MatrikonOPC

OPC: The Ins and Outs to What It's About

"The Every Man's Guide to OPC"

Darek Kominek, P. Eng. Alberta, Canada - 2009

Executive Summary

The Every Man's Guide to OPC is an easy-to-read overview of the most popular industrial open connectivity standard - OPC. This paper first introduces the main idea behind OPC, shows why OPC is different from conventional (often proprietary) communication protocols, and explains how OPC helps overcome the limitations of such native protocols.

Next, the paper explains what the building blocks of OPC are (OPC Clients and OPC Servers) and how they work together to make data-sharing possible. Using clear illustrations and minimal technical jargon, The Every Man's Guide to OPC helps readers of all technical levels quickly grasp what OPC is and how they can use it to establish inter-connectivity in their own environments.

HOW OPC SOLVES AUTOMATION'S DATA CONNECTIVITY PROBLEM

Today, automation is used prominently in every major industry. While different industries often use different specialized devices, control systems, and applications, they all share a common, rapidly growing challenge - how to share data both amongst all these components and the rest of the enterprise.

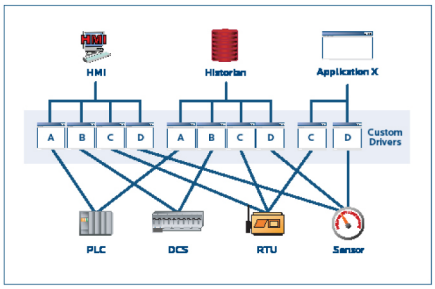

Figure 1: Custom Driver Problem - Each application requires a device or protocol specific driver to allow it to communicate with each respective device. Drivers are not re-usable between applications because each application uses its own data format(s).

Before looking at what OPC is and how it goes about solving one of automation's biggest communications headaches, it is worth looking at what the problems were in the first place. Following is a list of factors that have traditionally caused the biggest data sharing issues, followed by a brief explanation of what impact OPC has hadon each issue:

Proprietary Protocols: Vendors often used proprietary protocols that allowed products from a particular product line to communicate among each other, but required custom drivers to communicate with other vendors' products. To make matters worse, different product lines from the same vendor often did not communicate amongst each other, which necessitated the need for additional connectors.

OPC resolves this by making it unnecessary for the Data Sink to know anything about how the Data Source communicates or organizes its data.

Custom Drivers: Every end-to-end connection required a custom driver to facilitate communications between specific endpoints. For example, if an HMI needed to communicate with a PLC, it required a custom HMI driver written for the specific protocol used by the PLC. If this PLC data was to be historized, the historian required its own driver because the HMI's custom driver could only be used to communicate with the HMI, not the historian. If a custom driver for the specific endpoints was not available, data communications were difficult and expensive to establish.

"OPC eliminates the need for custom drivers between each new application and Data Source."

OPC eliminates the need for custom drivers between each new application and Data Source. In Figure 1, a single standard PLC driver could be shared by both the HMI and the Historian via an OPC connector with the added benefit that the OPC connector would only require a single connection to the PLC – reducing controller loading.

Complex Integration: The use of custom drivers between every endpoint meant that even a small number of devices and applications quickly involved the use of many drivers. The same HMI running on multiple computers, all communicating with the same device, required multiple installations and configurations of the same driver on each computer. If the HMIs communicated with additional devices, each HMI required its own set of custom drivers for each of the devices. This created a version maintenance nightmare.

Using OPC greatly simplifies integration because once an OPC Connector for a particular Data Source is configured, all OPC-enabled applications can start sharing data with that Data Source with no concern for additional custom drivers.

"Traffic and hence device loading is greatly reduced by using OPC Connectors [because all clients use a single connection to the data source.]"

Device and Controller Loading: Each driver establishes its own connection to the device or controller that it is designed to communicate with. Given the large number of custom drivers being used in a typical installation, the controller was often bombarded by many requests for the same information from every application that it needed to communicate with. In addition, many devices could only accept a limited number of simultaneous connections. If the number of drivers trying to connect to a device exceeded the number of connections it had, further workarounds were needed.

Traffic, and hence device loading, is greatly reduced by using OPC Connectors because a single Device-specific OPC connector requires only a single connection to the Data Source while simultaneously communicating with many applications.

Obsolescence of Legacy Infrastructure: As vendors release new products they eventually stop supporting older ones. When a new version of an HMI comes out, it may require its own set of device drivers that sometimes may no longer support communications with a device the previous version of the HMI supported.

OPC extends the useful life of legacy systems because once an OPC connector for a legacy system is configured, it allows any OPC-enabled application (most are) to communicate with the legacy system regardless of whether the application natively supports communication with the legacy system or not. Thus, OPC allows the newest applications to continue communicating with the oldest systems.

"OPC extends the useful life of legacy systems... Thus, OPC allows the newest applications to continue communicating with the oldest systems."

Enterprise Wide Data Connectivity: As the need for automation data grows throughout the enterprise, data-connectivity problems are compounded because applications from the corporate side were not designed to communicate with devices and controllers. This can potentially add extra load to the automation infrastructure and raise various automation security concerns.

OPC makes true enterprise-wide automation data sharing possible by allowing approved applications to share data with automation Data Sources without the need for installing custom drivers—all that's required is an OPC connector.

Are there some Real-World Examples of OPC in Action?

Yes. For some real-world examples of how OPC has been used to resolve various third- party data connectivity issues check out these OPC Case studies (http://www.matrikonopc.com/resources/case-studies.aspx).

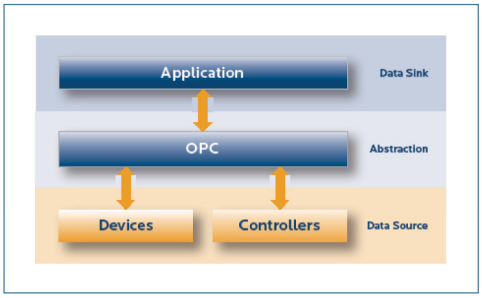

Figure 2: OPC serves as an abstraction layer between Data Sources and Data Sinks - enabling intercommunication without either side having to know the other's native protocol.

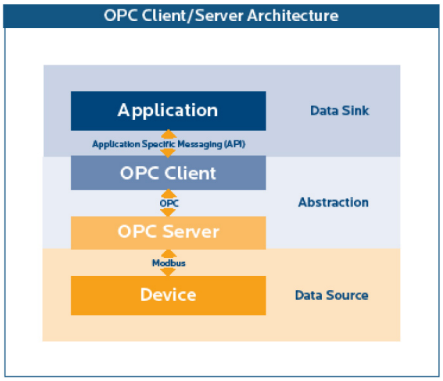

Figure 3: OPC Client/Server Architecture - A closer look at the OPC abstraction layer reveals two components: an OPC Client and an OPC Server. OPC specifications define the messaging between these two components.

INTRODUCTION TO OPC

What is OPC?

OPC is the world's most popular standards-based data-connectivity method. It is used to answer one of the automation industry's biggest challenges: how to communicate between devices, controllers, and/or applications without getting caught up in the usual custom driver-based connectivity problems.

Why OPC Succeeds where Custom Drivers Fail

The key to OPC's success in creating truly vendor-independent communications is that OPC abstracts the Data Source (e.g., PLC) and Data Sink (e.g., HMI) implementation details from each side so data can be exchanged between them without requiring them to know anything about each other's native communications protocol and internal data organization. This is in sharp contrast to the custom driver approach of writing applications that, by definition, are required to natively communicate with both the Data Source and the Data Sink.

How OPC Communication works (Conceptual)

OPC can be represented as an "abstraction" layer that sits between the Data Source and the Data Sink, allowing them to exchange data without knowing anything about each other.

How OPC Works (Functional View)

The OPC "device abstraction" is realized by using two, specialized OPC components called an OPC Client and OPC Server. Each of which is described in a following section. What's important to note is that just because the Data Source and Data Sink can communicate with each other via OPC does not mean their respective native protocols are no longer necessary or have been replaced by OPC. Instead, these native protocols and/or interfaces are still present, but only communicate with one of the two OPC components. In turn, the OPC components exchange information amongst each other and the loop is closed. Data can travel from the Application to the Device without having one talk directly to the other.

Benefits of using OPC Connectivity

At first glance, trading a single Custom Driver for two OPC components (OPC Client and OPC Server) may not look like much of an improvement but as experience has shown, it is. Following are some key benefits of using OPC:

1. An OPC enabled Application can freely communicate with any OPC-enabled Data Source visible to it on the network without the need for any driver software, specific to the Data Source.
2. OPC-enabled applications can communicate with as many OPC-enabled Data Sources as they need. There is no inherent OPC limitation on the number of connections made.
3. Today OPC is so common that there's an OPC connector available for almost every modern and legacy device on the market. It's easy to get started using OPC.

4. OPC-enabled Data Sources may be swapped out, exchanged, or upgraded without the need to update the drivers used by each application (Data Sink) communicating with the Data Source via OPC. Only the OPC Server for that Data Source needs to be kept current.
5. Users are free to choose the best-suited devices, controllers, and applications for their projects without worrying about which vendor each came from and whether they will communicate with each other… inter-communication is now assumed!

"OPC addresses the challenges of working with different data categories by independently specifying how each one is to be transmitted."

What Types of Data does OPC Support?

The most common types of Automation data transferred between devices, controllers, and applications break down into three broad categories:

- Real-time data
- Historical data
- Alarm & Event data

Each of the above also supports a wide range of value types. Some common examples of these data types are integer, floating point, string, date, and various array types to name a few. OPC addresses the challenges of working with these different data categories by independently specifying how each one is to be transmitted via the OPC Client and OPC Server architecture.

The three OPC specifications corresponding to the three data categories are:

1. OPC Data Access Specification (OPC DA) – used to transport real-time data
2. OPC Historical Data Access Specification (OPC HDA) – used to transport historical data
3. OPC Alarms & Events Specification (OPC A&E) – used to transport alarming information

Do All OPC Connectors Support all OPC Specifications?

No. OPC connectors are not required to support all of the OPC specifications. Historically, most common were OPC servers that supported real-time data followed by OPC HDA implementations. It is prudent to inquire what OPC specifications an OPC connector supports before choosing it for use in a project.

"It is prudent to inquire what OPC specifications an OPC connector supports before choosing it for use in a project."

Does it Matter what OPC Specification an OPC Client or OPC Server Supports?

Yes, this is crucial. While all three OPC Specifications (OPC DA, OPC HDA, OPC A&E) use the same underlying OPC Client/Server architecture to transfer the different data category types, both the OPC Client and OPC Server must support the same OPC specification to properly coordinate passing data between each other, and to work correctly with the Data Sink and Data Source respectively.

OPC SERVERS

What is an OPC Server?

An OPC Server is a software application, a "standardized" driver, specifically written to comply with one or more OPC specifications. The word "server" in "OPC Server" does not refer to the type of computer being used but instead reflects its relationship with its OPC counterpart, the OPC Client.

What Do OPC Servers Do?

OPC Servers are connectors that may be thought of as translators between the OPC world and a Data Source's native communication protocol or interface. Since OPC is bi-directional, this means OPC Servers can both read-from and write-to a Data Source. The OPC Client/OPC Server relationship is a Master/Slave one which means one OPC Server will only transfer data to/from a Data Source if an OPC Client commands it to.

What Types of Data Sources (Devices) can OPC Servers Communicate With?

"OPC Servers can communicate with virtually any Data Source."

OPC Servers can communicate with virtually any Data Source whose output can be read or written to via electronic means. A brief list of possible Data Sources includes: devices, PLCs, DCSs, RTUs, electronic scales, databases, historians, web-pages, and automatically updating CSV files. To communicate with any of these devices requires only the use of an OPC Server that employs the appropriate native protocol or interface. Once such an OPC Server is configured, any OPC enabled application (with permission) can begin communicating with the Data Source without concern for how the Data Source communicates natively.

Where can I Find an OPC Server for Device X?

While many vendors provide OPC Servers with their devices, controllers, and applications, there are many who do not. MatrikonOPC is the world's largest provider of high-quality, OPC connectors for hundreds of devices. A good place to start is on the MatrikonOPC Server website, or by calling MatrikonOPC directly.

How do OPC Servers Work?

While users do not need to know anything about the inner workings of OPC Servers to be able to use them, a conceptual understanding of what goes on under the hood helps shed light on why the quality and performance of different vendors' OPC Servers vary greatly.

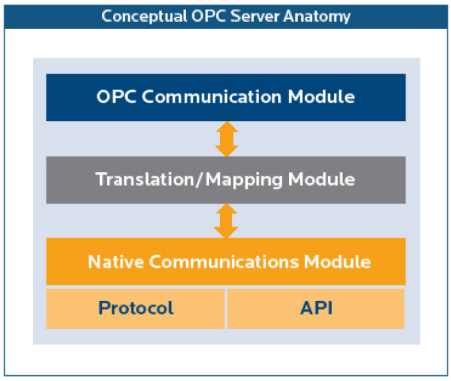

Figure 4: Conceptual OPC Server Anatomy - Conceptually, an OPC Server can be broken down into three modules: OPC Communications module, Translation/Mapping module, and a Native Communications module.

A conceptual view of the inner workings of an OPC Servers looks as follows:

- ***OPC Communications Module:*** This part of the OPC Server is responsible for properly communicating with a given OPC Client. Well written OPC Servers must be fully compliant with the OPC Specification(s) they implement to ensure they properly communicate with OPC Clients.

- ***Native Communications Module:*** The OPC Server should employ the most efficient method of communicating with the Data Source. In some cases this means connecting to the Data Source directly via its native protocol, while in other cases, this means communicating with the Data Source via its custom driver via an Application Programming Interface (API). Typically, the more experience the OPC Server vendor has with the device, the better the OPC Server will utilize the device's communications options.

- ***Translation/Mapping Module:*** This is where all the "magic" in an OPC Server happens. This module is tasked with properly interpreting the arriving OPC requests from the OPC Client and in turning them into proper native requests that get sent to the Data Source and vice versa. If this is done efficiently, the OPC vendor can keep Data Source loading to a minimum while maximizing data throughput.

Can an OPC Server from one Vendor Communicate with OPC Clients from Other Vendors?

Yes, assuming both the OPC Client and OPC Server are compliant with the same OPC specifications, they should be capable of communicating with each other regardless of which vendor each OPC component came from.

Can OPC Servers Share Information with other OPC Servers?

OPC Servers do not communicate directly with each other; they are only designed to communicate with OPC Clients. There are however, OPC utilities like the MatrikonOPC Data Manager (http://www.matrikonopc.com/products/opc-data-management/opc-data-manager.aspx), designed to specifically make this OPC Server-to-OPC Server communication trivial.

"Well-written OPC Servers must be fully compliant with the OPC specification(s) they implement."

OPC CLIENTS

What is an OPC Client?

An OPC client is software written to communicate with OPC connectors. It uses messaging defined by a specific OPC Foundation specification.

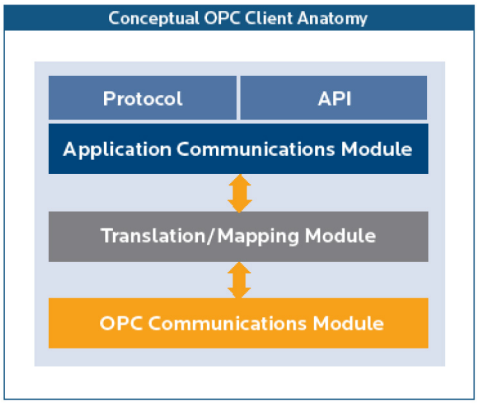

Figure 5: Conceptual OPC Client Anatomy - Mirroring the OPC Server, an OPC Client can also be thought of as consisting of three modules: Native application communications, Translation/ Mapping module, and a Communications module.

What does an OPC Client do?

Conceptually: OPC Clients represent a data-sink. They initiate and control communications with OPC Servers based on what the application they are embedded in requests of them. OPC Clients translate a given application's communication requests into an OPC equivalent request and send it to the appropriate OPC Server for processing. In turn, when OPC data returns from the OPC Server, the OPC Client translates it back into the application's native format so the application can properly work with the data.

Technically: OPC Clients are software modules used by an application to enable it to communicate with any compliant OPC Server visible to it on the network. Typically, OPC Clients are embedded in applications such as HMIs, trending packages, historians, and report writers to make them inherently OPC-enabled.

It is common to refer to the application with an OPC Client embedded in it as the "OPC Client" even though only the OPC implementation is the true OPC Client.

Can OPC Clients Simultaneously Communicate with Multiple Devices (OPC Servers)?

There are two parts to this answer:

1. First, Semantics: It's important to remember that OPC Clients are by design only capable of communicating with OPC Servers, not the end devices or other data sources. This is necessary because OPC Clients must remain protocol independent, otherwise they would fall into the original device-driver trap of the past.
2. Yes, OPC Clients can communicate with multiple OPC Servers at the same time. Effectively, this means an OPC Client can read and write data to and from multiple data sources via their respective OPC Servers.

How does an OPC Client Work?

As with the OPC Server, the OPC Client can be conceptually broken down into three modules that include: the Application Communications Module, Translation/Mapping Module, and OPC Communications Module. Each of whose functions are described below.

"OPC Clients can communicate with multiple OPC Servers at the same time."

OPC Communication Module: While not as involved as the OPC Server (the OPC Server portion is more complicated) it is still crucial for the OPC Client to behave correctly as it connects with an OPC Server, exchanges data with it, and disconnects without destabilizing the OPC Server.

Application Communications Module: The OPC Client is typically written to work within a specific application, so it relies on a few calls to the Application's Programming Interface (API) to allow data to be passed from the application down to the OPC Server/Data-Source via the OPC Client. It is also possible for a generic OPC Client to communicate with an application via a protocol rather an API if the application supports such a protocol. (An example of this is the MatrikonOPC Client for ODBC which uses SQL statements over ODBC to communicate with a Relation Database application.)

"…there is no theoretical limit placed on how many OPC Servers an OPC Client can connect to."

Translation/Mapping Module: A key function of the OPC Client is to bi-directionally translate information as the application it represents requests data to be read-from or written-to the end device or Data-Source.

How many OPC Servers can an OPC Client connect to?

The short answer is—as many as it needs to. Within the OPC framework, there is no theoretical limit placed on how many OPC Servers an OPC Client can connect to.

Can OPC Clients communicate directly with other OPC Clients?

No. OPC Client-to-OPC Client communications are not defined in OPC. Only the OPC Client /OPC Server architecture is supported. However, if an application is expected to provide OPC Data to other clients, it needs to have an OPC Server of its own. This OPC Server will then allow OPC Clients from other applications to use this application as an OPC Data Source.

Where is the OPC Client Installed?

OPC Clients are typically built into the applications that use them, such as HMIs or historians for example. If for some reason the application at hand does not have a built in OPC Client, one may be available from the Application vendor or a third-party OPC vendor like MatrikonOPC. An OPC Client external to the application would typically communicate with the application via one of its native protocols. In this case, the OPC Client would not even have to reside on the same computer as the application.

BRINGING IT ALL TOGETHER

This paper explained how OPC solves modern industry's growing challenge of accessing and sharing data between devices, controllers, and applications regardless of what their native communications are or what vendor they are from. A high-level description illustrated what OPC is and then explained what OPC Clients and OPC Servers (OPC's building blocks) are and the roles they play in OPC communications. While in-depth knowledge of OPC's inner-workings is not a requirement for using it, even a brief familiarity with its main concepts is beneficial.

"OPC's hallmark "plug-and-play" approach to data connectivity lead to its rapid rise in popularity."

OPC's hallmark "plug-and-play" approach to data connectivity lead to its rapid rise in popularity, making it the world's most popular data-connectivity technology. Thanks to its versatility, OPC is used at all enterprise levels and across all industrial verticals.

Toll Free 1-877-628-7456
Ph: +1 (780) 945-4099
Email: info@MatrikonOPC.com
Web: www.MatrikonOPC.com