Distributed System Communications in a Multi-Station Manufacturing Cell using OPC UA

Author(s):
Ryan Vallieu, Senior Automation Systems Architect, Data Science Automation, Inc.

NI Product(s) Used:
LabVIEW 2016
LabVIEW Data Logging and Supervisory Control Module (DSC)
LabVIEW OPC UA Toolkit

Category:
Machine Control

The Challenge
Providing an easy to use extensible and adaptable data communication method for an automation assisted product assembly line employing a distributed system architecture with the need to allow future expansion of data access outside of the LabVIEW environment/application space.

The Solution
DSA was able to leverage the NI OPC UA Toolkit to create an OPC UA suite containing a Server, Client Tag Browser, and Tag Access Library that could be utilized by our customer for communicating between multiple distributed LabVIEW applications and third-party commercial off the shelf OPC UA compliant hardware.

Introduction
For 25 years, Data Science Automation® (DSA) has been a premier automation systems integrator, leveraging commercial off-the-shelf tools in the design and implementation of custom-engineered, complete, and highly-adaptive solutions in laboratory automation, embedded/new product development, manufacturing and test automation. The company provides an extensive array of automation engineering, programming, consulting & training services to dramatically improve research, manufacturing, government & business operations. DSA is fast and methodical, staffed with exceptional, multi-disciplinary, NI Certified professionals that consistently apply CSIA-certified best practices to deliver the lowest total cost of ownership.

Our customer opted to have one of our engineers provide onsite custom training regarding the implementation of these OPC UA tools and developer library toolkit with their actual system. They desired support transitioning from the early examples leveraging Configuration Files for the OPC UA Tag Definitions to their desired SQL database for maintaining the Tag Specification data. Prior to this DSA taught some of the customer engineers in the formal National Instruments LabVIEW classes, most specifically applicable being the LabVIEW Connectivity class.

Help for the humans
DSA’s client was building a distributed production line for their product <hydraulic valve> assembly process. The line consisted of several stations with automated testing in place after manual assembly actions were performed by the operators. Part selection is performed under the guidance of a camera based light guide system. Assembly inspection at various points involved unit testing, motion control, visions system part inspection, and mechanical testing. The new production line was being built to provide higher throughput while at the same time lowering instances of incorrect assemblies to provide higher product yield.

Distributed Machine and Software Architecture
Our customer had decided early on to leverage LabVIEW for the software to run each station due to the adaptability of the LabVIEW environment and the ability to communicate with all of the third party hardware selected for the system such as digital and analog IO modules, bar code scanners, pneumatic tester, RFID readers, etc. There were at least six Windows based computers in the system, basically broken out to cover one station per computer. Each would be running a specific LabVIEW application. Each application and the overall
system needed to be able to share data across the line and in the future provide visibility up to the Enterprise level and possibly data Historians to allow for production data analysis. Our client was unsure which Ethernet based communication protocol would best serve their needs of multi-system data communication and future expandability and adaptability, realizing they could have leveraged TCP/IP, Classic OPC, or Network Shared Variables and the Network Shared Variables API, OPC UA, etc from within the LabVIEW environment. DSA recommended utilizing OPC UA for the Client/Server data communication architecture as OPC UA was designed to meet the machine-to-machine communication requirements presented by our client in a platform independent extensible framework.

Hosting the Party
DSA developed a stand-alone OPC UA Server Application for our client to leverage in their system. They had the choice of either running it as a stand-alone application or copying the example code and placing it in parallel with their system code. The code was written to leverage two possible sources of tag specifications; a LabVIEW Configuration File, or an SQL database. The OPC UA implementation in LabVIEW provides for creation of Tags, and Properties of Tags and also allows for creations of Folders so that a hierarchical data structure can be created. This makes viewing the tags and groupings the tags much easier for the system developers.

At the outset of the project our client was unsure if they were going to be able to provide a centrally hosted database or rely on a configuration file for storing the OPC UA Tag Specifications. Thus DSA created an example leveraging the LabVIEW Configuration File format, while assuring that the base Server architecture could handle Tag specification being sourced from any repository type. A Tag Specification File reader was created that would read the file and parse the information into the correct selected Tag Specification Cluster format as seen in Figure 1. Later in the project the OPC UA Server code was easily updated to obtain the information from a SQL database. This was a painless switch as the OPC UA Server Software was not written with the source of the Tag Specification hard coded into the module since our developers knew to program in this adaptability.

![OPC UA Tag Configuration Cluster](image)

**Figure 1 - OPC UA Tag Configuration Cluster**

The Node Type field allows for defining whether the item being created in the OPC UA Server is a Folder, Item, or Property. Items and Property are tags that the software can manipulate. Folders are merely containers that allow
for the hierarchical implementations of the Tags. Owner allows specifying the containers above the Folder, Item, or Property being created. Name defines the name of the Folder, Item or Property.

Once the Tag Specification is loaded, the Server can be launched and the tags created. After the Server is created client applications may connect and interact with the tags. Figure 2 shows the front panel of the DSA OPC UA Server implementation.

![Figure 2 - OPC UA Server GUI](image)

The front panel of the OPC UA Server allows the developer to preview the Tag hierarchy in a Tree Control. This allow for easy interaction to view the tags and to interact with the Tag Viewer for Data Forcing if needed. An example Tag Viewer is shown below. The example Tag shown in the viewer was created as a way for developers testing the system on remote computers to send messages to each other.
Tag value forcing can be useful for verifying tag communication and system function prior to completing all of the software. Read only tags can have the value forced from the Server since the Server has the permissions to make these changes and hosts the tags. This is useful for beta testing communications and system functionality through manual manipulation of tags if needed.

**Dropping in on the Conversation**

In addition to the Server LabVIEW module developed by DSA to allow our customers to host OPC UA Tags for inter-process and inter-machine communications an OPC UA Client Tag Browser was also created. This was provided as a stand-alone executable. This tool can be installed on any of the customers PCs and allow the operator to connect to an OPC UA Server. The GUI is similar to the OPC UA Server implementation shown above and can be seen in Figure 4.
The DSA OPC UA Browser allows for connection to any OPC UA Server provided the correct security certificates are in place. The operator simply has to enter the correct Server URL and press <Connect and Browse> and the OPC UA Browser will recursively search for all of the hosted Tags on the Server. This information is presented on the Server Browser tab as shown in Figure 5.
The OPC UA Browser application allows for remote viewing of tags from a computer not utilized on the production line, thus a support engineer could view tags of interest during the production runs without utilizing one of the production computers and potentially interfering with assembly technician workflow. The OPC UA Browser also allows for verification of the Server deployment. The Tags can be selected for Viewing/Forcing by double clicking on the Tag of choice in the Tree.

Figure 5 - OPC UA Browser - Server Browser Tab with Tag Tree
The Browser Tag Viewer is only able to force tags that have write permission set for the Access Level at the time of the tag creation on the OPC UA Server. More than one Tag Viewer can be open at a time if needed.

The last tab on the Browser interface allows the user to view a table of current tag values as seen in Figure 7. This tab allows the user to view the tag values in one location and the tag values update when any of the tags on the list receive a new value. This is accomplished by leveraging the OPC UA capability to trigger a value change event on a User Event Structure. Subscriptions can be altered on the fly by providing lists of tags of interest to the subscription VI and the User Event Structure Code automatically updates based on the latest list of value changes. All tags are converted to string for viewing on the table. The individual Tag Viewer can also be opened from this screen by double clicking on a tag in the visible list.
Power to the Developer

The last piece of the OPC UA suite provided by DSA to our customers is the Generic Client Tag Access toolkit. This toolkit is provided for developers to utilize the OPC UA suite leveraging a LabVIEW lvlib with Application Programming Interface (API) VIs provided as Polymorphic instances. Example VIs are included with the toolkit.

The toolkit comes in two ‘flavors’ for implementation into larger projects. One implementation allows the developer to utilize tag inputs as String inputs. Separate levels of the API allow for Connection Management, Reading Tags, Writing Tags and Requesting OPC UA References.

The Init/Connect VI allows the developer to specify the IP Address and Port of the Server and to also specify the “Top Level Folder” under which the Tags are created. This is a suggested implementation method for tag creation hierarchy – to separate specific application areas under a top level folder. Only one top level folder can be specified by the client in each application instance of the client, or copy of the client library. The encapsulated core VI that internally maintains the client reference also stores this top folder level so that the developer is not required to add it to the ‘Owner’ string input. Of course the ability to override using the saved top level folder is included with the read/write level API VIs. In this instance the developer can fill out the whole “Owner” chain to access any tag on the Server.

The examples include two application VIs that run in parallel that handshake each other with a Handshake Count and Ready tags for Client 1 and Client 2. The example application VIs also include code demonstrating how to register Tags for Value Change Events using the ‘Message’ tag that allows users accessing the individual applications to send simple text messages back and forth.
Figure 8 - Client Tag Access Library Example Initialization

Figure 9 - Client Tag Access Library Example Main Loop
The other Client Tag Access API that is provided for developers to utilize in their applications allows the use of a Type Defined Enum to specify the tags. The main base API is still utilized for Connection Management and accessing the Client References, but the String Read and Write API VIs are replaced with the Enum read and write API version. The Enum API was added as a separate API at the time of initial release in order to allow the Write Polymorphic VIs to leverage the “Allow Polymorphic VI to Adapt to Data Type” setting on the Polymorphic VI. It is perfectly fine for the developer to also mix the utilization of the tag API types such as for accessing the OPC UA Server level tags using the string implementation if they are not added to the Enum, see Figure 11. The OPC UA Server Tags are created by default as specified by the OPC UA Specifications outside of the Tag specifications outlined by the system developers for their particular application.

As can be seen in the Figure 11 the Enum Read/Write API VIs do away with the Owner String input and Tag Name String input and rely on the Type Defined Enum being wired into the Variant Input terminal on the top middle connector pane terminal. If a non-Enum data type is wired into the OPC UA Enum API Read and Write VIs a LabVIEW Error 1 is generated indicating that the datatype wired is incorrect. Any tag name entered that is incorrect will generate the usual OPC UA LabVIEW Error Codes.
Figure 11 - Example of Enumerated Data Type Definition for Tag Access

The data types for the Tag Value Read and Write VIs match those in the native LabVIEW OPC UA VIs provided by National Instruments. This toolkit provides encapsulation of the OPC UA and makes utilization and management easier. DSA created a utility VI that takes in the Tag Specification information and updates the desired Enum Type Definition control for the developer.

Conclusion
The NI OPC UA Suite created by Data Science Automation based upon the National Instruments OPC UA Toolkit allowed us to meet our customer’s requirements for distributed system communications between concurrently running LabVIEW applications on a production line. We were able to deploy successfully on the first day and share data from across systems. The selection of this 3rd party protocol guaranteed that the system would be able to expand to include other targets running non-Windows operating systems, such as an NI CompactRIO or Compact Vision System, or even third party OPC UA Compliant tools such as Enterprise Historians that they might desire to use.