OPC Overview

OPC (**O**LE for **P**rocess **C**ontrol)
is a series of standards specifications. The first standard (originally called simply the OPC Specification and now called the Data Access Specification) resulted from the collaboration of a number of leading worldwide automation suppliers working in cooperation with Microsoft. Originally based on Microsoft's OLE COM (component object model) and DCOM (distributed component object model) technologies, the specification defined a standard set of objects, interfaces and methods for use in process control and manufacturing automation applications to facilitate interoperability. The COM/DCOM technologies provided the framework for software products to be developed. There are now hundreds of OPC Data Access servers and clients available.

Adding the OPC specification to Microsoft's OLE technology in Windows allowed standardization. Now the industrial devices' manufacturers could write the OPC DA Servers and the software (like Human Machine Interfaces HMI) could become OPC Clients.

The benefit to the software suppliers was the ability to reduce their expenditures for connectivity and focus them on the core features of the software. For the users, the benefit was flexibility. They don't have to create and pay for a custom interface. OPC interface products are built once and reused many times, therefore, they undergo continuous quality control and improvement.

The user's project cycle is shorter using standardized software components. And their cost is lower. These benefits are real and tangible. Because the OPC standards are based in turn upon computer industry standards, technical reliability is assured.

The original specification standardized the acquisition of process data. It was quickly realized that communicating other types of data could benefit from standardization. Standards for Alarms & Events, Historical Data, and Batch data were launched.

Current and emerging OPC Specifications include:

|

Specification	Description
OPC Data Access	The originals! Used to move real-time data from PLCs, DCSs, and other control devices to HMIs and other display clients. The Data Access 3 specification is now a Release Candidate. It leverages earlier versions while improving the browsing capabilities and incorporating XML-DA Schema.
OPC Alarms & Events	Provides alarm and event notifications on demand (in contrast to the continuous data flow of Data Access). These include process alarms, operator actions, informational messages, and tracking/auditing messages.
OPC Batch	This specification carries the OPC philosophy to the specialized needs of batch processes. It provides interfaces for the exchange of equipment capabilities (corresponding to the S88.01 Physical Model) and current operating conditions.
OPC Data eXchange	This specification takes us from client/server to server-to-server with communication across Ethernet fieldbus networks. This provides multi-vendor interoperability! And adds remote configuration, diagnostic and monitoring/management services.
OPC Historical Data Access	Where OPC Data Access provides access to real-time, continually changing data, OPC Historical Data Access provides access to data already stored. From a simple serial data logging system to a complex SCADA system, historical archives can be retrieved in a uniform manner.
OPC Security	All the OPC servers provide information that is valuable to the enterprise and if improperly updated, could have significant consequences to plant processes. OPC Security specifies how to control client access to these servers in order to protect this sensitive information and to guard against unauthorized modification of process parameters.
OPC XML-DA	Provides flexible, consistent rules and formats for exposing plant floor data using XML, leveraging the work done by Microsoft and others on SOAP and Web Services.
OPC Complex Data	A companion specification to Data Access and XML-DA that allows servers to expose and describe more complicated data types such as binary structures and XML documents.
OPC Commands	A Working Group has been formed to develop a new set of interfaces that allow OPC clients and servers to identify, send and monitor control commands which execute on a device.

Distributed COM

The DCOM Architecture

DCOM is an extension of the Component Object Model (COM). COM defines how components and their clients interact. This interaction is defined such that the client and the component can connect without the need of any intermediary system component. The client calls methods in the component without any overhead whatsoever.

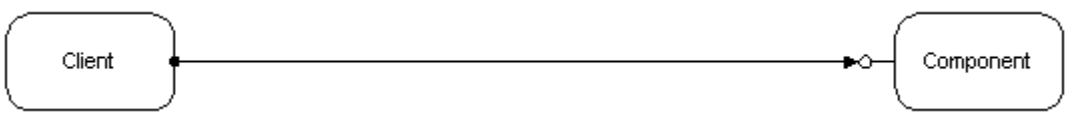

Figure 1: COM components in the same process

In today's operating systems, processes are shielded from each other. A client that needs to communicate with a component in another process cannot call the component directly, but has to use some form of interprocess communication provided by the operating system. COM provides this communication in a completely transparent fashion: it intercepts calls from the client and forwards them to the component in another process.

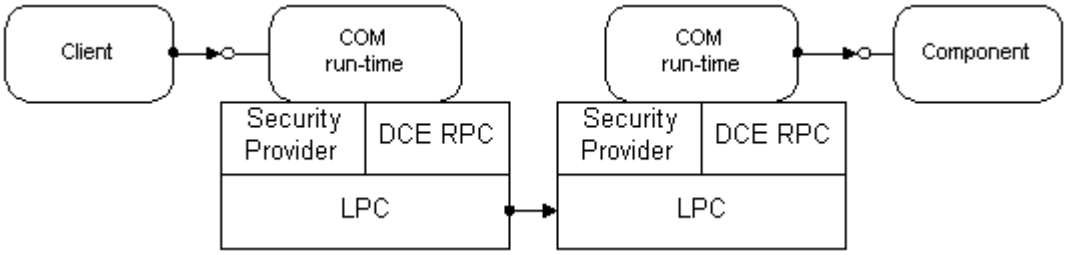

Figure 2: COM components in different processes

When client and component reside on different machines, DCOM simply replaces the local interprocess communication with a network protocol. Neither the client nor the component is aware that the wire that connects them has just become a little longer.

Figure 3 shows the overall DCOM architecture: The COM run-time provides object-oriented services to clients and components and uses RPC and the security provider to generate standard network packets that conform to the DCOM wire-protocol standard.

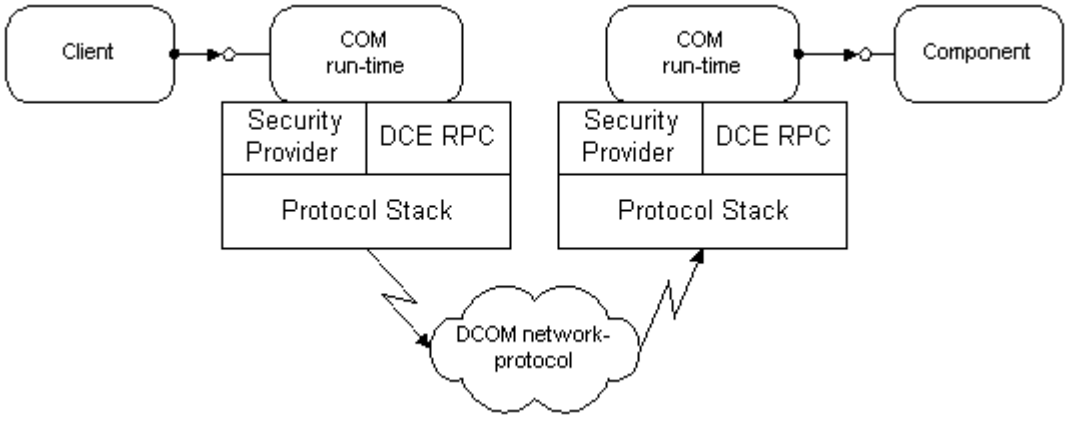

Figure 3: DCOM: COM components on different machines

Components and Reuse

Most distributed applications are not developed from scratch and in a vacuum. Existing hardware infrastructure, existing software, and existing components, as well as existing tools, need to be integrated and leveraged to reduce development and deployment time and cost. DCOM directly and transparently takes advantage of any existing investment in COM components and tools. A huge market for off-the-shelf components makes it possible to reduce development time by integrating standardized solutions into a custom application. Many developers are familiar with COM and can easily apply their knowledge to DCOM-based distributed applications.