OPC

OLE for Process Control

OPC Overview

Version 1.0

October 27, 1998

Specification Type	Industry Standard Specification		
Title:	**OPC Overview**	Date:	October 27, 1998
Version:	1.0	Soft	MS-Word
		Source:	Opcovw.doc
Author:	Opc Task Force	Status:	**Release**

Synopsis:

This specification serves as overview to OPC. It gives background information, motivation, architectural highlights and an abstract for each OPC topic.

Trademarks:

NON-EXCLUSIVE LICENSE AGREEMENT

The OPC Foundation, a non-profit corporation (the "OPC Foundation"), has established a set of standard OLE/COM interface protocols intended to foster greater interoperability between automation/control applications, field systems/devices, and business/office applications in the process control industry.

The current OPC specifications, prototype software examples and related documentation (collectively, the "OPC Materials"), form a set of standard OLE/COM interface protocols based upon the functional requirements of Microsoft's OLE/COM technology. Such technology defines standard objects, methods, and properties for servers of real-time information like distributed process systems, programmable logic controllers, smart field devices and analyzers in order to communicate the information that such servers contain to standard OLE/COM compliant technologies enabled devices (e.g., servers, applications, etc.).

The OPC Foundation will grant to you (the "User"), whether an individual or legal entity, a license to use, and provide User with a copy of, the current version of the OPC Materials so long as User abides by the terms contained in this Non-Exclusive License Agreement ("Agreement"). If User does not agree to the terms and conditions contained in this Agreement, the OPC Materials may not be used, and all copies (in all formats) of such materials in User's possession must either be destroyed or returned to the OPC Foundation. By using the OPC Materials, User (including any employees and agents of User) agrees to be bound by the terms of this Agreement.

LICENSE GRANT:

Subject to the terms and conditions of this Agreement, the OPC Foundation hereby grants to User a non-exclusive, royalty-free, limited license to use, copy, display and distribute the OPC Materials in order to make, use, sell or otherwise distribute any products and/or product literature that are compliant with the standards included in the OPC Materials.

All copies of the OPC Materials made and/or distributed by User must include all copyright and other proprietary rights notices include on or in the copy of such materials provided to User by the OPC Foundation.

The OPC Foundation shall retain all right, title and interest (including, without limitation, the copyrights) in the OPC Materials, subject to the limited license granted to User under this Agreement.

WARRANTY AND LIABILITY DISCLAIMERS:

User acknowledges that the OPC Foundation has provided the OPC Materials for informational purposes only in order to help User understand Microsoft's OLE/COM technology. THE OPC MATERIALS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF PERFORMANCE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. USER BEARS ALL RISK RELATING TO QUALITY, DESIGN, USE AND PERFORMANCE OF THE OPC MATERIALS. The OPC Foundation and its members do not warrant that the OPC Materials, their design or their use will meet User's requirements, operate without interruption or be error free.

IN NO EVENT SHALL THE OPC FOUNDATION, ITS MEMBERS, OR ANY THIRD PARTY BE LIABLE FOR ANY COSTS, EXPENSES, LOSSES, DAMAGES (INCLUDING, BUT NOT LIMITED TO, DIRECT, INDIRECT, CONSEQUENTIAL, INCIDENTAL, SPECIAL OR PUNITIVE DAMAGES) OR INJURIES INCURRED BY USER OR ANY THIRD PARTY AS A RESULT OF THIS AGREEMENT OR ANY USE OF THE OPC MATERIALS.

GENERAL PROVISIONS:

This Agreement and User's license to the OPC Materials shall be terminated (a) by User ceasing all use of the OPC Materials, (b) by User obtaining a superseding version of the OPC Materials, or (c) by the OPC Foundation, at its option, if User commits a material breach hereof. Upon any termination of this Agreement, User shall immediately cease all use of the OPC Materials, destroy all copies thereof then in its possession and take such other actions as the OPC Foundation may reasonably request to ensure that no copies of the OPC Materials licensed under this Agreement remain in its possession.

User shall not export or re-export the OPC Materials or any product produced directly by the use thereof to any person or destination that is not authorized to receive them under the export control laws and regulations of the United States.

The Software and Documentation are provided with Restricted Rights. Use, duplication or disclosure by the U.S. government is subject to restrictions as set forth in (a) this Agreement pursuant to DFARs 227.7202-3(a); (b) subparagraph (c)(1)(i) of the Rights in Technical Data and Computer Software clause at DFARs 252.227-7013; or (c) the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 subdivision (c)(1) and (2), as applicable. Contractor/ manufacturer is the OPC Foundation, P.O. Box 140524, Austin, Texas 78714-0524.

Should any provision of this Agreement be held to be void, invalid, unenforceable or illegal by a court, the validity and enforceability of the other provisions shall not be affected thereby.

This Agreement shall be governed by and construed under the laws of the State of Minnesota, excluding its choice or law rules.

This Agreement embodies the entire understanding between the parties with respect to, and supersedes any prior understanding or agreement (oral or written) relating to, the OPC Materials.

Table of Contents

1. Introduction

1.1 Readers Guide

This document serves as an overview to OPC. It gives background information, motivation, architectural highlights and an abstract for each OPC topic.

Specific interface specifications to develop OPC clients and/or OPC Servers (e.g., for DataAccess, Alarm&Event Handling or Historical DataAccess) and a specification for interfaces that are common for all OPC Servers are available as separate documents.

Chapter 1 gives some background information. It describes the purpose of OPC and why and how both vendors and customers have advantages in using OPC.

Chapter 2 provides a technical overview, describing the fundamentals of the design and characteristics of OPC components.

Chapter 3 describes the way OPC supports browsing of servers both locally and on remote machines.

1.2 OPC Background

A standard mechanism for communicating to numerous data sources, either devices on the factory floor, or a database in a control room is the motivation for OPC.

The information architecture for the Process Industry shown in Figure 1-1 involves the following levels:

Field Management. With the advent of “smart” field devices, a wealth of information can be provided concerning field devices that was not previously available. This information provides data on the health of a device, its configuration parameters, materials of construction, etc. All this information must be presented to the user, and any applications using it, in a consistent manner.

Process Management. The installation of Distributed Control Systems (DCS) and SCADA systems to monitor and control manufacturing processes makes data available electronically which had been gathered manually.

Business Management. Benefits can be gained by installing the control systems. This is accomplished by integrating the information collected from the process into the business systems managing the financial aspects of the manufacturing process. Providing this information in a consistent manner to client applications minimizes the effort required to provide this integration.

To do these things effectively, manufacturers need to access data from the plant floor and integrate it into their existing business systems. Manufacturers must be able to utilize off the shelf tools (SCADA Packages, Databases, spreadsheets, etc.) to assemble a system to meet their needs. The key is an open and effective communication architecture concentrating on data access, and not the types of data.

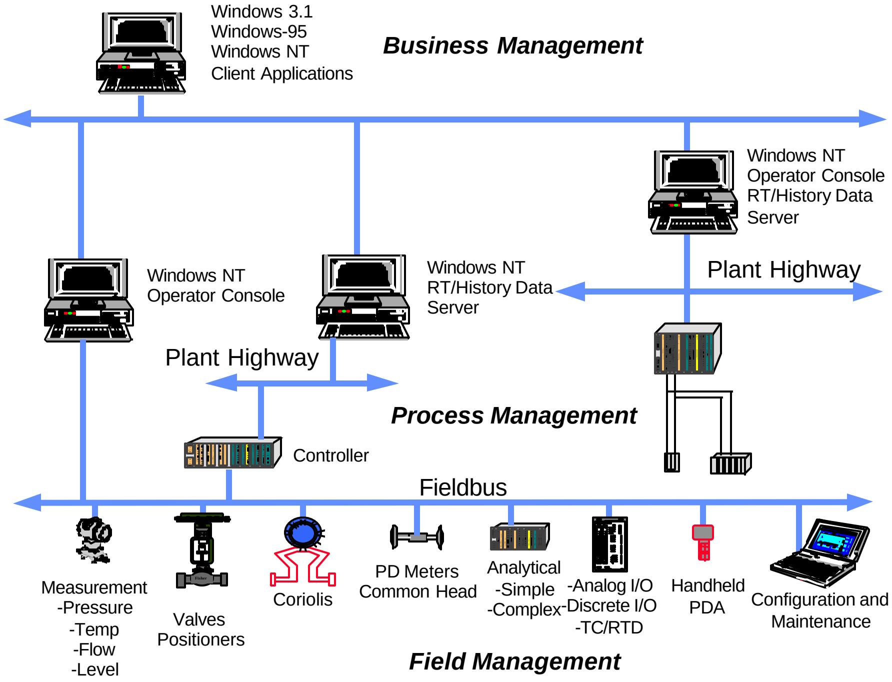

Figure 1-1 Process Control Information Architecture

1.3 Purpose

What is needed is a common way for applications to access data from any data source like a device or a database.

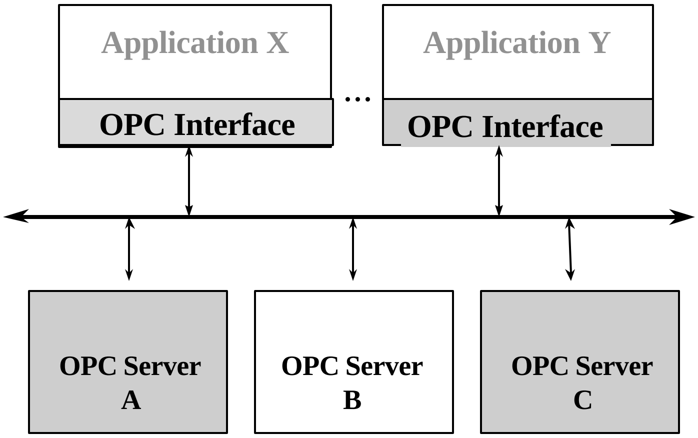

Figure 1-2. Applications Working with Many OPC Servers

OPC Server in this figure and in the following sections is used as synonym for any server that provides OPC interfaces, e.g.,

OPC DataAccess Server,
OPC Alarm&Event Server,
OPC HistoricalData Server.

1.3.1 The Current Client Application Architecture

There are many client applications that have been developed that require data from a data source and access that data by independently developing "Drivers" for their own packages.

This leads to the problems that follow:

- **Much duplication of effort**
  Everyone must write a driver for a particular vendor's hardware.
- **Inconsistencies between vendors drivers**
  Hardware features not supported by all driver developers.
- **Support for hardware feature changes**
  A change in the hardware's capabilities may break some drivers
- **Access Conflicts**
  Two packages generally cannot access the same device simultaneously since they each contain independent Drivers.

Hardware manufacturers attempt to resolve these problems by developing drivers, but are hindered by differences in client protocols. Today they cannot develop an efficient driver that can be used by all clients.

OLE for Process Control (OPC™) draws a line between hardware providers and software developers. It provides a mechanism to provide data from a data source and communicate the data to any client application in a standard way. A vendor can now develop a reusable, highly optimized server to communicate to the data source, and maintain the mechanism to access data from the data source/device efficiently. Providing the server with an OPC interface allows any client to access their devices.

1.3.2 The Custom Application Architecture

A growing number of custom applications are being developed in environments like Visual Basic (VB), Delphi, Power Builder, etc. OPC must take this trend into account. Microsoft understands this trend and designed OLE/COM to allow components (written in C and C++ by experts in a specific domain) to be utilized by a custom program (written in VB or Delphi for an entirely different domain). Developers will write software components in C and C++ to encapsulate the intricacies of accessing data from a device, so that business application developers can write code in VB that requests and utilizes plant floor data.

The intent of all specifications is to facilitate the development of OPC Servers in C and C++, and to facilitate development of OPC client applications in the language of choice.

The architecture and design of the interfaces are intended to support development of OPC servers in other languages as well.

1.3.3 General

OLE for Process Control (OPC™) is designed to allow client applications access to plant floor data in a consistent manner. With wide industry acceptance OPC will provide many benefits:

- Hardware manufacturers only have to make one set of software components for customers to utilize in their applications.
- Software developers will not have to rewrite drivers because of feature changes or additions in a new hardware release.
- Customers will have more choices with which to develop World Class integrated manufacturing systems.

With OPC, system integration in a heterogeneous computing environment will become simple. Leveraging OLE/COM the environment shown in Figure 1-3 becomes possible.

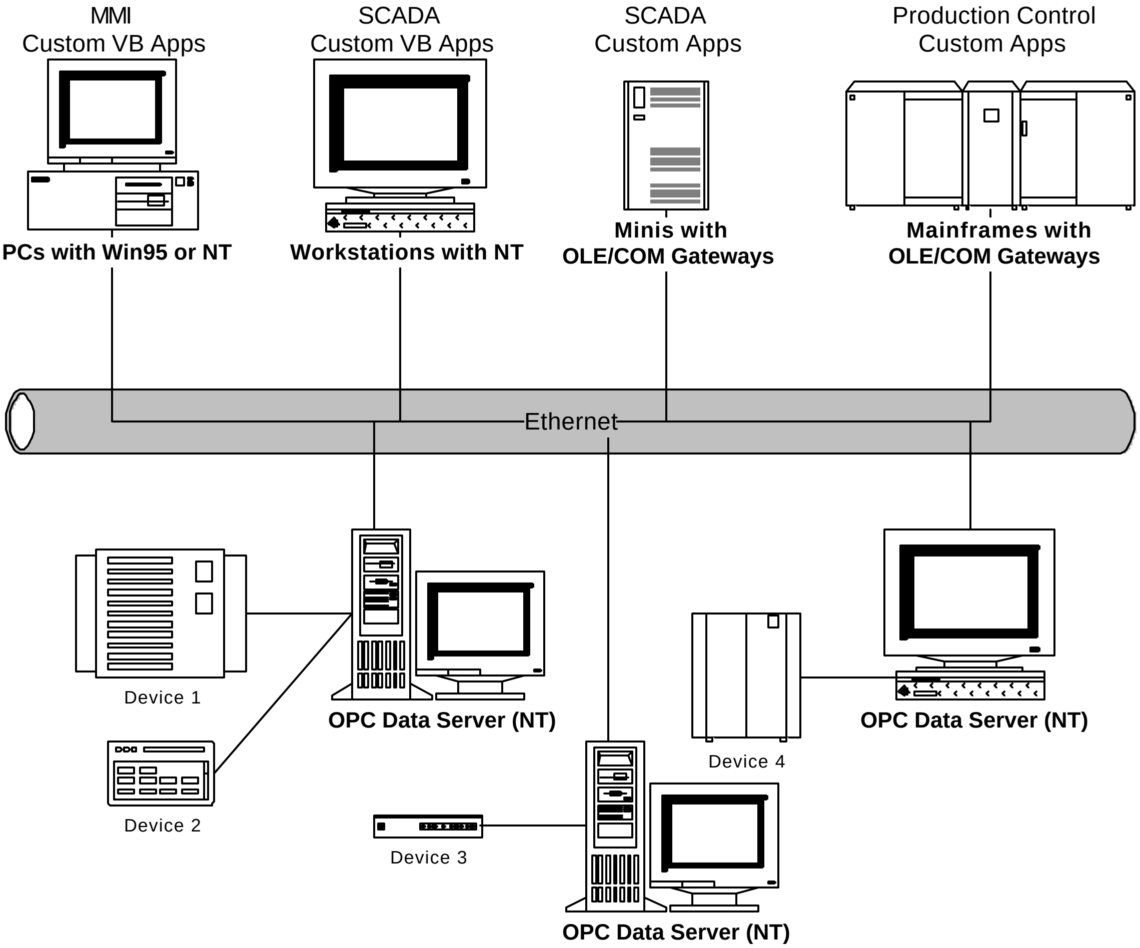

Figure 1-3. Heterogeneous Computing Environment

1.4 Scope

A primary goal for OPC is to deliver specifications to the industry as quickly as possible. With this in mind, the scope of the first document releases is limited to areas common to all vendors. Additional functionality will be defined in future releases. Therefore, the first releases focus on

- Online DataAccess, i.e., the efficient reading and writing of data between an application and a process control device flexibly and efficiently;
- Alarm and Event Handling, i.e., the mechanisms for OPC Clients to be notified of the occurrence of specified events and alarm conditions, and
- Historical Data Access, i.e., the reading, processing and editing of data of a historian engine.

Functionality such as security, batch and historical alarm and event data access belong to the features which are addressed in subsequent releases. The architecture of OPC leverages the advantages of the COM interface, which provides a convenient mechanism to extend the functionality of OPC.

Other goals for the design of OPC were as follows:

- simple to implement
- flexible to accommodate multiple vendor needs
- provide a high level of functionality
- allow for efficient operation

The specifications include the following:

1. A set of custom COM interfaces for use by client and server writers.
2. References to a set of OLE Automation interfaces to support clients developed with higher level business applications such as Excel, Visual Basic, etc.

The architecture is intended to utilize the Microsoft distributed OLE technology (DCOM) to facilitate clients interfacing to remote servers.

1.5 References

Kraig Brockschmidt ,Inside OLE, Second Edition, Microsoft Press, Redmond, WA, 1995.

Microsoft COM Specification, version 0.9, 10/24/95 (available from Microsoft's FTP site).

OLE Automation Programming Reference, Microsoft Press, Redmond, WA, 1996.

OLE 2 Programming Reference, Vol. 1, Microsoft Press, Redmond, WA, 1994.

The OPC Data Access Custom Specification 1.0A, OPC Foundation 1997.

The OPC Data Access Custom Specification 2.0, OPC Foundation 1998.

The OPC Data Access Automation Specification 2.0, OPC Foundation 1998.

The OPC Alarm and Event Access Specification 1.0, OPC Foundation 1998.

The OPC Historical Data Access Specification 1.0, OPC Foundation 1998.

2. OPC Fundamentals

OPC is based on Microsoft's OLE/COM technology.

2.1 OPC Objects and Interfaces

This specification describes the OPC COM Objects and their interfaces implemented by OPC Servers. An OPC Client can connect to OPC Servers provided by one or more vendors.

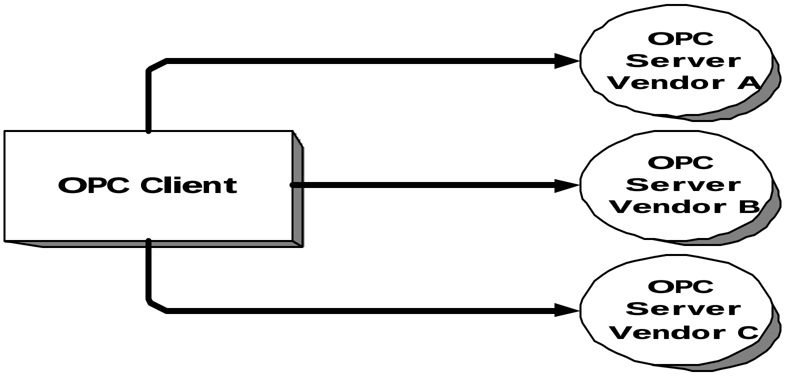

Figure 2-1 OPC Client

OPC Servers may be provided by different vendors. Vendor supplied code determines the devices and data to which each server has access, the data names, and the details about how the server physically accesses that data.

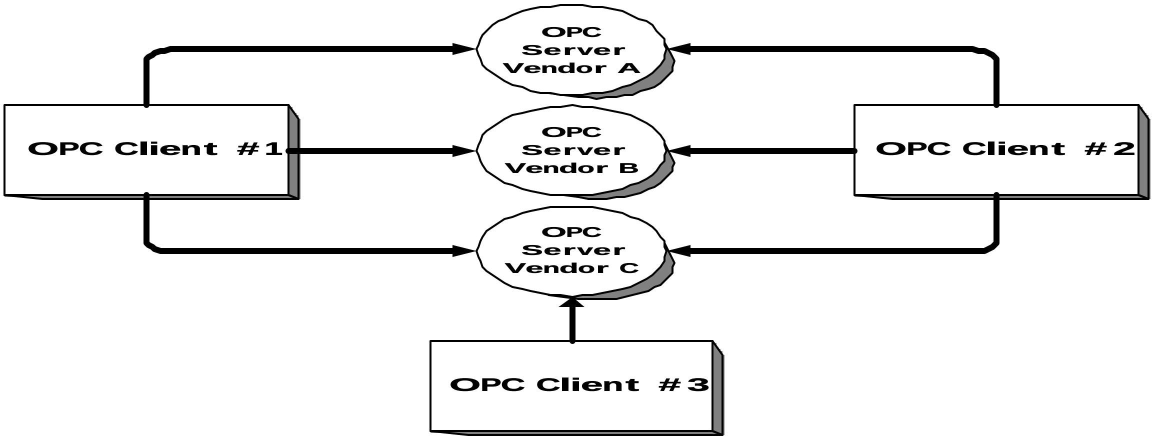

Figure 2-2 OPC Client/Server Relationship

2.1.1 OPC DataAccess Overview

At a high level, an OPC DataAccess Server is comprised of several objects: the server, the group, and the item. The OPC server object maintains information about the server and serves as a container for OPC group objects. The OPC group object maintains information about itself and provides the mechanism for containing and logically organizing OPC items.

The OPC Groups provide a way for clients to organize data. For example, the group might represent items in a particular operator display or report. Data can be read and written. Exception based connections can also be created between the client and the items in the group and can be enabled and disabled as needed. An OPC client can configure the rate that an OPC server should provide the data changes to the OPC client.

There are two types of groups, public and local (or 'private'). Public is for sharing across multiple clients, local is local to a client. Refer to the section on public groups for the intent, purpose, and functionality and for further details. There are also specific optional interfaces for the public groups.

Within each Group the client can define one or more OPC Items.

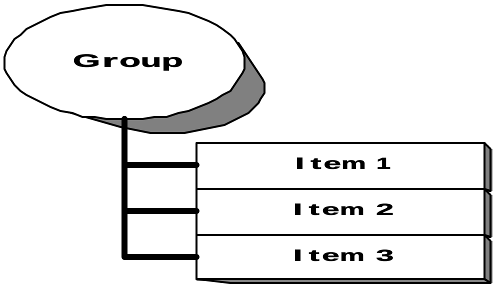

Figure 2-3 - Group/Item Relationship

The OPC Items represent connections to data sources within the server. An OPC Item, from the custom interface perspective, is not accessible as an object by an OPC Client. Therefore, there is no external interface defined for an OPC Item. All access to OPC Items is via an OPC Group object that "contains" the OPC item, or simply where the OPC Item is defined.

Associated with each item is a Value, Quality and Time Stamp. The value is in the form of a VARIANT, and the Quality is similar to that specified by Fieldbus.

Note that the items are not the data sources - they are just connections to them. For example, the tags in a DCS system exist regardless of whether an OPC client is currently accessing them. The OPC Item should be thought of as simply specifying the address of the data, not as the actual physical source of the data that the address references.

2.1.2 OPC Alarm and Event Handling Overview

These interfaces provide the mechanisms for OPC Clients to be notified of the occurrence of specified events and alarm conditions. They also provide services which allow OPC Clients to determine the events and conditions supported by an OPC Server, and to obtain their current status.

We make use of entities commonly referred to in the process control industry as *alarms* and *events*. In informal conversation, the terms *alarm* and *event* are often used interchangeably and their meanings are not distinct.

Within OPC, an *alarm* is an abnormal *condition* and is thus a special case of a *condition*. A *condition* is a named state of the OPC Event Server, or of one of its contained objects, which is of interest to its OPC Clients. For example, the tag FC101 may have the following conditions associated with it: HighAlarm, HighHighAlarm, Normal, LowAlarm, and LowLowAlarm.

On the other hand, an *event* is a detectable occurrence which is of significance to the OPC Server, the device it represents, and its OPC Clients. An event may or may not be associated with a condition. For example, the transitions into HighAlarm and Normal conditions are events which are associated with conditions. However, operator actions, system configuration changes, and system errors are examples of events which are not related to specific conditions. OPC Clients may subscribe to be notified of the occurrence of specified events.

The IOPCEventServer interface provides methods enabling the OPC Client to:

- Determine the types of events which the OPC Server supports.
- Enter subscriptions to specified events, so that OPC Clients can receive notifications of their occurrences. Filters may be used to define a subset of desired events.
- Access and manipulate conditions implemented by the OPC Server.

In addition to the IOPCEventServer interface, an OPC Event Server may support optional interfaces for browsing conditions implemented by the server and for managing public condition groups (defined in the following section).

2.1.3 OPC Historical Data Access Overview

Historical engines today produce an added source of information that must be distributed to users and software clients that are interested in this information. Currently most historical systems use their own proprietary interfaces for dissemination of data. There is no capability to augment or use existing historical solutions with other capabilities in a plug-n-play environment. This requires the developer to recreate the same infrastructure for their products as all other vendors have had to develop independently with no interoperability with any other systems.

In keeping with the desire to integrate data at all levels of a business, historical information can be considered to be another type of data.

There are several types of Historian servers. Some key types supported by this specification are:

- Simple Trend data servers. These servers provided little else then simple raw data storage. (Data would typically be the types of data available from an OPC Data Access server, usually provided in the form of a tuple [Time Value & Quality])
- Complex data compression and analysis servers. These servers provide data compression as well as raw data storage. They are capable of providing summary data or data analysis functions, such as average values, minimums and maximums etc. They can support data updates and history of the updates. They can support storage of annotations along with the actual historical data storage.

2.2 Where OPC Fits

Although OPC is primarily designed for accessing data from a networked server, OPC interfaces can be used in many places within an application. At the lowest level they can get raw data from the physical devices into a SCADA or DCS, or from the SCADA or DCS system into the application.. The architecture and design makes it possible to construct an OPC Server which allows a client application to access data from many OPC Servers provided by many different OPC vendors running on different nodes via a single object.

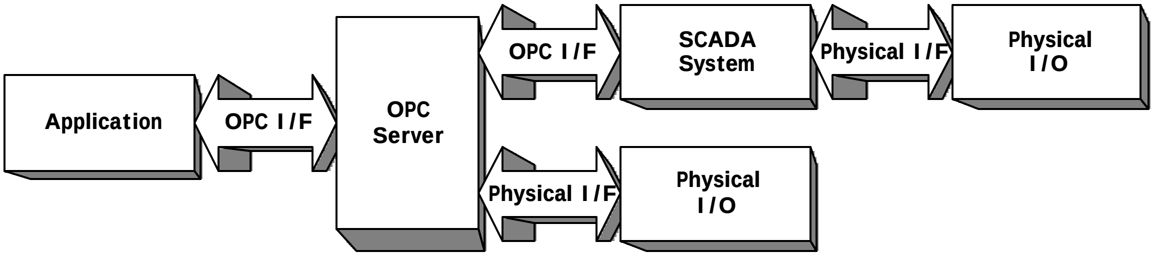

Figure 2-4 - OPC Client/Server Relationship

2.3 General OPC Architecture and Components

OPC specifications always contain two sets of interfaces; Custom Interfaces and Automation interfaces. This is shown in Figure 2-5.

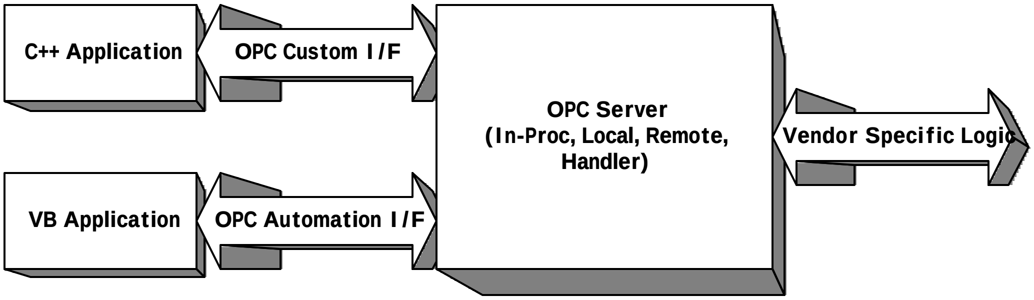

Figure 2-5 - The OPC Interfaces

The OPC Specification specifies COM interfaces (what the interfaces are), not the implementation (not the how of the implementation) of those interfaces. It specifies the behavior that the interfaces are expected to provide to the client applications that use them.

Included are descriptions of architectures and interfaces that seemed most appropriate for those architectures. Like all COM implementations, the architecture of OPC is a client-server model where the OPC Server component provides an interface to the OPC objects and manages them.

There are several unique considerations in implementing an OPC Server. The main issue is the frequency of data transfer over non-sharable communications paths to physical devices or other data bases. Thus, we expect that OPC Servers will either be a local or remote EXE which includes code that is responsible for efficient data collection from a physical device or a data base.

An OPC client application communicates to an OPC server through the specified custom and automation interfaces. OPC servers must implement the custom interface, and optionally may implement the automation interface. In some cases the OPC Foundation provides a standard automation interface wrapper. This “wrapperDLL” can be used for any vendor-specific custom-server.

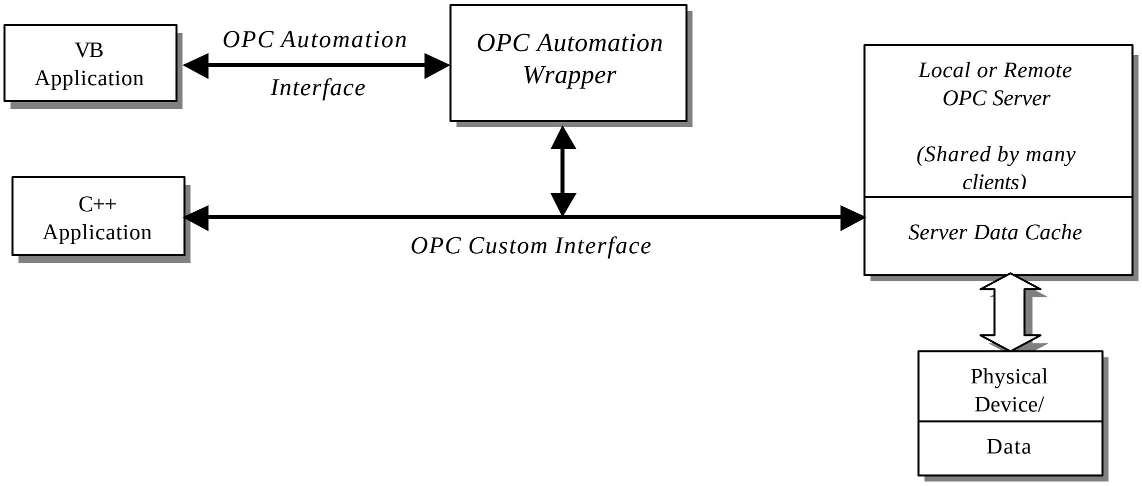

Figure 2-6 - Typical OPC Architecture

2.4 Local vs. Remote Servers

It is expected that OPC Server vendors will take one of two approaches to networking:

1. They can indicate that the client should always connect to a local server which makes use of an existing proprietary network scheme. This approach will commonly be used by vendors who are adding OPC capability to an existing distributed product.
2. They can indicate the client should connect to the desired server on the target node and make use of DCOM™ to provide networking. For this reason all of the RPC_E_* error codes should also be considered as possible returns from the functions below.

3. OPC Server Browser

The Interface of the OPC Server Browser (IOPCServerList) is specified as part of the OPCCommon document.

3.1 Overview of the Problem

The issue is, "How does a client program show the user what OPC Servers are available on a particular machine?". OPC Servers now register with the system via Component Categories. This allows the Microsoft ICatInformation (IID_ICatInformation) Interface on the StdComponentCatagoriesMgr (CLSID_StdComponentCategoriesMgr) to be used to determine which OPC servers are installed on the local machine. The problem is that this does not work for remote machines because the Component Categories Manager is a DLL and the ICatInformation interface only works in-process. As a result there is no easy way for a Client (including the Foundation supplied Automation Wrappers) to obtain a list of OPC Servers installed on a remote machine.

NOTE: the issue under discussion here is Server Browsing. This is entirely different from the Address Space browsing discussed in the OPC Data Access Interface.

3.2 Overview of the Solution

The OPC Foundation supplied Server Browser OPCENUM.EXE can reside on any machine, will access the local Component Categories Manger and provides a new interface IOPCServerList which can be marshaled and used by remote clients. This server has a published classid (see below) and can be installed once on any machine which hosts OPC servers. The client still needs to know the nodename of the target machine however he can now create this object remotely and use it's IOPCServerList interface to determine what types and brands of servers are available on that machine.