OPC Security White Paper #1

Understanding OPC and How it is Deployed

byres research
intrinsically secure

po box 178
unit#5 – 7217 Lantzville rd
lantzville, bc
canada v0r 2h0
office 250.390.1333
fax 250.390.3899
www.byressecurity.com

Digital Bond

suite 130
1580 sawgrass corp pkwy
sunrise, FL 33323
office 954.315.4633
www.digitalbond.com

PREPARED BY:
Digital Bond
British Columbia Institute of Technology
Byres Research

July 27, 2007
OPC Security WP 1 (Version 1-3b).doc

Revision History

Revision	Date	Author	Details
0.7	May 15, 2006	E. Byres, J. Carter, M Franz, W. Henning, J. Karsch	Draft internal review version
1.0	May 31, 2006	E. Byres, J. Carter, M Franz W. Henning, J. Karsch	Draft for controlled public review
1.1	August 31, 2006	E. Byres, M. Franz	2nd Draft for controlled public review
1.2	February 9, 2007	E. Byres, D. Pederson	3rd Draft for controlled public review
1.3	April 3, 2007		Public Release Version
1.3a	June 27, 2007		Grammatical errors corrected
1.3b	July 27, 2007		Grammatical errors corrected

Acknowledgements

The Group for Advanced Information Technology (GAIT) at the British Columbia Institute of Technology (BCIT), Digital Bond, and Byres Research would like to thank all the vendors and end users that generously supported our efforts through numerous interviews and by providing us with documents that could only be described as extremely sensitive. Unfortunately we can not name you for obvious security reasons, but we appreciate your time, trust and encouragement.

Two people stood out in their contributions and advice for this document that we would like to acknowledge. These people are Bill Cotter and Chip Lee. Thank you for all your help.

Finally we would like to thank Evan Hand, formerly of Kraft Foods Limited, for his vision and support. Without him, this project never would have been possible.

Disclaimer

Deployment or application of any of the opinions, suggestions or configuration included in this report are the sole responsibility of the reader and are offered without warrantee of any kind by the authors.

byres research
intrinsically secure

Table of Contents

Executive Summary

In recent years, Supervisory Control and Data Acquisition (SCADA), process control and industrial manufacturing systems have increasingly relied on commercial Information Technologies (IT) such as Ethernet™, Transmission Control Protocol/Internet Protocol (TCP/IP) and Windows® for both critical and non-critical communications. This has made the interfacing of industrial control equipment much easier, but has resulted in significantly increased connectivity to the outside world, which in turn results in an increase in the risk of cyber-based attacks impacting industrial production and human safety.

Nowhere is this benefit/risk combination more pronounced than the wide-spread adoption of Object Linking and Embedding (OLE) for Process Control (OPC). OPC is increasingly being used to interconnect Human Machine Interface (HMI) workstations, data historians and other hosts on the control network with enterprise databases, Enterprise Resource Planning (ERP) systems and other business oriented software. Unfortunately, securely deploying OPC applications has proven to be a challenge for most engineers and technicians. While OPC is an open protocol with the specifications freely available, engineers must wade through a large amount of very detailed information to answer even the most basic OPC security questions.

To address this need for security guidance on OPC deployment, a joint research team with staff from BCIT, Byres Research and Digital Bond were commissioned by Kraft Foods Inc. to investigate current practices for OPC security. The results of this study were then used to create three white papers that:

1. Provide an overview of OPC Technology and how it is actually deployed in industry
2. Outline the risks and vulnerabilities incurred in deploying OPC in a control systems environment
3. Summarizes current good practices for securing OPC applications running on Windows-based hosts.

The white paper you are now reading is the first of the three and explains what OPC is and describes the results of a survey of OPC end-users on how OPC is actually used in the field. It is intended to be read and understood by IT administrators and control systems engineers/technicians rather than OPC programming or security experts

The results of the end-user survey indicate that OPC is not just used for data management purposes on the plant floor, but instead is a critical component of many production systems. Over quarter of the end-users reported that loss of OPC communications would result in a loss of production. In addition,

approximately 20% of the companies reported deploying OPC over site business networks, enterprise networks or corporate Intranets and approximately 10% used OPC over the Internet itself. All these results highlight the urgent need for better OPC security.

The challenges of securing OPC deployments are also clear. The inherent architectural complexity of OPC, the default security posture of many OPC servers, and the lack of unambiguous guidance with regard to security all contribute to the difficulties of securing OPC deployments. In addition, OPC's reliance upon the Microsoft platform is both a curse and a blessing - while Windows has flaws, there are a wealth of practices for hardening Windows servers that can be applied to OPC clients and servers. In the follow-on white papers these solutions are discussed in detail.

1 Introduction

This report is the first of three white papers outlining the findings from a study on OPC security conducted by Byres Research, Digital Bond and the British Columbia Institute of Technology. The objective of this study was to create a series of simple, authoritative white papers that summarized current good practices for securing OPC client and server applications running on Windows-based hosts. The full study is divided into three Good Practice Guides for Securing OPC as follows:

- **OPC Security White Paper #1 – Understanding OPC and How it is Used:** An introduction to what OPC is, what are its basic components and how is it actually deployed in the real world.
- **OPC Security White Paper #2 – OPC Exposed:** What are the risks and vulnerabilities incurred in deploying OPC in a control environment?
- **OPC Security White Paper #3 – Hardening Guidelines for OPC Hosts:** How can a server or workstation running OPC be secured in a simple and effective manner?

All three white papers are intended to be read and understood by IT administrators and control systems technicians who have no formal background in either Windows programming or security analysis.

1.1 The Issues

In recent years, Supervisory Control and Data Acquisition (SCADA), process control and industrial manufacturing systems have increasingly relied on commercial information technologies (IT) such as Ethernet™, TCP/IP and Windows® for both critical and non-critical communications. The use of these common protocols and operating systems has made the interfacing of industrial control equipment much easier, but there is now significantly less isolation from the outside world. Unless the controls engineer takes specific steps to secure the control system, network security problems from the Enterprise Network (EN) and the world at large will be passed onto the SCADA and Process Control Network (PCN), putting industrial production and human safety at risk.

The wide-spread adoption of OLE for Process Control (OPC) standards for interfacing systems on both the plant floor and the business network is a classic example of both the benefits and risks of adopting IT technologies in the control world. OPC is an industrial standard based on the Microsoft Distributed Component Object Model (DCOM) interface of the Remote Procedure Call (RPC) service. Due to its perceived vendor-neutral position in the industrial controls market, OPC is being increasingly used to interconnect

Human Machine Interface (HMI) workstations, data historians and other servers on the control network with enterprise databases, ERP systems and other business-oriented software. Furthermore, since most vendors support OPC, it is often thought of as the one of the few universal protocols in the industrial controls world, adding to its widespread appeal.

Unfortunately, viruses and worms from the IT world may be increasingly focusing on the underlying RPC/DCOM protocols used by OPC, as noted in the following discussion of attack trends:

> *"Over the past few months, the two attack vectors that we saw in volume were against the Windows DCOM (Distributed Component Object Model) interface of the RPC (remote procedure call) service and against the Windows LSASS (Local Security Authority Subsystem Service). These seem to be the current favorites for virus and worm writers, and we expect this trend to continue."*[1]

At the same time, news of the vulnerabilities in OPC are starting to reach the mainstream press, as seen in the March 2007 eWeek article entitled "Hole Found in Protocol Handling Vital National Infrastructure"[2]. Thus, the use of OPC connectivity in control systems and servers leads to the possibility of DCOM-based protocol attacks disrupting control systems operations.

Complicating matters even more is Microsoft's goal of retiring DCOM in favor of .NET and movement towards Service Oriented Architectures. The good news is that most OPC applications will eventually be migrated from the DCOM-based architecture to a potentially more secure .NET-based architecture (See Section 5.7: OPC Unified Architecture for more details). The bad news is that Microsoft's desire to discontinue support for DCOM in the long term may require some companies to use unsupported software with serious vulnerabilities. Regardless, DCOM-based OPC is what is used on the plant floor and it will continue to be used for many years to come. Thus, this document focuses almost exclusively on OPC over DCOM.

Despite all these concerns, it is our belief that the most serious issue for OPC is that securely deploying OPC applications has proven to be a challenge for most engineers and technicians. While OPC is an open protocol with the specifications freely available, engineers must wade through a large amount of very detailed information to answer even basic security questions. There is little direct guidance on securing OPC, and our research indicates that much of what is available may actually be ineffective or misguided. All things

---

[1] Bruce Schneier, "Attack Trends" QUEUE Magazine, Association of Computing Machinery, June 2005

[2] Lisa Vaas, "Hole Found in Protocol Handling Vital National Infrastructure" eWeek, http://www.eweek.com/article2/0,1759,2107265,00.asp, March 23, 2007

considered, there is little doubt that some clear advice for the control engineer on how best to secure OPC systems would be very useful.

1.2 Organization & Methodology of the Study

While researching this study we found few treatments of OPC that were useful for readers who were not experienced software developers. Thus, we begin this first white paper with a review of the OPC specifications, focusing on details that are relevant from a security point of view and might be useful to users wishing to understand the risks of OPC deployments. Following this conceptual overview, we describe the real-world operation of OPC applications, identifying components that need to be understood to harden hosts running OPC client and server applications.

In White Paper #2 we define a set of vulnerabilities and likely threats based on OPC's architecture (such as the use of DCOM, the reliance upon an OPC Server Browser, etc.) and common mis-configuration vulnerabilities found in OPC servers.

In White Paper #3 we use all this information, plus the results of the surveys, to give the OPC end-user a series of practical recommendations they can draw on to secure their OPC host machines.

Creating these recommendations required the following four-phase approach to the study:

1. Data Gathering
    - Conducting user surveys and collecting information on OPC deployments in order to get a representative sample of how actual OPC deployments were configured in the field by our target audience.
    - Reviewing OPC Foundation and vendor configuration guidelines.
    - Conducting a literature search for OPC-related papers and guidelines.
2. Ascertaining potential threats and vulnerabilities in OPC systems
    - Identifying what operating system configuration issues exist in typical OPC deployments.
    - Identifying what OPC, RPC and DCOM issues exist in typical OPC deployments.
3. Creating recommendations for mitigating potential threats and vulnerabilities

- Determining what could be done to secure the underlying operation system without impacting the OPC functionality.
- Determining what could be done to secure RPC/DCOM components in an OPC host.
- Determining OPC-specific client and server security configurations.

4. Testing the Security Recommendations
   - Lab testing all recommendations in a typical OPC environment and modifying our recommendations accordingly.

1.3 Limitations of this Study

It is important to understand that this report is not intended to be a formal security analysis of OPC or DCOM, but instead is a set of observations and practices that will help end-users secure their OPC systems. As well, this report is focused only on securing the host computers that are running OPC. Securing the network OPC operates over is an interesting and important area of research, but is beyond the scope of this report. A follow-on study is planned to investigate these network security aspects and consider solutions for OPC/DCOM in the network infrastructure, including firewall rule-sets and analysis of third party OPC tunnelling solutions.

As well, we cannot guarantee that following our recommendations will result in a completely secure configuration. Nor can we guarantee that these recommendations will work in all situations; some modifications may be required for individual OPC client and server applications or Microsoft Windows network deployments. However, we are confident that using these guidelines will result in more secure systems as compared to the typical default application and operating system settings we have seen in our investigations.

2 What is OPC?

OLE for Process Control (OPC) is a software interface technology used to facilitate the transfer of data between industrial control systems, Human Machine Interfaces (HMI), supervisory systems and enterprise systems such as historical databases. It was developed in response to the need for a standardized method for allowing different control systems to interface with each other. Today it has grown to be the leading technology for integrating different control products.

The primary value of OPC is that it provides a common interface for communicating with diverse industrial control products, regardless of the software or hardware used in the process. Before OPC, application developers had to develop specific communications drivers for each control system they wished to interface with. For example, HMI vendors had to develop hundreds of different drivers for the different Distributed Control Systems (DCS) and Programmable Logic Controllers (PLCs) on the market.

Using OPC, these application vendors no longer need to develop separate drivers for each network or processor. Instead, they create a single optimized OPC client and/or server for their product. This OPC client would then communicate with OPC servers designed and sold by the manufacturers of the other networks and controllers.

It is important to understand that OPC does not eliminate the need for drivers. Typically each manufacturer develops an OPC server for their specific product using whatever protocol their device needs, since they are best suited to build a server that will take full advantage of their product. However, once an OPC server exists for a piece of equipment or an application, it becomes much easier to integrate its data with other OPC compliant software.

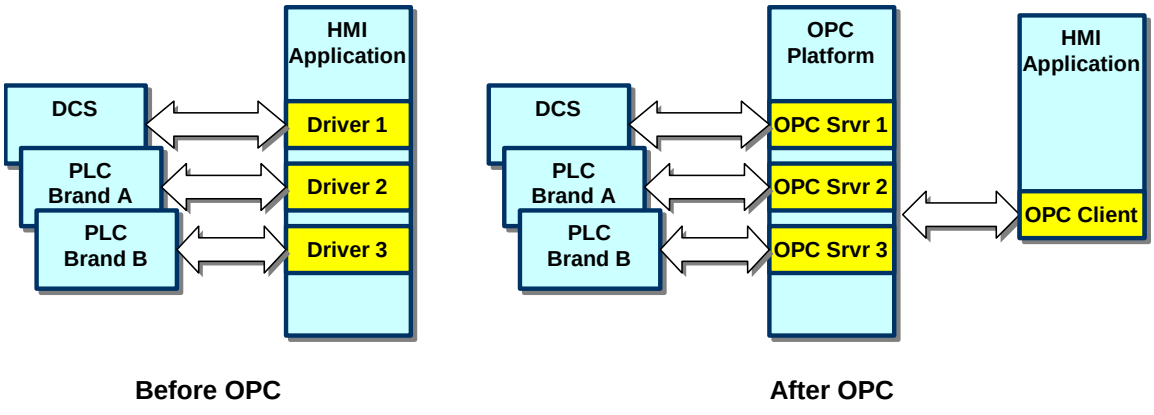

Figure 2-1: OPC Efficiency in Driver Development

OPC is based on Microsoft's Distributed Component Object Model (DCOM) technology, which is the culmination of a number of other technologies including Component Object Model (COM) and the Object Linking and Embedding (OLE). Many people have heard of OLE and have used its capabilities when adding a spreadsheet to a word processing document. OLE allows the spreadsheet application to dynamically update the information in the word processing document. Typically the user isn't required to do even the slightest configuration beyond the click of a mouse. The OLE specification defines how the spreadsheet (in this case the OLE server) will format and send data to the word processor document (the OLE client).[3]

OPC is based on a *client-server* architecture. An OPC server is a software application that typically gathers information from devices (such as PLC, DCS or SCADA controllers) using these device's native protocols (such as MODBUS or PROFIBUS). The server then provides access to this data via COM objects and method calls, allowing multiple OPC clients to indirectly read and write to the field device via the OPC server.

An OPC client is an application that accesses data held by OPC servers. For example, an HMI package may contain an OPC client that allows it to access data provided by an OPC server application resident on another machine. The HMI package could also act as an OPC server, allowing other OPC clients to access the data it has aggregated either directly from field controller or from other OPC servers.

To illustrate this client-server architecture, imagine a simple system with three basic components designed for controlling the water level in a tank:

- A MODBUS-capable PLC performing the actual control,
- An OPC platform that contains an OPC server and a MODBUS protocol driver,
- A HMI for operator access to the control system.

Within the PLC, the data registers and discrete points might look like this:

Register	Name (Tag)	Description
40001	SP	Water Level Setpoint
40002	CO	Pump Control Output
40003	PV	Water Level Sensor
10001	LoAlarm	Tank Dry Alarm
10001	HiAlarm	Tank Overflow Alarm

Table 2-1: Example Data Points in the PLC

[3] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndcom/html/msdn_dcomtec.asp

The HMI will need to be able to write the set point in the controller, read the current water level, and monitor the controlled output (the pump) and alarms. If the HMI needs to read a value from the PLC, it sends a request via an OPC Application Programming Interface (API) call and the server translates this into a MODBUS message for communications to the PLC. When the desired information is returned from the PLC to the OPC server it then translates that back to OPC for transmission to the HMI.

Figure 2-2 shows a simplified illustration of the communications in this system.

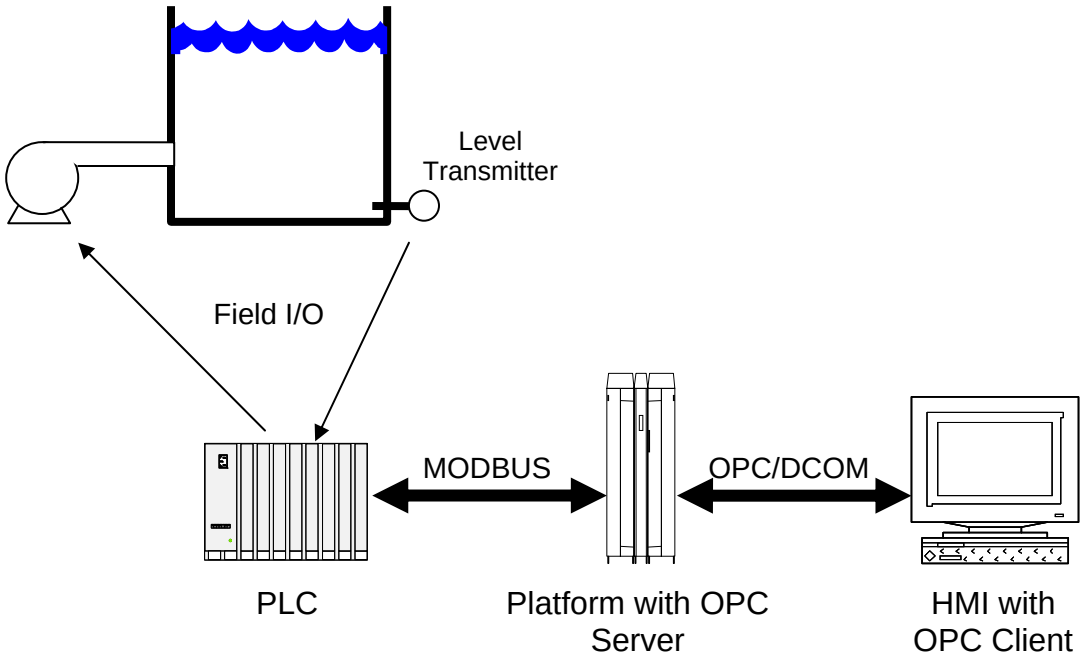

Figure 2-2: Example of Possible OPC Client-Server Architecture in Tank Level Control

3 How Industry Uses OPC

As part of the research effort in creating this series of reports, a survey of OPC end-users was conducted by the study team and the Instrumentation, Automation and Systems Society (ISA) in the Winter of 2006. The intent was to determine how end-user companies actually deploy OPC in their process and manufacturing environments. Security recommendations developed in this study were then tailored to meet the needs of the largest number of actual users, as identified by the survey.

The following two sections discuss the types of OPC configurations users reported as actually used in industry. We then illustrate three of the common configurations involving OPC, that end-user companies reported using on their sites. The intent of this section is not to recommend or endorse these deployments, but rather show how OPC is actually used in the real world.

3.1 Key Findings from the OPC Deployment Survey

The OPC User Survey was conducted in the winter of 2006 using a web survey designed by study team and managed by ISA. A total of 113 individuals responded, the vast majority of them were end users of control systems.

3.1.1 What do End-Users use OPC for?

The first question in the survey asked "*how does your company typically use OPC in its operations*"*?* Not surprisingly, OPC was always or often used for data transfer to historians, data aggregation in HMIs and supervisory control in the majority of the end users facilities. What was surprising was that 30% of the end users reported employing OPC for data sharing to 3rd parties such as business partners and suppliers. Since it is likely that most 3rd parties are located remotely from the users' production facilities, this indicates that OPC is being used for data transfer far beyond the plant floor.

	Always or Often	Sometimes	Rarely or Never
Transfer to Historians	72%	16%	12%
Data Aggregation in HMIs	50%	20%	29%
Supervisory Control	50%	17%	33%
System Control Data	40%	21%	39%
Data Between Partners	30%	11%	59%
System Interlocks	17%	13%	70%

Table 3-1: Question #1: How does your company typically use OPC in its operations?

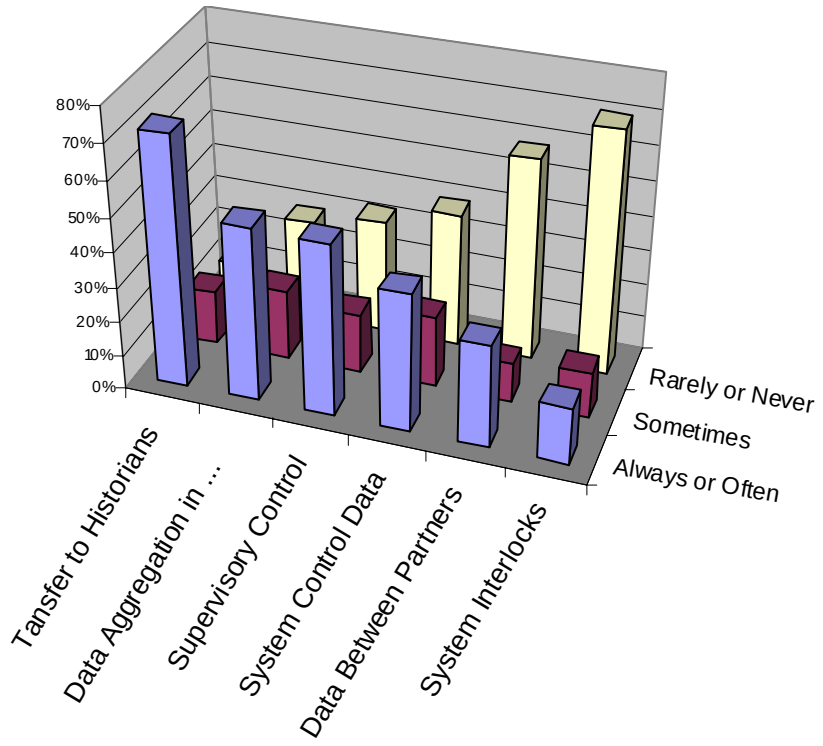

Figure 3-1: Question #1: How does your company typically use OPC in its operations?

3.1.2 What OPC Functionality do End-Users Deploy?

The next question asked the respondent to indicate what OPC functionality their company used. The results indicate that Data Access (DA), Historical Data Access (HDA) and Alarms and Events (A&E) are the primary OPC specifications that actually get used on the plant floor. The remaining specifications are only used in limited cases.

	Always or Often	Sometimes	Rarely or Never
Data Access (DA)	82%	13%	5%
Historical Data Access (HDA)	53%	15%	33%
Alarms and Events (A&E)	42%	18%	40%
Data eXchange (DX)	25%	15%	60%
XML Data Access (XML-DA)	23%	14%	63%
Web Services	19%	12%	69%
Batch	17%	18%	65%
Security	15%	15%	70%
Unified Architecture (UA)	10%	16%	74%

Table 3-2: Question #2: What OPC Functionality Does Your Company Use?

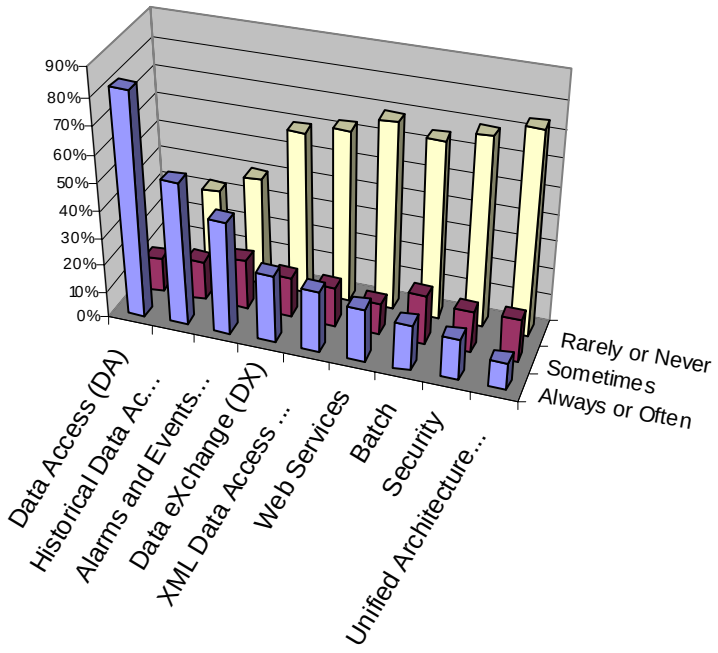

Figure 3-2: Question #2: What OPC Functionality Does Your Company Use?

3.1.3 What is the Impact if OPC Communications are Lost?

The next question asked the respondent to indicate what types of impact would the loss of OPC have on their operations and what percent of the OPC systems deployed would have this impact. What is interesting is that over a quarter of the sites reported that loss of OPC would result in a loss of production. Also interesting is that more systems would experience loss of view by the operators than not.

While some users remarked that they had deliberately structured their systems to minimize safety and operational effects on loss of OPC-based information, others stated the opposite; "*We control the motor drives by OPC with the DCS. If we lose the OPC we stop the production!"* Clearly OPC is not just being used for data management purposes on the plant floor, but instead is a critical component of many production systems. This highlights the urgent need for better OPC security.

	Most (80% or more)	Some (60 -40%)	Few (20% or less)
Temporary loss of historical data access	38%	28%	34%
Permanent loss of historical data	33%	18%	49%
Loss of view by operators	41%	23%	36%
Loss of production	27%	17%	57%
Other	16%	19%	65%

Table 3-3: Question #3: Percent of Systems with a Given Impact if OPC is Lost

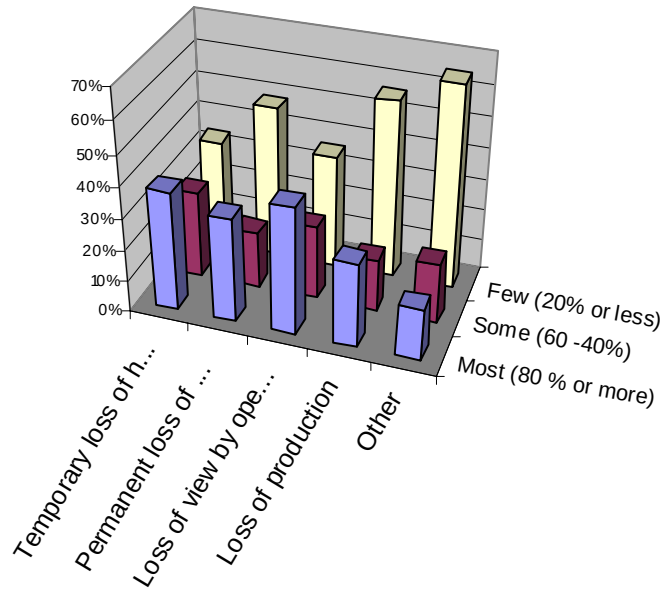

Figure 3-3: Question #3: Percent of Systems with a Given Impact if OPC is Lost

3.2 Customer Reference Implementations of OPC

The final question on OPC use was designed to determine which networks OPC traffic is actually found on. In other words, is most OPC traffic restricted to only the lowest levels of the control system or does it travel over upper levels such as the enterprise network or even the Internet. Closely corresponding to the response to question #1, OPC was used in about two-thirds of the sites for transfers in layers 1, 2 and 3 of the network (the layers refer to the ISA SP-99 General Reference Model and not the OSI model). This aligns with the response to Question #1 of the survey, which indicated that data transfer to historians, data aggregation in HMIs and supervisory control was a primary use in the majority of the end users facilities.

Also correlating with question #1 was the fact that approximately 20% of companies reported deploying OPC over the site business network, enterprise network or corporate Intranet and approximately 10% used OPC over the Internet. Clearly, the common belief that OPC is only found on the control network is badly mistaken.

	Always or Often	Sometimes	Rarely or Never
Internal to OPC Server Only (No network traffic)	43%	16%	40%
Control Network (Layer 1)	49%	15%	35%
HMI/Supervisory Network (Layer 2)	67%	17%	17%
Site Operations/DH Network (Layer 3)	62%	18%	20%
Control System DMZ	30%	11%	59%
Business Network (Layer 4)	22%	16%	62%
Enterprise Network (Layer 5)	18%	12%	70%
Corporate Intranet	22%	8%	70%
Internet via VPN	12%	8%	80%
Internet	8%	4%	88%

Table 3-4: Question #4: What Networks is Your OPC Traffic Operating Over?

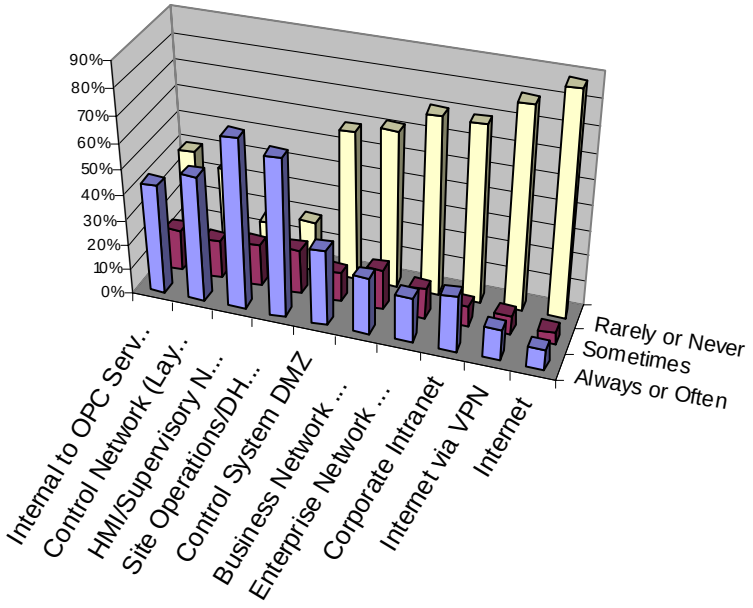

Figure 3-4: Question #4: What Networks is Your OPC Traffic Operating Over?

Using these results as a starting point, we conducted interviews with a number of end-users to understand the actual deployments that might produce these numbers. We quickly discovered three common deployments that accounted for the majority of all reported user architectures. We will describe each of these below.

3.2.1 Local OPC on Control/Supervisory Network

This first deployment is typical of how many companies use OPC for connecting control and interlock traffic between different vendors' control

systems. A vendor interface, such as RSLinx, brings up data from PLCs on the control layer into a HMI or OPC concentrator via a control protocol like Common Industrial Protocol (CIP) or Client Server Protocol (CSP). It then stores this data in the OPC server for exchange with other vendors' OPC clients and servers. All traffic is contained on the HMI layer and no OPC traffic crosses the firewall boundaries.

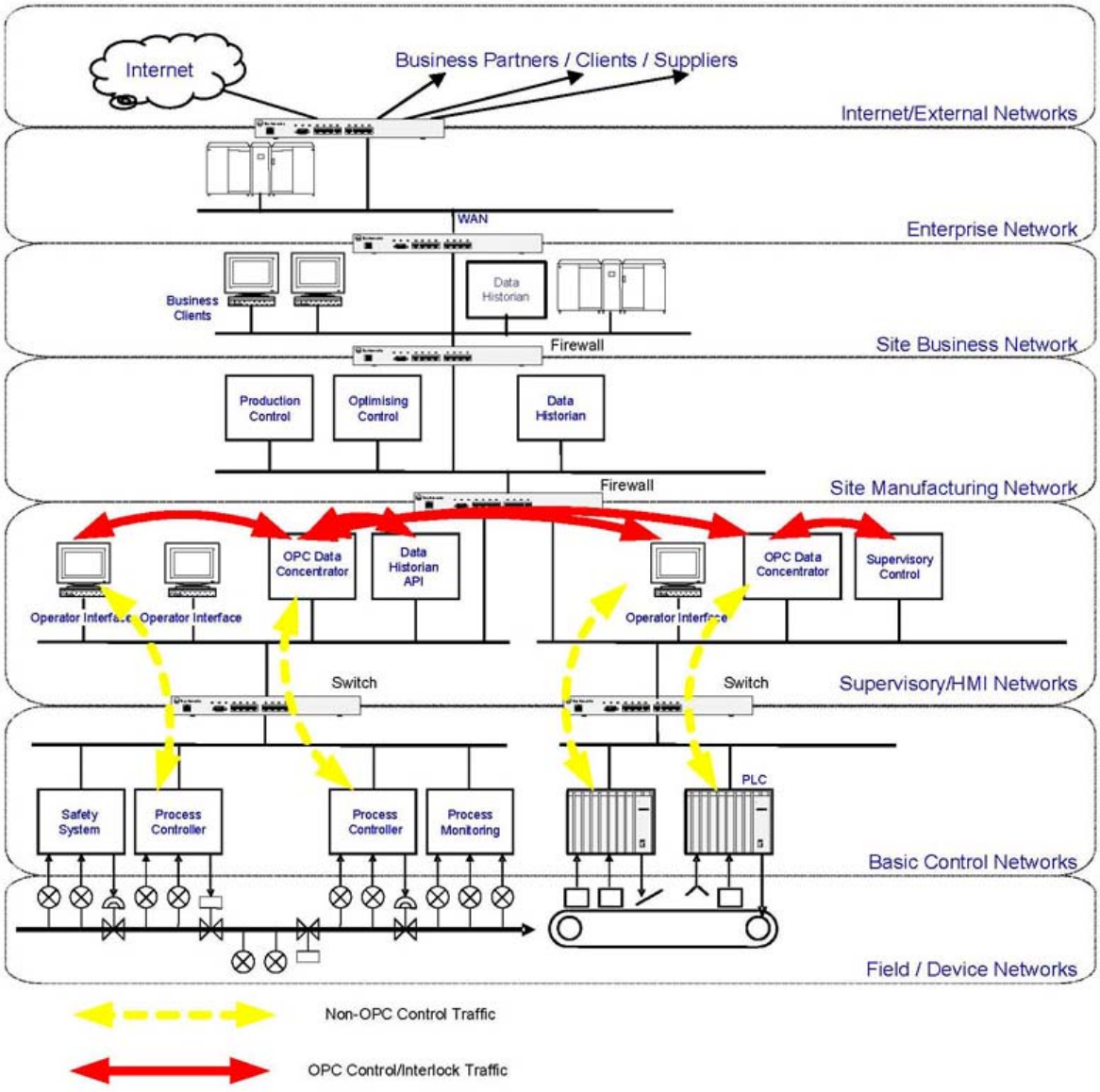

Figure 3-5: Local OPC on the Control/Supervisory Network Only

3.2.2 Local OPC on Control/Supervisory Network and Historian DMZ

The second deployment is typical of how companies use OPC to transfer both real-time and historical traffic between different vendors control systems. Again a vendor interface like RSLinx brings up data from the PLC's via a control protocol into a HMI or OPC concentrator and then stores it in

the OPC server to make it available for the data historian. Alternatively, the server may reside in the SCADA or DCS system itself and use OPC DA, HDA or A&E to transfer information. The historian computer can sit in a Demilitarized Zone (DMZ) for shared control/enterprise data servers or up on the business network, depending on the site. Typically the OPC traffic will cross at least one firewall or router with an Access Control List (ACL).

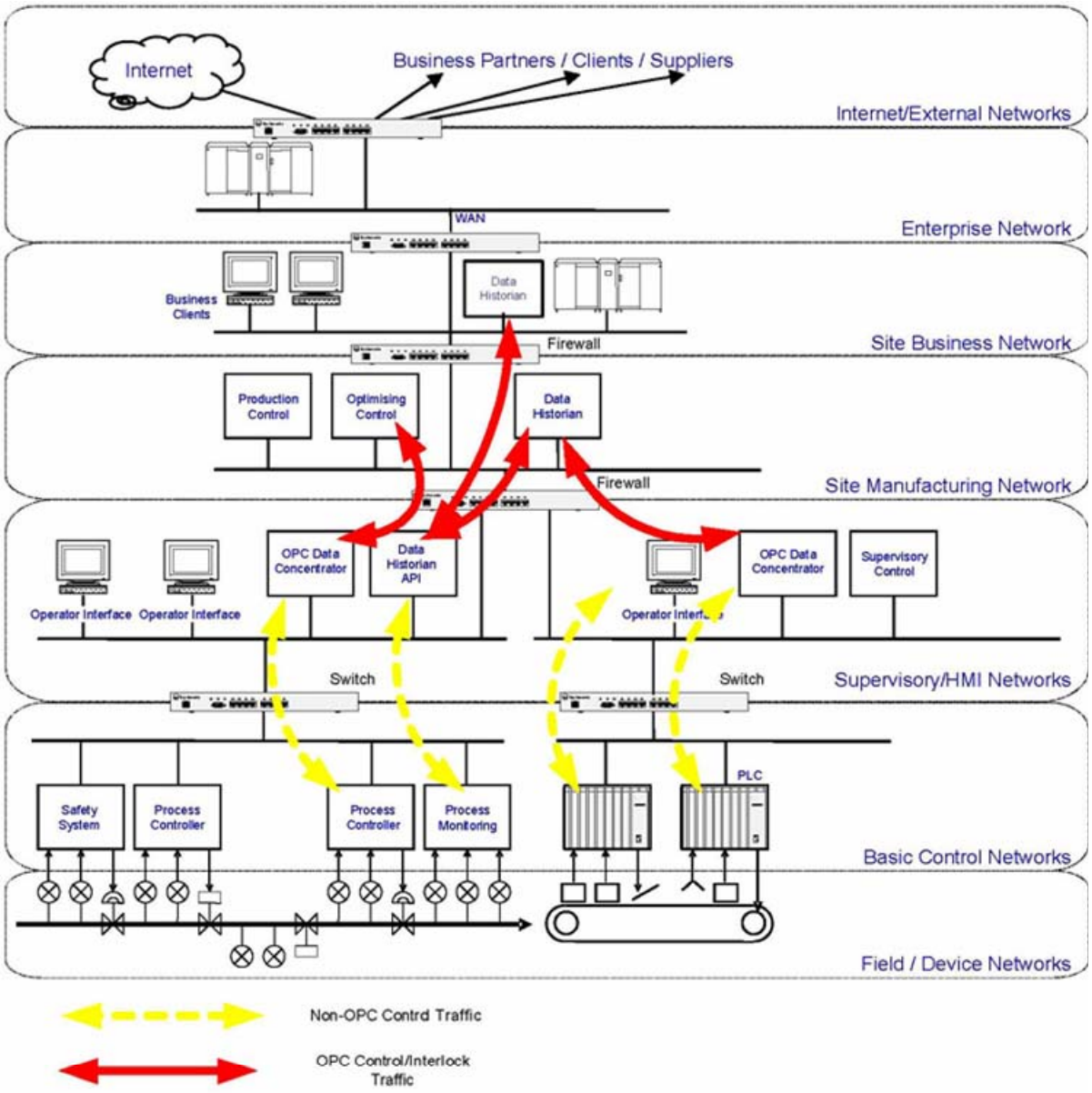

Figure 3-6: Local OPC on Control/Supervisory Network and Historian DMZ

3.2.3 Remote OPC between Plant Sites

The third deployment is typical of how some companies use OPC to aggregate data between related sites. Historical traffic between different

field stations or remote sites is transferred via OPC to a central data historian. Again this historian can sit in a DMZ or up on the business network, depending on the site. Typically the traffic will cross at least two firewall interfaces.

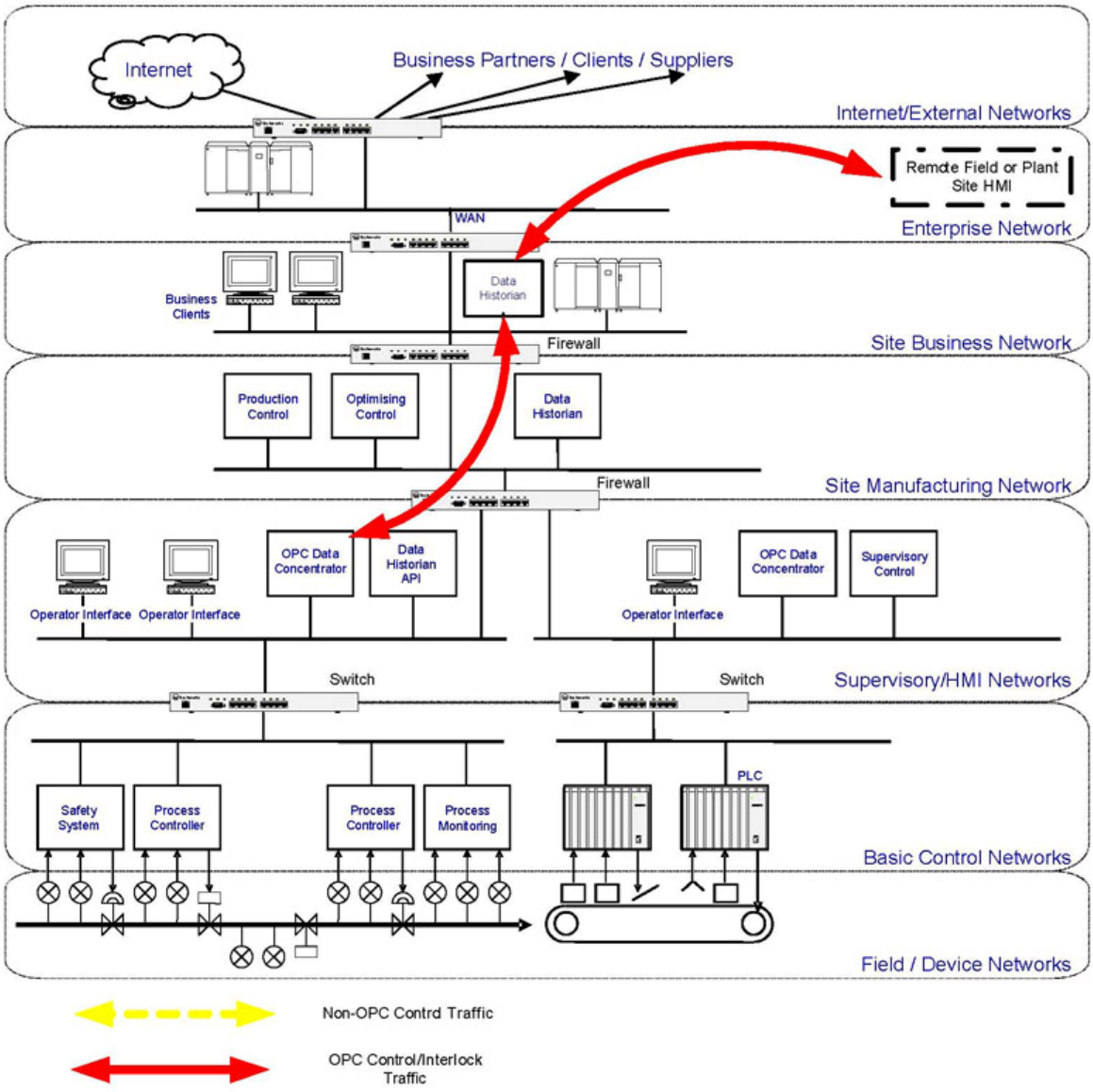

Figure 3-7: Remote OPC between Plant Sites

4 The OPC Architecture

4.1 The Relationship between OPC, COM, DCOM and RPC

One of the most important things to understand about OPC is that it is an Application Programming Interface (API) and not an "on the wire" protocol. It is at a higher level of abstraction than communications protocols such as Ethernet, TCP/IP or the even the MODBUS Application Protocol. For most developers using the OPC API, the underlying network transport or data encoding used by the API to exchange data is irrelevant.

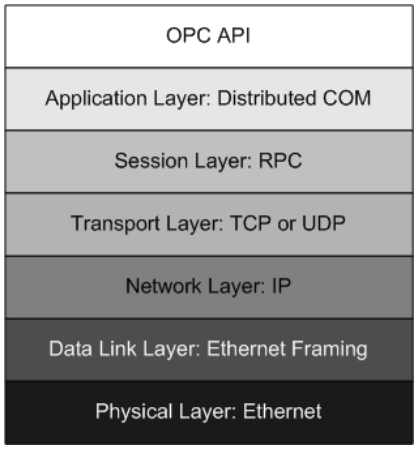

Figure 4-1: OPC Layering

As Figure 4-1 shows, underlying OPC are three very critical communications protocols; COM, DCOM and RPC.

Component Object Model (COM) is a successor to Dynamic Link Libraries (DLLs) and is a software architecture developed by Microsoft to build component-based applications. It allows programmers to encapsulate reusable pieces of code in such a way that other applications can use them without having to worry about implementation details. In this way, COM objects can be replaced by newer versions without having to rewrite the applications using them.

Distributed Component Object Model (DCOM) is a network-aware version of COM. It tries to hide the difference between invoking local (i.e. on the same computer) and remote interfaces (i.e. two different computers) from software developers. In order to do this, all the parameters must be passed by value and the returned value must also be passed by value. The process of converting the parameters to data to be transferred over the wire is called marshalling. Once marshalling is completed the data stream is serialized, transmitted and finally restored to its original data ordering on the other end of the connection.

DCOM uses the mechanism of Remote Procedure Calls (RPCs) to transparently send and receive information between COM components (i.e. clients and servers) on the same network. RPC allows system developers to control remote execution of programs without the need to develop specific procedures for the server. The client program sends a message to the server with the appropriate arguments and the server returns a message containing the results of the executed program.

4.2 OPC Data Model

The information available from the OPC server is organized into groups of related items for efficiency. Servers can contain multiple groups of items, and a group can either be:

- a public group, available for access by any client,
- a local group, only accessible by the client that created it.

In Figure 4-2 below we expand our earlier example to include two PLC's connecting to a computer running one or more OPC servers maintaining grouped information. The PLCs and the OPC servers communicate using the native PLC protocol while the OPC clients running on the other computers access the data in the OPC server via DCOM.

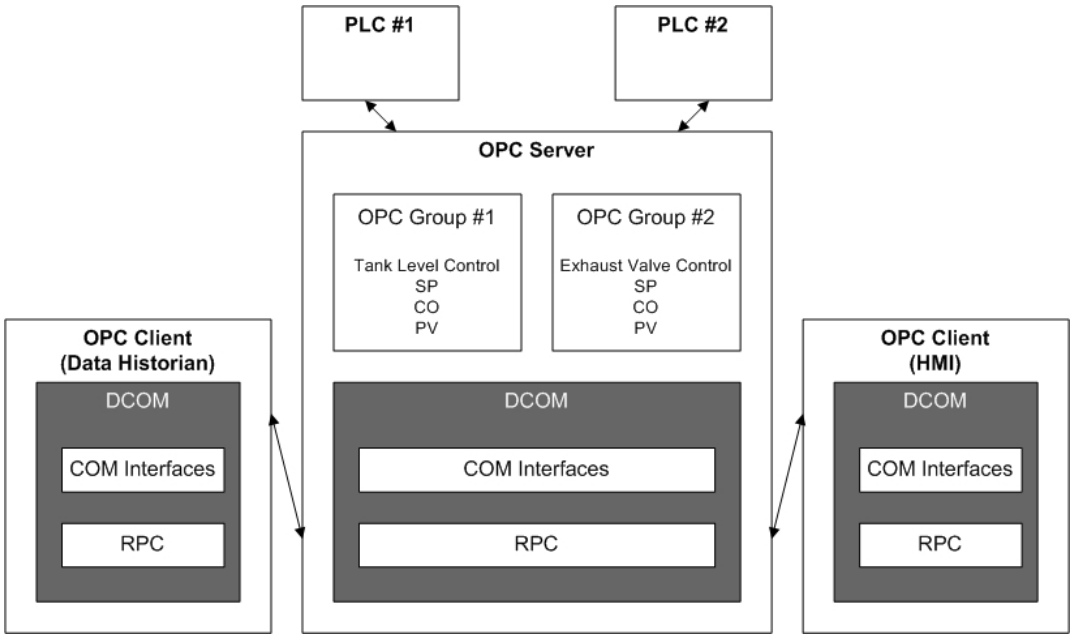

Figure 4-2: OPC Interaction

In the earlier example, where a MODBUS/TCP OPC server was connected to a MODBUS capable PLC, we might configure a "WaterLevel" group on an HMI with five members:

5. "SP" (setpoint),
6. "CO" (control output),
7. "PV" (process variable),
8. "LoAlarm" (Low Water Alarm),
9. "HiAlarm" (High Water Alarm).

The HMI could register the "WaterLevel" group with the SP, CO PV and Alarm members; then read the current values for all five items either at timed intervals or by exception (i.e. when their values changed). The HMI could also have write access to the "SP" setpoint variable.

One significant advantage of OPC is that we do not have to directly deal with the control device's internal architecture. The software can deal with named items and groups of items instead of dealing with raw register numbers and data types. This also allows for an easier job adding or changing control systems, such as when migrating from a proprietary protocol to an Ethernet-based protocol, without altering the client applications.

5 OPC Standards & APIs

One of the challenges faced by end-users attempting to secure their OPC deployments is the lack of useful information on the OPC API's which is relevant and particularly useful for non-developers to answer risk-related questions. To address this problem, this section provides an overview of the most important OPC specifications.

For each OPC specification, we discuss typical uses and key functionality that is provided by the specification. We also define important differences among the specifications, and provide a short overview of the namespace and object hierarchy to illustrate the type of data that is exchanged between OPC clients and servers. The number in parentheses indicates the version of the OPC specification that was reviewed.

5.1 OPC Data Access (3.0)

OPC Data Access (OPC-DA) is the oldest of the OPC specifications, originally released in 1998 as the "OPC Specification." As the name implies, OPC-DA is primarily used to provide real-time access to process control and manufacturing data in a single format, regardless of its origin. An OPC-DA server may allow access to the current values of PLC registers, DCS data points, and readings from a variety of I/O sources or other software applications.

OPC-DA provides access to the most current value for a given data point. The data itself may be cached locally within the OPC server or retrieved on request from another application or device.

Each data element is called a point and has three attributes:

- *Value* - the actual data being read or written.
- *Quality* - defines how trustworthy the data is (good, bad, uncertain) and more detailed information on the status of the data, for example, based on the link to the I/O device.
- *Time Stamps* - in some cases the device's protocol may provide this attribute for each value. If it does not, the OPC server will assign the time value based on its internal time clock.[4]

According to most industry users, OPC-DA offers good performance and offers robust communications once it is configured, hence its widespread popularity. It also supports the transfer of double precision real numbers

[4] Randy Kondor, "Understanding OPC: Basics For New Users", Industrial Ethernet Book, Issue 19:44, http://ethernet.industrial-networking.com/opc/articledisplay.asp?id=32

which many process control protocols can not do without difficult workarounds.

5.1.1 Example of OPC-DA Usage

Continuing the water level control example from above, an HMI (the OPC Client) would be able to read the current water level and control output in the PLC via the OPC server. In addition, it would be able to control the set point in the PLC via the OPC server. Generally the HMI would also monitor the quality attribute of all data and would indicate if it lost contact with the OPC server by some method such as graying, zeroing, or flashing the stale values in its user interface.

5.2 OPC Alarms & Events (1.10)

The OPC Alarms & Events (OPC A&E) specification defines an interface for alarm monitoring and acknowledgment. Unlike DA, A&E does not provide a continuous stream of data between client and server, but instead supplies a value only when a specific event occurs. These values include process alarms, operator actions, informational messages, and tracking/auditing messages. Several types of OPC A&E clients are defined in the specification:

- Operator stations
- Event/alarm logging components
- Event/alarm management subsystems

OPC A&E servers may be directly connected to the data sources (i.e. communication devices or local applications) or to local or external OPC-DA servers. The OPC A&E server may evaluate input from single or multiple data sources to determine whether an event (such as a device failure) has occurred and may report these events to one or more clients. OPC clients are normally "notified" of alarm conditions and irregular events using a technique known as "*callbacks*". These are mechanisms for the server to send messages to the client with information that the client has previously registered an interest in receiving. This is similar to unsolicited messages used in other SCADA protocols such as Distributed Network Protocol 3 (DNP3).

5.2.1 Example of OPC A&E Usage

Returning to the water tank control example, if you want to be notified only when a tank level reaches a high alarm limit you would use OPC A&E to capture the event. In contrast, OPC-DA would allow you to sample the tank level at a fixed interval (such as once per minute), but communications would not be particularly affected by any alarm or event in the process.

5.3 OPC Historical Data Access (1.20)

One of the limitations of OPC-DA is that it only provides visibility into a relatively short window of time, making it difficult to perform data visualization, trending, fault prediction or root cause analysis. OPC Historical Data Access (OPC-HDA) overcomes this problem by providing a flexible means of accessing two kinds of data, namely raw and aggregate process control data[5]. Raw data is a collection of individual samples of data stored in the server database, while aggregate data sources are summary values such as minimum, maximum, difference, or average over a period of time.

OPC-HDA does not specify (or limit) the type of data that may be accessed. Servers are also available for relational databases such as Oracle and Microsoft SQL server. Just as OPC-DA attempts to provide a common open interface to a number of different data sources, OPC-HDA does the same for historians, many of which have proprietary interfaces.

5.3.1 Example of OPC-HDA Usage

Continuing with the water tank example, an OPC HDA application could be used to log the analog values in the system on a continuous basis at the OPC server, allowing for later review of the data. This would then provide the HMI to access a historical record of the water levels, control outputs and set points.

5.4 OPC Data Exchange (1.0)

OPC Data Exchange (DX) defines an industry-standard set of interfaces that provide interoperable data exchange and server-to-server communications between devices and controllers connected to Ethernet networks using different protocols. It is an extension of the existing OPC data access specification, providing an application independent interface suitable for both factory and process automation[6].

OPC-DX allows OPC-DA servers to directly exchange data without the requirement of an intermediate OPC Client. Functionally, OPC-DX servers implement many of the features of DA servers, but provide more reliable delivery mechanisms. The best way to think of an OPC-DX server is as an OPC-DA server that can be configured to exchange data with other OPC-DA servers. As is the case with other OPC servers, a client is still used to configure, control, and monitor this data exchange.

---

[5] Iwanitz, Franz and Lange, Juergen; OPC: Fundamentals, Implementation and Application (2nd Edition) Huethig Verlag, Heidelberg, 2002 (p 58)

[6] Don Holley; OPC-DX Glues Fieldbuses Together, Industrial Ethernet Book, Issue 8:34, http://ethernet.industrial-networking.com/opc/articledisplay.asp?id=24

Based on our survey research, it is unclear whether OPC-DX has been widely adopted within the industry. In terms of implementation, many vendors have separate data bridge software for exchanging data between different packages. In most cases, OPC-DX seems to be implemented as a plug-in for an OPC server that allows users to easily mirror values to another OPC server.

5.4.1 Example of OPC-DX Usage

OPC-DX could be used to solve the interoperability problems when two dissimilar real time communications protocols are being used in a system. For example, perhaps a second water tank is controlled by a PLC that used Ethernet/IP rather than MODBUS/TCP. OPC-DA would require an intervening OPC client with some sort of "reflector" software to allow the two OPC servers to communicate, while OPC-DX would allow the two OPC servers to communicate directly.

5.5 OPC Security (1.0)

The OPC Security Specification is designed to ensure that OPC Servers implement operating system security APIs in a consistent manner. It defines three security levels:

1. *Disabled Security* – no security is enabled.

2. *DCOM Security* - use of DCOM settings to determine launch and access permissions as well as message privacy and integrity.

3. *OPC Security* - OPC server acts as a reference monitor to access vendor specific objects that are exposed by the OPC Server.

The specification document provides some background information defining common security concepts (such as authentication, authorization, reference, etc.) as well as some low-level information on how to implement the DCOM security API. However, this specification provides little useful information for end-users and provides minimal information on OPC threats and vulnerabilities, nor does it include host or network security configuration practices for OPC. Based on our research, it is unclear how many vendors or users actually implement the security standard. Web research showed that only Northern Dynamics, Yokogawa, Unicorn, and Novatek claim to support the standard.

5.6 OPC XML-Data Access (1.01)

EXtensible Markup Language (XML) Web Services are becoming the standard method for exchanging data between enterprise applications and are increasingly found in process control environments. OPC XML-DA was released in 2003 after several years of development, and provides a Simple

Object Application Protocol (SOAP) interface to OPC DA 2.0/3.0 objects. This allows client applications to be written in Java, Perl, Python, and other languages that support SOAP.

SOAP and XML Web Services use HyperText Transfer Protocol (HTTP) and HyperText Transfer Protocol over Secure Socket Layer (HTTPS) as their underlying transport mechanisms and provide a platform neutral architecture that is more suitable for Internet-based traffic, as compared to technologies such as DCOM or Common Object Request Broker Architecture (CORBA). However, due to possible performance limitations, OPC XML-DA is unlikely to be used for real time applications, although it is commonly used as a bridge between the enterprise and control network. Furthermore, only OPC-DA functionality is provided in XML-DA, so it can be best seen as a transitional path to a true Web Services architecture that is currently under development with OPC-UA (Unified Architecture) project.

5.6.1 Example of OPC XML-DA Usage

OPC XML DA can be implemented on any device supporting HTTP and XML, allowing limited OPC communications to non-OPC aware system. Continuing the water tank example, OPC-XML data access would allow clients that could formulate SOAP XML requests over HTTP to retrieve information about the water tank level. These would typically be enterprise software applications that do not typically support OPC

5.7 OPC Unified Architecture (RC 1.00)

OPC Unified Architecture (OPC-UA) reflects Microsoft's goal of retiring DCOM in favor of .NET and movement towards Service Oriented Architectures. The UA specification will ultimately consist of 13 parts, several of which have not been released publicly (even in draft form) at the time of writing. As the title suggests, OPC-UA integrates the functionality of previous specifications (OPC-DA, OPC-HDA, OPC A&E, OPC-DX, etc.) into a single integrated namespace. This will address the API differences in the current specifications that were developed independently of each other.

OPC-UA abandons COM/DCOM in favor of two different transports: SOAP/HTTP(S) and a binary message encoding scheme that operates direct communications on top of TCP. Due to the known performance limitations of XML, the binary message encoding scheme was provided to allow high performance data exchange, especially on embedded devices that maintain real-time communications.

It is premature to assess the security of OPC-UA relative to DCOM-based OPC, since the OPC-UA security APIs are still under development. However, since there is now a much greater awareness in the OPC Foundation, the OPC vendors, and Microsoft for the need for security, there is little question

that .NET will provide a more secure foundation than COM/DCOM. It will also make development of OPC Clients and Servers on non-Microsoft platforms much easier. OPC-UA also eliminates DCOM firewall issues, but this does not eliminate all security concerns. OPC-UA will expose a different attack surface to an equally hostile set of threats in an area where active vulnerability research is ongoing, with new web application threats and vulnerability discovered on a daily basis.

Only time will tell whether vendors can implement OPC web services securely, and whether end-users can harden the application infrastructure. Although vendors will most likely use existing XML stacks, the binary encoding routines are of particular concern, especially in embedded controllers, which have been especially prone to parsing errors in the past.

6 OPC Internals - Relevant OPC and Windows Components

The previous material in this white paper provided an overview of OPC, but to fully understand and address the security problems end-users may face in OPC deployments, more detail is needed. In this section we go beyond the high-level concepts presented in the specifications, and analyze the applications, processes, and system components running behind the scenes.

To begin with, it is important to understand that OPC Servers are not monolithic applications, but are made up of a number of related software components. Some of these components are part of the Windows operating system, while others were developed and released by the OPC Foundation. Still others are server applications developed by the OPC vendors. Finally, custom OPC applications may be developed by end-users using programming languages such as Visual Basic.

In most cases an OPC Server consists of a number of separate components:

- A service that is the actual “Engine”.
- A Graphical User Interface (GUI) for interacting with and configuring the “Engine”.
- DLL's implementing the code called by the “Engine”.
- One or more drivers for communicating to the control device over a non-OPC protocol (such as MODBUS); these may be implemented as a DLL or as a separate service.

In addition to server or client specific components, there are a number of general purpose pieces of software that we will describe below.

6.1 Proxy/Stub DLLs

The OPC Foundation provides a set of Dynamic Link Libraries (DLL) that define the client and server OPC interfaces. These components marshal and un-marshal interface pointers and the method parameters. The “proxy” is the client-side code, while the "stub" is the server side marshalling code that interacts with the OPC server code developed the server developer. Both the proxy and the stub are generated from the Interface Definition Language (IDL) within the OPC Standard.

In the past, vendors distributed their own versions of these files, but this led to application incompatibility and version management issues. To solve this problem, the OPC Foundation chose to distribute a single “approved”

version of these DLLs. Today, all vendors must include these components with their OPC servers. If a security bug were discovered in one of these DLL's, it would affect all OPC implementations, and the OPC Foundation would have to issue new versions of the proxy stub libraries to patch the vulnerability.

6.2 OPC Server Browser

The OPC Server Browser (typically implemented by the OPCEnum.exe executable), is a DCOM component that is used by the client software to retrieve information about OPC server applications that may be active on a given host. This component exposes interfaces that allow clients to query the Component Category Manager (CCM) in order to find out what OPC servers are available. The OPC Server Browser allows remote clients to determine which OPC Servers are available without having to directly browse the host's registry, as was done in early OPC servers. The OPC Server Browser listens on an arbitrary TCP port located above 1024. It is also referred to as the "OPC Discovery Service Executable."

6.3 Windows Components

Given OPC's reliance on DCOM, it should come as no surprise that OPC applications heavily rely on a number of Microsoft components for configuration and operation. Like most Windows applications, OPC and DCOM make extensive use of the Windows registry. When an OPC server is installed, it often adds entries to the Windows registry. Here are some typical entries:

1. *Program Identifier (ProgID)* - This string is defined as "Manufacturer.ServerName" for OPC servers. By browsing the registry it is possible to see which OPC applications are installed on a given system. The program identifier entry must also contain the sub-keys "OPC" and "Class Identifier (CLSID)."

2. *Class Identifier (CLSID)* - This is a standard 128-bit Globally Unique Identifier (GUID) which identifies all COM objects. Further Category Identifiers (CATID) are stored as sub-keys within the CLSID and define which of the OPC Specifications are active. Each OPC vendor has a unique CLSID and a unique value for each application

As noted earlier, older versions of OPC clients (prior to the inclusion of the OPC Server Browser) browsed the registry directly to identify available server applications. In some cases this required registry settings to be modified on the client host to be able to identify the server applications.

Based on in-lab observations of a number of OPC systems, it turned out that OPC Servers and Clients require a surprisingly small set of Windows system services for operation. These include:

- *OpcEnum* - OPC requires this service to be running so remote clients can determine which OPC Servers are running on a host.
- *Remote Procedure Call* - required by OpcEnum.
- *Server Process* – OPC servers are typically started as a service but a GUI client is used to configure and control the process.

Of course the underlying network-related services (such as IPSEC Services and DNS Services) are typically needed along with the RPC and COM+ services. As well, we found that the "Plug and Play" Windows service must be enabled for many OPC applications to perform reliably. A more detailed list for system configuration purposes is supplied in White Paper 2.

6.4 A Simple OPC Server

To help understand how these components fit together and how they impact the configuration of a host we selected a simple OPC server called DSxP[7] as an example. This software provides a simulation of a "real-world" OPC server that clients can connect to for testing purposes. Installing this small server does the following to a Windows host:

1. Creates two directories:
   - C:\Program Files\DSxP
   - C:\Program Files\DSxP\DSxPOpcSimulator
2. Places four files into the main directory of the server:
   - C:\Program Files\DSxP\DSxPOpcSimulator\DSxPOpcSimulator.exe
   - C:\Program Files\DSxP\DSxPOpcSimulator\opcdata.xml
   - C:\Program Files\DSxP\DSxPOpcSimulator\unins000.exe
   - C:\Program Files\DSxP\DSxPOpcSimulator\unins000.dat
3. Sets up the Start menu:
   - C:\Documents and Settings\All Users\Start Menu\Programs\DSx
   - C:\Documents and Settings\All Users\Start Menu\Programs\DSxP\DSxPOpcSimulator.lnk
   - C:\Documents and Settings\All Users\Start Menu\Programs\DSxP\DSxPOpcSimulator.pif

[7] http://www.dsxp.com

4. Places a link on the desktop, with a program information file:

    - C:\Documents and Settings\opcadmin\Desktop\DSxPOpcSimulator.lnk
    - C:\Documents and Settings\opcadmin\Desktop\DSxPOpcSimulator.pif

5. Creates a registry entry:

    - HKEY_LOCAL_MACHINE\SOFTWARE\CLASSES\DSXPOpcSimulator.TSxOpcSimulator.1Software\Microsoft\Windows\CurrentVersion\Uninstall\DSxPOpcSimulator_is1

This simple example illustrates some of the changes that running an OPC service can have on a Windows host device. Typically, the more full-function OPC applications would invoke two or three times the number of entries or changes noted here.

7 Conclusions

As we have discussed in the white paper, OPC is a software interface technology designed to facilitate the transfer of data between industrial control systems, HMIs, supervisory systems and enterprise systems such as historical databases. It was developed in response to the need for a standardized method for allowing different control systems to interface with each other. Today it has grown to be the leading technology for integrating different control products.

Despite claims to the contrary, OPC is not just used for data management purposes on the plant floor, but instead is a critical component of many production systems. Over a quarter of the end-users reported that loss of OPC communications would result in a loss of production. In addition, approximately 20% of the companies reported deploying OPC over the site business network, enterprise network or corporate Intranet and approximately 10% used OPC over the Internet. This highlights the urgent need for better OPC security guidance.

The challenges of securing OPC deployments are clear. The inherent architectural complexity of OPC, the default security posture of many OPC servers, and the lack of unambiguous guidance with regard to security all contribute to the difficulties of securing OPC deployments. As well, OPC's reliance upon the Microsoft platform is both a curse and a blessing - while Windows has flaws, there are a wealth of practices for hardening Windows servers that can be applied to OPC clients and servers. We will discuss these solutions in the white papers #2 and #3.

Glossary

ACL - Access Control List: List of rules specifying access privileges to network resources.

API - Application Programming Interface: The specification of the interface an application must invoke to use certain system features.

CATID - Category Identifier: Specifies the active OPC specifications.

CCM - Component Category Manager: A utility that creates categories, places components in specified categories, and retrieves information about categories.

CERN - Conseil Europeen Recherche Nucleaire: European Laboratory for Particle Physics.

CIFS - Common Internet File System: Updated version of SMB.

CIP - Common Industrial Protocol: CIP is an open standard for industrial network technologies. It is supported by an organization called Open DeviceNet Vendor Association (ODVA).

COM – Component Object Model: Microsoft's architecture for software components. It is used for interprocess and interapplication communications. It lets components built by different vendors be combined in an application.

CLSID - Class Identifier: An identifier for COM objects.

CORBA - Common Object Request Broker Architecture: Architecture that enables objects, to communicate with one another regardless of the programming language and operating system being used.

CSP - Client Server Protocol: An Allen-Bradley protocol used to communicate to PLCs over TCP/IP.

DDE – Dynamic Data Exchange: A mechanism to exchange data on a Microsoft Windows system.

DCOM – Distributed Component Object Model: This is an extension to the Component Object Model that Microsoft made to support communication among objects on difference computers across a network.

DCS – Distributed Control System: A Distributed Control System allows for remote human monitoring and control of field devices from one or more operation centers.

DDE - Dynamic Data Exchange: An interprocess communication system built into Windows systems. DDE enables two running applications to share the same data.

DLL - Dynamic Link Libraries: A file containing executable code and data bound to a program at load time or run time, rather than during linking.

DMZ - Demilitarized Zone: A small network inserted as a "neutral zone" between a trusted private network and the outside untrusted network.

DNP3 - Distributed Network Protocol 3: A protocol used between components in process automation systems.

DNS – Domain Name System: A distributed database system for resolving human readable names to Internet Protocol addresses.

EN - Enterprise Network: A private communication network of a firm.

ERP - Enterprise Resource Planning: Set of activities a business uses to manage it's key resources.

GUI - Graphical User Interface: Graphical, as opposed to textual, interface to a computer.

GUID - Globally Unique Identifier: A unique 128-bit number that is produced by the Windows operating system and applications to identify a particular component, application, file, database entry or user.

HMI - Human Machine Interface: This interface enables the interaction of man and machine.

HTML - Hypertext Markup Language: The authoring software language used on the Internet's World Wide Web.

HTTP - HyperText Transfer Protocol: The protocol used to transfer Web documents from a server to a browser.

HTTPS - HyperText Transfer Protocol over SSL: A secure protocol used to transfer Web documents from a server to a browser.

IIS - Internet Information Server: Microsoft's web server.

IDL - Interface Definition Language: Language for describing the interface of a software component.

IDS - Intrusion Detection System: A system to detect suspicious patterns of network traffic.

IPX - Internetwork Packet Exchange: A networking protocol used by the Novell Incorporated.

IPSEC – Internet Protocol SECurity: An Internet standard providing security at the network layer.

IP - Internet Protocol: The standard protocol used on the Internet that defines the datagram format and a best effort packet delivery service.

I/O - Input/Output: An interface for the input and output of information.

ISA - Instrumentation, Automation and Systems Society: ISA is a nonprofit organization that helps automation and control professionals to solve technical instrumentation problems.

IT - Information Technology: The development, installation and implementation of applications on computer systems.

LAN - Local Area Network: A computer network that covers a small area.

LM - LAN MANager: An old Microsoft Windows authentication protocol.

LDAP - Lightweight Directory Access Protocol: Protocol to access directory services.

MBSA - Microsoft Baseline Security Analyzer: A tool from Microsoft used to test a system to see if Microsoft best practices are being used.

MIB - Management Information Base: The database that a system running an SNMP agent maintains.

MODBUS A communications protocol designed by Modicon Incorporated for use with its PLCs.

NETBEUI - NetBIOS Extended User Interface: An enhanced version of the NetBIOS protocol.

NetBIOS - Network Basic Input Output System: A de facto IBM standard for applications to use to communicate over a LAN.

NTLM – New Technology LAN Manager: A challenge - response authentication protocol that was the default for network authentication for Microsoft Windows New Technology (NT) operating systems.

OLE – Object Linking and Embedding: A precursor to COM, allowing applications to share data and manipulate shared data.

OPC – OLE for Process Control: A standard based on OLE, COM and DCOM, for accessing process control information on Microsoft Windows systems.

OPC-A&E - OPC Alarms & Events: Standards created by the OPC Foundation for alarm monitoring and acknowledgement.

OPC-DA - OPC Data Access OPC-DA: Standards created by the OPC Foundation for accessing real time data from data acquisition devices such as PLCs.

OPC-DX - OPC Data Exchange: Standards created by the OPC Foundation to allow OPC-DA servers to exchange data without using an OPC client.

OPC-HDA - OPC Historical Data Access: Standards created by the OPC Foundation for communicating data from devices and applications that provide historical data.

OPC-UA - OPC Unified Architecture: A standard being created by the OPC Foundation to tie together all existing OPC technology using the .NET Architecture.

OPC XML-DA - OPC XML Data Access: Standards created by the OPC Foundation for accessing real time data, carried in XML messages, from data acquisition devices such as PLCs.

OPCENUM – OPC ENUMerator: A service for enumerating OPC servers.

PLC – **Programmable Logic Controller**: A PLC is a small dedicated computer used for controlling industrial machinery and processes.

PCN - Process Control Network: A communications network used to transmit instructions and data to control devices and other industrial equipment.

PROGID - Program Identifier: A string that identifies the manufacturer of an OPC server and the name of the server.

RPC – Remote Procedure Call: A standard for invoking code residing on another computer across a network.

RSLinx Software providing plant floor device connectivity for a wide variety of applications.

SCADA – Supervisory Control And Data Acquisition: A system for industrial control consisting of multiple Remote Terminal Units (RTUs), a communications infrastructure, and one or more Control Computers.

SID – Security Identifier: A unique name that is used to identify a Microsoft Windows object.

SP - Service Pack: A bundle of software updates.

SPX - Sequenced Packet Exchange: A transport Layer protocol used by Novell Incorporated.

SMB - Server Message Block: A Microsoft network application-level protocol used between nodes on a LAN.

SNMP - Simple Network Management Protocol: A protocol used to manage devices such as routers, switches and hosts.

SOAP - Simple Object Access Protocol: A protocol for exchanging XML-based messages using HTTP.

SSL - Secure Socket Layer: A de facto standard for secure communications created by Netscape Incorporated.

TCP - Transmission Control Protocol: The standard transport level protocol that provides a reliable stream service.

UDP - User Datagram Protocol: Connectionless network transport protocol.

URL - Uniform Resource Locator: The address of a resource on the Internet.

WS-Security - Web Services Security: A communications protocol providing a means for applying security to Web Services.

XML - eXtensible Markup Language: A general-purpose markup language for creating special purpose markup languages that are capable of describing many different kinds of data.