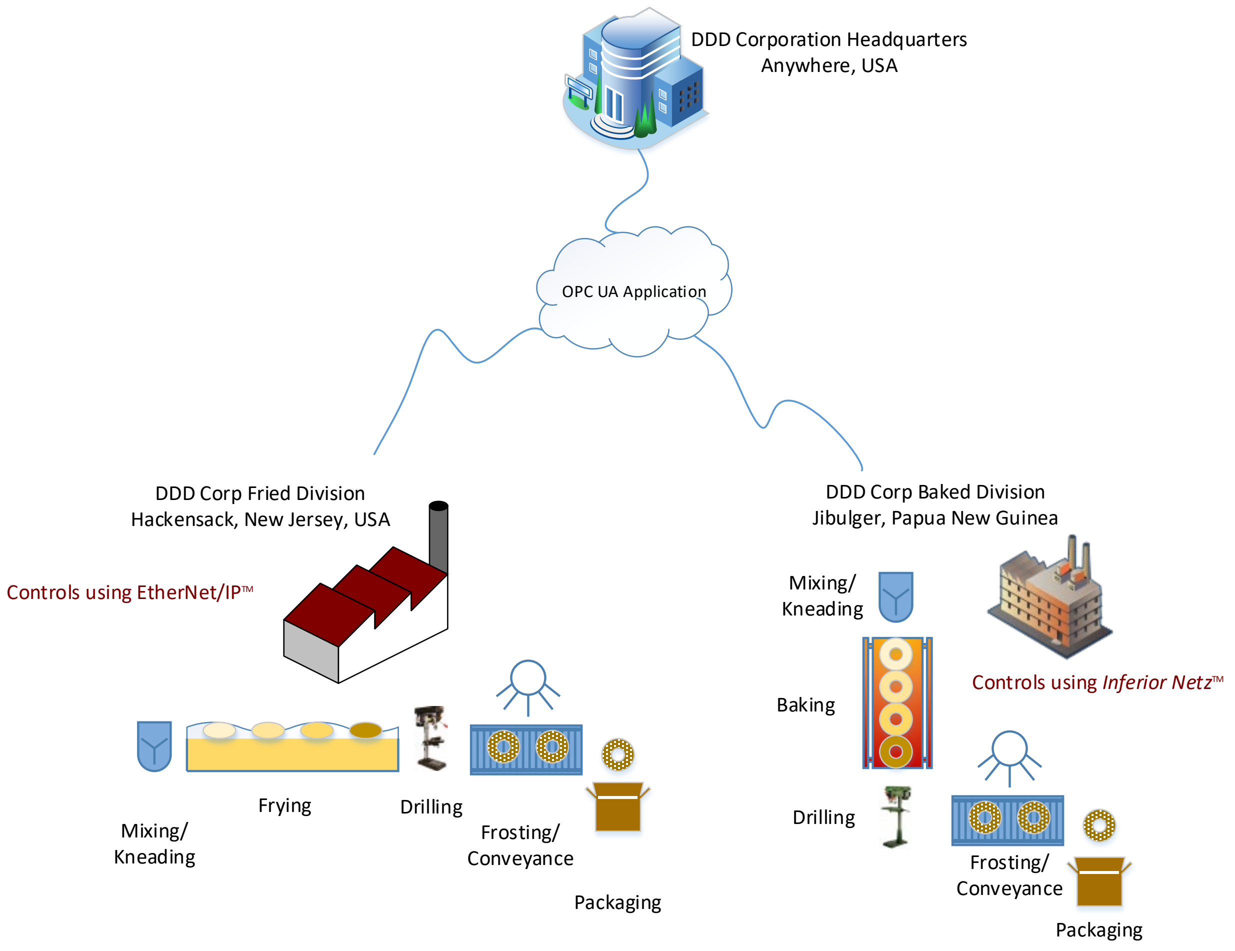

*Figure 9 - Multinational Company Managing Assets from a Centralized Location*

The Quality Assurance department at the Jibulger plant has recently reported uneven baking resulting in raw and burnt donuts that are out of specification.  Plant engineers researched the issue and determined that it could be addressed with a firmware update available for the oven.  Following this update, product is produced within specifications.  When headquarters learns of this issue and resolution, they order the asset management team to find all the firmware updates available for their equipment.

The asset management team maintains a database of all equipment in DDD's facilities using an OPC UA application.  The application was built using the Donut Industry Companion Specification (fictional, but certainly needed).  That specification defines object types to represent all the equipment used in the donut-making process (e.g., DrillPressType) and as shown in Figure 1, those types are built referencing interfaces defined in OPC's Part 100.

The asset management team finds a security bulletin from the drill press vendor indicating that attackers could change the direction of motion during operation.  This attack would lead to imperfect holes of varying sizes. The remedy is a firmware update from the vendor. This vendor uses EtherNet/IP controls and has identified the affected products using the Vendor ID, Device Type, Product Code and Revision. The team first searches the database for this Vendor's products. It uses the Manufacturer property from Part 100. As Table 2 shows, CIP products will have their numeric Vendor ID mapped to a vendor name. Once all occurrences of this vendor are found in the database, the asset management team can filter for the Device Type and Product Code. Both values are available because the gateway used at the Hackensack plant uses the CIP Companion Specification to incorporate common OPC data with CIP data. All devices are represented with the CipIdentityType which includes both the Identity Object information as well as the Part 100: Devices model.

This example demonstrates the merging of data from core OPC UA specifications, industry companion specifications, and the CIP object model.

## Conclusion

OPC UA continues to gain momentum as a vendor-independent mechanism to collect data. The publication of our companion specification will be essential to help our end users realize the potential of their intelligent devices. This paper discussed progress that has been made towards an OPC UA Companion Specification for CIP. There are still many challenges that need to be addressed like offline and online representation of all the data a device has to offer, metadata that fully describes CIP attributes, handling of complex data and services, and adoption of the UAFX asset model. All these issues still need to be investigated to achieve a feature rich solution that satisfies all the use cases we envision. The material we have today provides good coverage for many asset management functions and will provide immediate benefit if released as a first version while we continue to develop the model to cover more diverse use cases.

## Works Cited

[1] P. Brooks, K. Hopwood, F. Latino and S. Roby, *Use Cases for a CIP Companion Specification for OPC UA,* Palm Harbor: ODVA, Inc., 2020.

[2] ODVA, The CIP Networks Library, Volume 1: Common Industrial Protocol, PUB00001, Ann Arbor: ODVA, Inc., 2001-2021.

[3] ODVA, The CIP Networks Library, Volume 2: EtherNet/IP Adaptation of CIP, PUB000002, Ann Arbor: ODVA, Inc., 1999-2021.

[4] ODVA, The CIP Networks Library, Volume 7B: Integration of HART Devices into the CIP Architecture, Ann Arbor: ODVA, Inc., 2018-2021.

[5] ODVA, The CIP Networks Library, Volume 7C: Integration of IO-Link Devices into the CIP Architecture, Ann Arbor: ODVA, Inc., 2019-2021.

[6] ODVA, The CIP Networks Library, Volume 7A: Integration of Modbus Devices into the CIP Architecture, Ann Arbor: ODVA, Inc., 2007-2021.

[7] OPC Foundation, OPC Unified Architectue Part 5: Information Model, Scottsdale: OPC Foundation, Inc., 2006-2021.

[8] OPC Foundation, OPC Unified Architecture Part 100: Devices, Scottsdale: OPC Foundation, Inc., 2006-2021.

[9] OPC Foundation, *OPC Unified Architecture Part 22: Base Network Model,* Scottsdale: OPC Foundation, Inc., 2021.