SCADAServer Host /OPC Server

User Manual

CONTROL MICROSYSTEMS

SCADA products... for the distance

28 Steacie Drive
Kanata, Ontario
K2K 2A9
Canada

Telephone: 613-591-1943
Facsimile: 613-591-1022
Technical Support: 888-226-6876
888-2CONTROL

SCADAServer Host OPC User Manual

Printed in Canada.

Trademarks

Table of Contents

TABLE OF CONTENTS 3

OVERVIEW 8

Organization of the Manual 9

Additional Documentation 9

GETTING STARTED 11

System Requirements 11

SCADAServer Host Installation 11

License the SCADAServer Host Program 11

Run the SCADAServer Host Program 12

CONFIGURING THE SCADASERVER HOST 13

Step 1: Create a SCADAServer Host Configuration File. 13

Step 2: Add a new SCADAServer Host Connection. 13

Step 3: Configure Serial Connection Properties. 14

Port Properties 14

Type Properties 14

Step 4: Configure Internet Client or Server Connection Properties. 15

Internet Client Connection Properties 15

Internet Server Connection Properties 16

Step 5: Configure Virtual PLC Properties. 16

Step 6: Close Connection Properties Dialog. 16

Step 7: Configure SCADAServer Host Options. 16

Step 8: Configure Data Views. 17

Step 9: Save SCADAServer Host Configuration. 17

SCADASERVER VIRTUAL PLC 18

SCADASERVER ADDRESS SPACE 19

Item ID 19

Access Path 19

SCADASERVER HOST REFERENCE 21

SCADAServer Host Display Window 21

Title Bar 21

Tool Bar 22

Status Bar 23

Menu Bar 23

File Menu Commands 23

New Command....23
Open Command....23
Save Command....23
Save As Command....24
Exit Command....24
View Menu Commands....24
Log Command....24
Pools Command....25
Statistics Command....25
Toolbar Command....26
Status Bar Command....26
Server Menu Commands....26
Connections Command....26
Options Command....40
Clear Log Command....41
Clear Statistics Command....41
Window Menu Commands....41
New Window Command....41
Cascade Command....41
Tile Command....41
Arrange All Command....41
Open Window List....41
Help Commands....42
Contents Command....42
About Command....42

SCADASERVER CLIENT EXAMPLES....43
TelePACE Client....43
Configuring PC Communication Settings....43
ISaGRAF Client....43
Configuring PC Communication Settings....44
TelePACE Firmware Loader Client....44
Configuring PC Communication Settings....44
RealFLO Client....45
Configuring PC Communication Settings....45
SCADALog Client....46
Configuring PC Communication Settings....46
LOOKOUT 4.0.1 Client....47
Configuring OPCClient Object....47
Configuring OPCClient Database....48
Wonderware® OPCLink and Wonderware Client....49
Configure OPCLink....49
Connect to SCADAServer Host with Wonderware Client....52

POLLNOW FEATURE....58

USING A REMOTE COMPUTER CONNECTION....59

SCADASERVER OLE AUTOMATION REFERENCE....61
OPCServer Object....62

Summary of Properties 62
Summary of Methods 62
Summary of Events 62
OPCServer Properties 62
StartTime 62
CurrentTime 63
LastUpdateTime 63
MajorVersion 63
MinorVersion 63
BuildNumber 64
VendorInfo 64
ServerState 64
LocaleID 65
Bandwidth 65
OPCGroups 66
PublicGroupNames 66
ServerName 66
ServerNode 66
ClientName 67
OPCServer Methods 67
GetOPCServers 67
Connect 68
Disconnect 68
CreateBrowser 69
GetErrorString 69
QueryAvailableLocaleIDs 69
QueryAvailableProperties 70
GetItemProperties 70
LookupItemIDs 71
OPCServer Events 72
ServerShutDown 72
OPCBrowser Object 72
Summary of Properties 73
Summary of Methods 73
OPCBrowser Properties 74
Organization 74
Filter 74
DataType 74
AccessRights 74
CurrentPosition 75
Count 75
OPCBrowser Methods 75
Item 75
ShowBranches 76
ShowLeafs 76
MoveUp 76
MoveToRoot 76
MoveDown 77
MoveTo 77
GetItemID 77
GetAccessPaths 77
OPCGroups Object 78
Summary of Properties 78
Summary of Methods 78

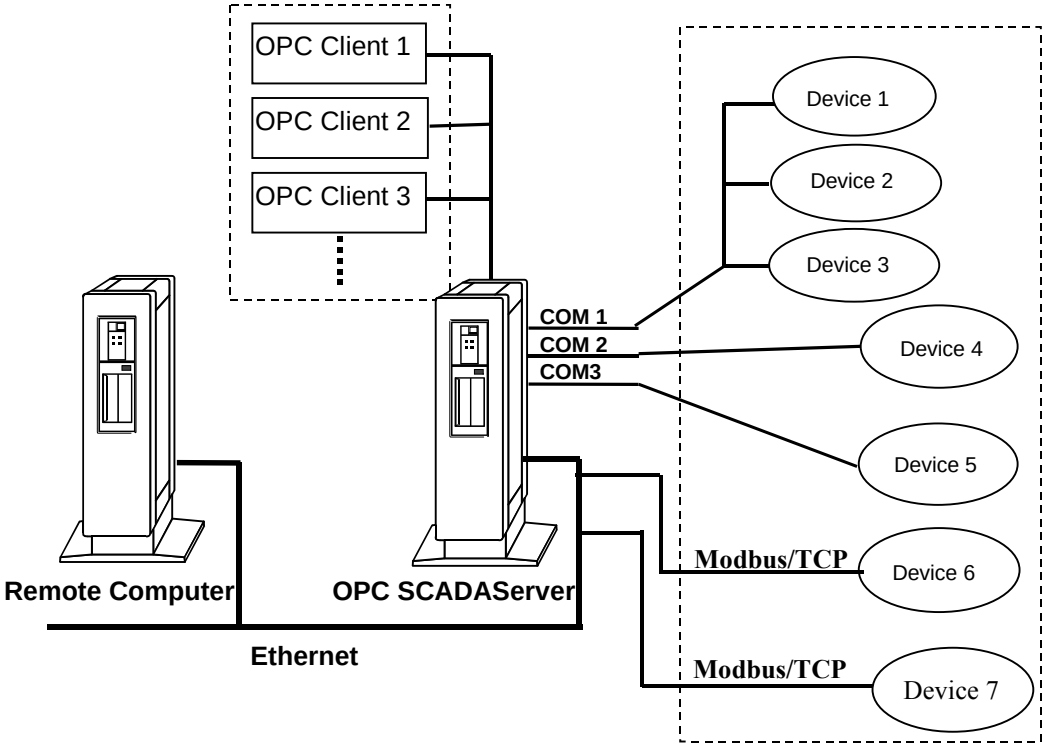
Summary of Events 78
OPCGroups Properties 79
Parent 79
DefaultGroupIsActive 79
DefaultGroupUpdateRate 79
DefaultGroupDeadband 80
DefaultGroupLocaleID 80
DefaultGroupTimeBias 80
Count 80
OPCGroups Methods 81
Item 81
Add 81
GetOPCGroup 81
Remove 81
RemoveAll 82
ConnectPublicGroup 82
RemovePublicGroup 82
OPCGroups Events 83
GlobalDataChange 83
OPCGroup Object 84
Summary of Properties 84
Summary of Methods 84
Summary of Events 84
OPCGroup Properties 85
Parent 85
Name 85
IsPublic 85
IsActive 86
IsSubscribed 86
ClientHandle 86
ServerHandle 87
LocaleID 87
TimeBias 87
DeadBand 88
UpdateRate 88
OPCItems 89
OPCGroup Methods 89
SyncRead 89
SyncWrite 90
AsyncRead 91
AsyncWrite 92
AsyncRefresh 92
AsyncCancel 93
OPCGroup Events 94
DataChange 94
AsyncReadComplete 94
AsyncWriteComplete 95
AsyncCancelComplete 95
OPCItems Object 96
Summary of Properties 96
Summary of Methods 96
OPCItems Properties 97
Parent 97
DefaultRequestedDataType 97

DefaultAccessPath 97
DefaultIsActive 97
Count 98
OPCItems Methods 98
Item 98
GetOPCItem 98
AddItem 98
AddItems 99
Remove 99
Validate 100
SetActive 100
SetClientHandles 101
SetDataTypes 101
OPCItem Object 102
Summary of Properties 102
Summary of Methods 102
OPCItem Properties 102
Parent 102
ClientHandle 102
ServerHandle 102
AccessPath 103
AccessRights 103
ItemID 103
IsActive 103
RequestedDataType 104
Value 104
Quality 104
TimeStamp 105
CanonicalDataType 105
EUType 105
EUInfo 105
OPCItem Methods 106
Read 106
Write 106

Overview

OPC is OLE for Process Control. It is a communication technology that provides interoperability between products from many vendors. The standard OPC interface allows makers of Human-Machine-Interface software, custom application developers and makers of field hardware, including PLCs, to know that they will be able to communicate with others. It also allows multiple applications and hosts to share communication networks with field devices.

OPC has two main components. They are OPC data servers and OPC clients. OPC servers form the interface between custom protocols or hardware, and HMI software or custom applications. OPC clients are the programs that use the data and provide control information to field devices.



Clients include Control Microsystems TelePACE, ISaGRAF, TelePACE Firmware Loader and RealFLO applications and many third-party applications. The OPC Automation interface allows clients such as Visual BASIC and Delphi applications to access the OPC server.

When SCADAServer Host is installed on the same PC as the client applications it is referred to as a local SCADAServer connection for the client applications. All Control Microsystems clients support a local SCADAServer connection. When SCADAServer Host is installed on a different computer than the client applications it is referred to as a remote SCADAServer connection for the client applications. All Control Microsystems clients, with the exception of TelePACE, support a remote SCADAServer connection.

SCADAServer Host supports serial connections. Installed serial ports are automatically detected by SCADAServer Host, which provides a serial connection to each. The following three serial connection types are supported:

- No Flow Control.
- RTS/CTS Flow Control.
- Dial-up.

SCADAServer Host supports Internet Client connections. The following Internet Client connections are supported:

- Modbus/TCP
- Modbus/UDP
- Modbus RTU in TCP
- Modbus RTU in UDP
- Modbus ASCII in TCP
- Modbus ASCII in UDP

SCADAServer Host supports Internet Server connections. The following Internet Server connections are supported:

- Modbus ASCII in TCP Server
- Modbus ASCII in UDP Server
- Modbus RTU in TCP Server
- Modbus RTU in UDP Server
- Modbus/TCP Server
- Modbus/UDP Server

Complete information for Modbus/TCP and serial Modbus may be found on-line at **www.modbus.org/**.

Field devices include Control Microsystems SCADAPack controller family and Micro 16 controllers. Third party devices include Man Machine Interfaces (MMI), PLCs and RTUs that communicate using Modbus RTU or ASCII protocol.

A feature unique to the SCADAServer Host is the Virtual PLC. The SCADAServer Host can be configured to act as a slave device or devices on Modbus networks. Thereby creating a "Virtual" controller that exists only in the SCADAServer Host. The Virtual PLC is ideal when implemented in a report-by-exception operation.

Organization of the Manual

New users should read through the sections, in order, to gain an understanding of underlying concepts of the SCADAServer Host. The remainder of the user manual is organized as follows.

Getting Started section describes system components and installation procedures.

Configuring the SCADAServer Host section describes how to use and configure the SCADAServer Host.

SCADAServer Host Reference section provides a complete description of the SCADAServer Host commands.

SCADAServer Client Examples section describes using TelePACE, ISaGRAF, RealFLO and Firmware Loader applications as OPC Clients.

SCADAServer OLE Automation Reference section describes using the OPC Automation interface with the SCADAServer Host.

Additional Documentation

The ***SCADAPack & Micro16 System Manual*** is a complete reference to controller and I/O modules used with SCADAPack

The ***TelePACE Ladder Logic Reference and User Manual*** describes the creation of application programs in the Ladder Logic language.

The ***TeleBUS Protocols User Manual*** describes communication using Modbus compatible protocols.

Getting Started

To use the SCADAServer Host, you need to install the program on your system. The automated installation takes only a few minutes. Some virus checking software may interfere with the Setup. If you experience problems with Setup, disable your virus checker and run Setup again.

System Requirements

The SCADAServer Host program requires the following minimum system configuration.

- Microsoft Windows NT, Windows 2000 or Windows XP operating systems.

  NOTE: Windows 95, 98 and ME operating systems are no longer supported.
- Minimum 8 MB of memory.
- Mouse or compatible pointing device.
- Hard disk with approximately 2.5 Mbytes of free disk space.

SCADAServer Host Installation

To install the SCADAServer Host:

- Insert the SCADAServer Host CD in your CD-ROM drive.
- From the Start menu select Run.
- Enter d:\setup (where d is your CD-ROM drive) and click OK.
- Follow the on screen instructions.

License the SCADAServer Host Program

The SCADAServer Host software will function for a seven-day trial period from the date of original installation on your PC. You MUST contact Control Microsystems Sales Department for a SCADAServer Host software license to enable the software to function for longer than the seven-day trial period.

To License the SCADAServer Host software:

- Run the SCADAServer Host from the Windows Start menu.
- Press the <enter> key on your keyboard when the SCADAServer Host copyright screen is displayed.

Note:	Once the seven-day trial period is completed the SCADAServer Host License Configuration Dialog will be displayed each time an attempt is made to run SCADAServer Host.

- Click the **Display Site Code…** button to display the License Agreement. You must accept the License Agreement in order to proceed.
- The **Display Site Code…** button disappears and your 18 digit **Site Code** is displayed.

- Contact Control Microsystems Sales Order Entry at 1-888-267-2232 or orderprocessing@controlmicrosystems.com to receive the Site Key for your SCADAServer Host software.
- You will need to have your original Sales Order Number and the Site Code available when you contact Sales Order Entry.
- Type the Site Key provided to you into the Site Key entry window.
- Confirm the correct Site Key is entered and click the **Validate** button to license the SCADAServer Host software.

Contact Control Microsystems Technical Support at 1-888-226-6876 or support@controlmicrosystems.com if you have any questions.

Run the SCADAServer Host Program

To run the SCADAServer Host Program:

- In the SCADAServer Host group of the Program manager, double click on the SCADAServer Host icon.

Configuring the SCADAServer Host

This section of the manual is organized in such a way that will lead the new user through the steps required to configure and use the SCADAServer Host. It is recommended that new users read through this section of the manual and then follow the steps shown to understand and properly configure the SCADAServer Host.

Step 1: Create a SCADAServer Host Configuration File.

Step 2: Add a new SCADAServer Host Connection.

Step 3: Configure Serial Connection.

Step 4: Configure Internet Client or Server Connection.

Step 5: Configure Virtual PLC.

Step 6: Close Connection Properties Dialog.

Step 7: Configure SCADAServer Host Options.

Step 8: Configure Data Views.

Step 9: Save SCADAServer Host Configuration.

Step 1: Create a SCADAServer Host Configuration File.

The SCADAServer Host configuration file contains all the configuration parameters for the SCADAServer Host. Only one SCADAServer Host configuration file may be opened at one time. To create a new configuration file:

- Open the SCADAServer Host program.
- Select **New** from the **File** menu.

Step 2: Add a new SCADAServer Host Connection.

The SCADAServer Host communicates with remote controllers through the serial ports or an internal Ethernet card on the PC running the SCADAServer Host application. Each serial port on the PC, to a maximum determined by the operating system limitation, may be connected to a single controller or a group of controllers. The Ethernet card may be connected to any number of single controllers or groups of controllers limited only by the availability of IP addresses.

All the controllers in a group and the SCADAServer Host serial port connected to the group must have the same serial communication properties. These properties include the type of connection used and the serial communication parameters for the connection.

- Select **Connections** from the **Server** menu.

The Connections dialog box appears allowing the connection of SCADAServer-detected serial ports or an internal Ethernet card in the PC with the SCADAServer Host application. These serial ports or Ethernet card will be used by the SCADAServer Host to communicate with a controller or group of controllers.

- Select the **Add** button and then choose one of the available serial, internet client or internet host connections.

If a serial connection has been chosen skip step 4. If an internet connection has been chosen, skip step 3.

Step 3: Configure Serial Connection.

SCADAServer Host accommodates direct serial connections, dial-up modem connections and leased line or radio modem connections. These are described in the following sections.

Port Properties

The port properties set the serial parameters for the selected serial port, which is displayed in the header of the Properties dialog.

- Select the **Serial Port** connection from the Connections list.
- From the Connections dialog box select the **Properties** button.
- Select the **Port** tab from the Serial Port Properties dialog.

The serial port parameters include the protocol type used for the port, baud rate, parity setting, number of data and stop bits, protocol addressing type, the timeout of the communication attempt and the number of communication attempts.

- Enter the required parameters for the serial port. Refer to the ***SCADAServer Host Reference*** section for complete information about each entry parameter.

Type Properties

The SCADAServer Host supports three types of serial connection types. These are no flow control, RTS/CTS flow control and dial up modem connections.

- Select the **Type** tab from the Serial Port Properties dialog.
- Select one of the connection types by clicking on the appropriate radio button.

No Flow Control

The No Flow Control type connections do not require hardware control lines. These connections are typically used with a direct serial cable connection between the SCADAServer Host and the controller. This type of connection will work with a three wire (Rx, Tx and GND) serial cable.

- Click the **No Flow Control** radio button if this type of serial connection is to be used.

RTS/CTS Flow Control

RTS/CTS flow control connections are used with any communication system requiring hardware flow control. Hardware flow control or RTS (request to send) / CTS (clear to send) handshaking is typically used with leased line or radio modems, such as the 5902 Bell 202 modem.

- Click the **RTS/CTS Flow Control** radio button.
- Click the **RTS/CTS Hardware Control** radio button if this type of serial connection is to be used.
- Refer to the ***SCADAServer Host Reference*** section for complete information about RTS/CTS Flow Control settings.

Dial Up Modem

Dial-up connections are used to communicate with all the controllers in a pool using modems connected to dial up telephone lines. The Dial up type connection sends the configured modem initialization string and telephone number of the controller to the modem when it is selected.

- Click the **Dial Up** radio button if this type of serial connection is to be used.
- Refer to the ***SCADAServer Host Reference*** section for complete information about Dial Up modem settings.
- The **Phone Numbers** page defines the phone numbers for each station on a Dial Up serial port. Refer to the ***SCADAServer Host Reference*** section for complete information on how to enter telephone numbers. The edit boxes on this page are available only if Dial-Up is selected on the Type page.

Step 4: Configure Internet Client or Server Connection.

Only one connection of type Modbus/TCP Server, Modbus/UDP Server, Modbus RTU in TCP Server, Modbus RTU in UDP Server, Modbus ASCII in TCP Server and Modbus ASCII in UDP Server may be added to the connection list. Multiple connections of type Internet Client may be added to the list.

Refer to the ***Internet Client Properties Dialog*** or the ***Server Properties Dialog*** sections for more information on these connection types.

Internet Client Connection Properties

The **Access Path** edit-box specifies the Internet Client connection name. It is a 16-character string. For a Modbus/TCP connection to a physical device the Access Path is the connection name.

The **Connect to Host** edit-box specifies the IP services of controllers that SCADAServer wants to connect to or listen to. This can be a name, like "WORKSTATION123" or an IP address, like "123.45.67.89", where all of the numbers range from 0 to 255.

The **Addressing** edit box selects standard or extended Modbus addressing. Standard addressing allows 255 stations and is compatible with standard Modbus devices. Extended addressing allows 65534 stations, with stations 1 to 254 compatible with standard Modbus devices.

The Connection Type edit-box specifies the Internet Client connection. It is one of the following:

- Modbus/TCP
- Modbus/UDP
- Modbus RTU in TCP
- Modbus RTU in UDP
- Modbus ASCII in TCP
- Modbus ASCII in UDP

The **TCP/UDP Port** edit-box sets the port number of the slave that SCADAServer wants to connect to. It is an integer in the range 1 to 65535.

- Modbus/TCP recommended port 502
- Modbus/UDP recommended port 502
- Modbus RTU in TCP recommended port 49152

- Modbus RTU in UDP recommended port 49152
- Modbus ASCII in TCP recommended port 49153
- Modbus ASCII in UDP recommended port 49153

The **Time Out** edit-box sets the length of time, in seconds, to wait for a response to a command. It is an integer in the range 1 to 255 seconds. The default value is 3.

The **Attempts** edit-box sets the number of communication attempts before a message is aborted. It is an integer in the range 1 to 20. The default value is 3.

Internet Server Connection Properties

- The **Access Path** edit-box specifies the Server connection name. It is a 16-character string.
- The **Addressing** edit box selects standard or extended Modbus addressing. Standard addressing allows 255 stations and is compatible with standard Modbus devices. Extended addressing allows 65534 stations, with stations 1 to 254 compatible with standard Modbus devices.
- The **Port To Listen Number** edit-box sets port number of master that SCADAServer wants to listen to. It is an integer in the range 1 to 65535. The default value is 49152 for Modbus RTU in TCP or in UDP servers, 49153 for Modbus ASCII in TCP or in UDP servers and 502 for Modbus/TCP or Modbus/UDP servers.

Step 5: Configure Virtual PLC.

Virtual PLCs are an extension to the SCADAServer Host unique to Control Microsystems. The virtual PLCs allow the SCADAServer Host to act as slave device on Modbus networks.

- If a virtual PLC is to be configured select the **Virtual PLC** tab from the Properties dialog.
- In the **New Station** edit box enter the address of the virtual PLC.
- Click the **Add** button to add the new station address.
- If more than one connection is required, select **Add** and follow the previous steps to configure the next connection.
- Select **Ok** to close the Properties Dialog.
- Refer to the ***SCADAServer Host Reference*** section of this manual for complete details on using and configuring virtual PLCs.

Step 6: Close Connection Properties Dialog.

- When the Properties for all connections have been completed click the **Ok** button in the Properties dialog.
- If another connection is required repeat steps 2, 3, 4 and 5 as required.
- Once the properties for all connections have been completed click the **Ok** button in the Connections dialog.

Step 7: Configure SCADAServer Host Options.

- Select **Options** from the Server menu.

- If desired, the SCADAServer Host can be configured to be minimized on start-up, to save Virtual PLC registers on shutdown and restored on startup, to automatically open a specified configuration file upon start-up and to log events to a specified text file.
- Refer to the ***SCADAServer Host Reference*** section for complete details on setting SCADAServer options.
- Select **Ok** to close the dialog.

Step 8: Configure Data Views.

The View commands menu contains commands to display the event Log, polling Pools and communication Statistics and to control the display of both the Toolbar and the Status bar.

- Select **Statistics** from the **View** menu.
- Select **New Window** from the **Window** menu.
- Select **Pools** from the **View** menu.
- Select **New Window** from the **Window** menu.
- Select **Log** from the **View** menu.
- Select **Tile** from the **Window** menu.
- Refer to the ***SCADAServer Host Reference*** section for complete details on configuring SCADAServer data views.

Step 9: Save SCADAServer Host Configuration.

- Select **Save As** from the File menu to save the SCADAServer Host configuration file.
- If the server was configured to automatically open a specified configuration file upon start-up, save the configuration file to the filename specified in the Options dialog.
- Select **Exit** from the **File** menu to exit the SCADAServer Host.

SCADAServer Virtual PLC

Virtual PLCs are an extension to the SCADAServer Host unique to Control Microsystems. The virtual PLCs allow the SCADAServer Host to act as slave device on Modbus networks. This can be used to implement report by exception operation.

The SCADAServer Host can be configured to provide one or more virtual PLCs on each communication network. Multiple virtual PLCs can simplify the programming in report by exception systems. The program in each field controller can use the same Modbus registers in the virtual PLC. Only the station number to which it reports has to change.

Each virtual PLC appears to Modbus devices on the network as a normal Modbus slave device. Each virtual PLC has the following registers. Memory for the registers is allocated when the virtual PLC is first created and may be optionally saved on shutdown and restored on subsequent startups.

Type	***Range***	***Properties***
Discrete Output	00001 to 09999	single bit read/write
Discrete Input	10001 to 19999	single bit read only
Analog Input	30001 to 39999	16-bit or 32-bit read only
Analog Output	40001 to 49999	16-bit or 32-bit read/write

Virtual PLCs are unique to the serial port or server connection for which they are created. A virtual PLC cannot be shared by more than one connection. Multiple devices may communicate with a single virtual PLC on the connection.

Each virtual PLC has a station address, which is set using the server user interface. The user must ensure that the station numbers of the virtual PLCs are unique for the network to which they are connected.

The virtual PLCs appear to the SCADAServer Host as a controller. OPC clients access the virtual PLCs in the same manner as all other controllers. The Access Path for the OPC item determines which connection's virtual PLC is being addressed.

SCADAServer Address Space

The SCADAServer address space consists of all Modbus registers in all controllers that can be accessed by the server, on all communication connections. An Item ID and optional Access Path specify the registers associated with each item in a group.

Item ID

The Item ID is the unique identifier of each variable in the server address space. The Item ID specifies the controller station number, the register address and the register format. Strings in the following form describe the Item ID. Note that the format and access path parameters are optional.

```
station:register:[format][@accessPath]
```

The server supports standard and extended Modbus addressing. If standard addressing is configured the server allows station numbers from 1 to 255. If extended addressing is configured the server allows station numbers from 1 to 65534.

Each station can have the following register addresses.

Type	Range	Properties
Discrete Output	00001 to 09999	single bit read/write
Discrete Input	10001 to 19999	single bit read only
Analog Input	30001 to 39999	16-bit or 32-bit read only
Analog Output	40001 to 49999	16-bit or 32-bit read/write

The optional format parameter defines how input and holding registers are read. Each analog register in the controller address space is a 16-bit register. The server allows two consecutive registers to be read as a single 32-bit value. This provides maximum flexibility for OPC clients.

Format	Registers	Notes
I	1	16-bit integer (default if no format specified)
U	1	16-bit unsigned integer
D	2	32-bit integer (low word in first register)
L	2	32-bit integer (high word in first register)
UD	2	32-bit unsigned integer (low word in first register)
UL	2	32-bit unsigned integer (high word in first register)
F	2	IEEE 754 single precision floating-point

The optional Access Path parameter is used with OPC Clients that don't support access paths. In most cases it is not required (see below for details). If the parameter is used, add the character @ followed by the Access Path to the end of the Item ID.

Access Path

The optional Access Path parameter specifies the communication connection. It specifies a serial port or a Modbus/TCP device where the controller is connected.

The Access Path is a string. The string displayed for a serial port connection is one of `COMX` (where `X` is the comport's number) or `Modbus/TCPY` (where `Y` is the access path).

There are two ways to specify the Access Path.

- Most OPC Clients support Access Paths. Enter the Access Path parameter in the appropriate place. See you OPC Client documentation for details. Do not add the access path to the Item ID.
- Some OPC Clients do not support Access Paths. If you are using such a program, add the Access Path to the end of the item ID.

SCADAServer Host Reference

This section of the user manual contains detailed explanations for all the commands and functions described in the following section. In addition to this, the on-line help for the SCADAServer Host program contains a complete reference for the program.

SCADAServer Host Display Window

The SCADAServer Host display window is divided into the following areas. Each area is described in the following sections of this manual.

The ***Title Bar*** is located along the top of the SCADAServer Host window and contains configuration file information and window control functions.

The ***Menu Bar*** provides access to all SCADAServer Host commands. This reference contains complete information on all SCADAServer Host menu commands.

The ***Tool Bar*** is located below the Menu Bar and provides quick mouse access to SCADAServer Host functions.

The **Status Bar** is displayed across the bottom of the application window and describes actions of menu items as you use the arrow keys or mouse to navigate through menus.

Title Bar

The title bar is located along the top of a window. It contains the name of the application, SCADAServer Host, and the currently opened SCADAServer Host configuration file. The title bar provides commands for control of the opened application and the window display.

To move the window, drag the title bar. You can also move dialog boxes by dragging their title bars.

The application control menu button is the SCADAServer Host icon in the upper left corner of the SCADAServer Host window. When selected the following commands are displayed.

Restore Command

Use this command to return the active window to its size and position before you chose the Maximize or Minimize command.

Move Command

Use this command to display a four-headed arrow so you can move the active window or dialog box with the arrow keys. Note that this command is unavailable if you maximize the window. Using the mouse drag the window title bar to location required.

Size Command

Use this command to display a four-headed arrow so you can size the active window with the arrow keys. This command is unavailable if you maximize the window. Use the mouse to drag the size bars at the corners or edges of the window.

Minimize Command

Use this command to reduce the SCADAServer Host window or the view window to an icon. Use the mouse by clicking the minimize icon on the title bar.

Maximize Command

Use this command to enlarge the active window to fill the available space. Use the mouse by clicking the maximize icon on the title bar; or double-click the title bar.

Close Command

Use this command to close the active window or dialog box. Double-clicking a Control-menu box is the same as choosing the Close command. If you have multiple windows open for a single document, the Close command on the document Control menu closes only one window at a time. You can close all windows at once with the Close command on the File menu. Keyboard keys CTRL+F4 closes a document window and ALT+F4 closes the SCADAServer Host window or dialog box.

Tool Bar

The toolbar is displayed across the top of the application window, below the menu bar. The toolbar provides quick mouse access to many tools used in the SCADAServer Host.

To hide or display the Toolbar, choose **Toolbar** from the **View** menu. A check mark preceding **Toolbar** indicates that the Tool bar is already displayed.

The following commands and functions are displayed on the Toolbar.

Create a new configuration file.

Open an existing SCADAServer Host configuration file. The SCADAServer Host displays the Open dialog box, in which you can locate and open the desired configuration file.

Save the active configuration to a file. If an existing configuration file is open, the SCADAServer Host will overwrite the configuration file with the current configuration. If the file has not yet been saved, the SCADAServer Host displays the Save As dialog box, in which you can name and save the configuration to a file.

The Log command displays the event log in the current view. The column widths are set to the most recently set values for the Log view.

The Pools command displays the polling pools in the current view. The column widths are set to the most recently set values for the Pools view.

The Statistics command displays the communication statistics in the current view. The column widths are set to the most recently set values for the Statistics view.

The toolbar may:

- remain stationary along one side of its parent window;
- be dragged and docked, or attached, by the user to any side or sides of the parent window you specify;
- be floated, or detached from the frame window, in its own mini-frame window so the user can move it around to any convenient position; and
- be re-sized while floating.

To move the toolbar, click on the background of the toolbar. Drag the toolbar to the new location and release the mouse button.

Status Bar

The Status Bar is displayed across the bottom of the application window. The left area of the status bar describes actions of menu items as you use the arrow keys to navigate through menus. It also shows messages that describe the actions of toolbar buttons as you depress them, before releasing them. If after viewing the description of the toolbar button command you wish not to execute the command, then release the mouse button while the pointer is off the toolbar button.

The right areas of the status bar indicate which of the following keys are latched down:

CAP Indicates the Caps Lock key is latched down.

NUM Indicates the Num Lock key is latched down.

To hide or display the Status Bar, choose **Status Bar** from the **View** menu. A check mark preceding **Status Bar** indicates that the status bar is already displayed.

Menu Bar

The menu bar displays the configuration file management, view, SCADAServer Host configuration, window management and help functions available with the SCADAServer Host. Menu commands are displayed by clicking the mouse button on the menu item or by pressing the alt key and the underlined letter of the menu item.

File Menu Commands

The **File** commands menu contains commands to create, open and save SCADAServer Host configuration files and to exit the SCADAServer Host application.

New Command

Use this command to create a new SCADAServer Host configuration file. When a new file is started no connections are created, all the options on the Options dialog are not checked and all file names are blank.

Open Command

Use this command to open an existing SCADAServer Host configuration file. The SCADAServer Host is reconfigured to use the settings contained in the file. When the **Open** command is used the Open dialog is displayed. The following options allow you to specify which file to open.

The **Look in:** box lists the available folders and files.

The **File name:** box allows you to type or select the filename you want to open. This box lists files with the extension you select in the Files of Type box.

The **Files of type:** box lists the types of files SCADAServer Host can open. SCADAServer Host can open existing SCADAServer Host configuration (SSC) files.

The file name is displayed in the Title Bar.

Save Command

Use this command to save the current SCADAServer Host configuration to a file. If an existing file is open, the command will write the active configuration parameters to this file. If a file is not open or the configuration has not yet been saved to a file, the command will open the Save As dialog. See below for details.

Save As Command

Use this command to name and save the SCADAServer Host configuration file. When the **Save As** command is used the Save As dialog is displayed. The following options allow you to specify the file name and location:

The **Save in:** box lists the available folders and files.

The **File name:** box allows entry of a new filename to save the file with a different name. The SCADAServer Host automatically adds the extension SSC to the file name.

The **Save as type:** box lists the types of files the SCADAServer Host can save. The SCADAServer Host can save files as OPC Server Configuration (SSC) files only.

The **Save** button saves the file to the specified location.

The **Cancel** button closes the dialog without saving.

Exit Command

Use this command to exit the SCADAServer Host application.

View Menu Commands

The **View** commands menu contains commands to display the event Log, polling Pools and communication Statistics and to control the display of both the Toolbar and the Status bar.

Log Command

Use this command to display the event Log in the current view. The log view displays error and status messages. The view is a table. Each row in the table shows one message. Data in this view is updated when new error, warning or status messages are generated.

The columns in the table are listed below.

The ***Time*** column displays the date and time of the event occurrence in the short format defined from the Windows Control Panel.

The ***Severity*** column displays the severity of the message. It is one of error, warning or status.

The ***Message*** column displays the text of the message.

Sorting Data

Clicking on the column headings will sort the table, in ascending order, according to the column selected. Clicking subsequent times on the same column toggles the sort order between descending and ascending order.

Sizing Columns

Positioning the cursor over the line separating the columns in the heading can change the column widths. The cursor changes to a vertical bar with arrows pointing left and right. Click on the line, then slide the mouse left or right. Release the mouse button when the column is the desired width.

Pools Command

Use this command to display the polling Pools in the current view. The Pools view displays how the data is being polled on each connection.

The SCADAServer Host organizes data to be polled on each connection into blocks (pools) so the polling is as efficient as possible. Each block is called a pool. This view allows the user to diagnose the polling of data.

The view is a table. Each row in the table shows one message. The columns in the table are as follows.

The ***Connection*** column displays the name of the connection (e.g. com1, com2).

The ***Station*** column displays the station number of the device.

The ***First Register*** column displays the first register in the pool.

The ***Last Register*** column displays the last register in the pool.

The ***Update Rate*** column displays the desired update rate for the pool.

Sizing Columns

Positioning the cursor over the line separating the columns in the heading can change the column widths. The cursor changes to a vertical bar with arrows pointing left and right. Click on the line, then slide the mouse left or right. Release the mouse button when the column is the desired width.

Statistics Command

Use this command to display the communication Statistics in the current view. The Statistics view displays the communication statistics for each station on each connection. This view allows the user to diagnose the polling of data.

The view is a table. Each row in the table shows one message. The columns in the table are as follows.

The ***Connection*** column displays name of the connection (e.g. com1, com2).

The ***Station*** column displays the station number of the device.

The ***Total*** column displays the total number of messages sent to the station.

The ***Failed*** column displays the number of failed messages where an error occurred.

The ***% Failed*** column displays the percentage of failed messages rounded to the nearest tenth of a percent.

Sorting Data

Clicking on the column headings will sort the table, in ascending order, according to the column selected. Clicking subsequent times on the same column toggles the sort order between descending and ascending order.

Sizing Columns

Positioning the cursor over the line separating the columns in the heading can change the column widths. The cursor changes to a vertical bar with arrows pointing left and right. Click on the line, then slide the mouse left or right. Release the mouse button when the column is the desired width.

Toolbar Command

Use this command to display or hide the Toolbar. Selecting this command toggles the toolbar visibility.

A check mark preceding **Toolbar** indicates that the Toolbar is displayed.

Status Bar Command

Use this command to display or hide the Status bar. Selecting this command toggles the status bar visibility.

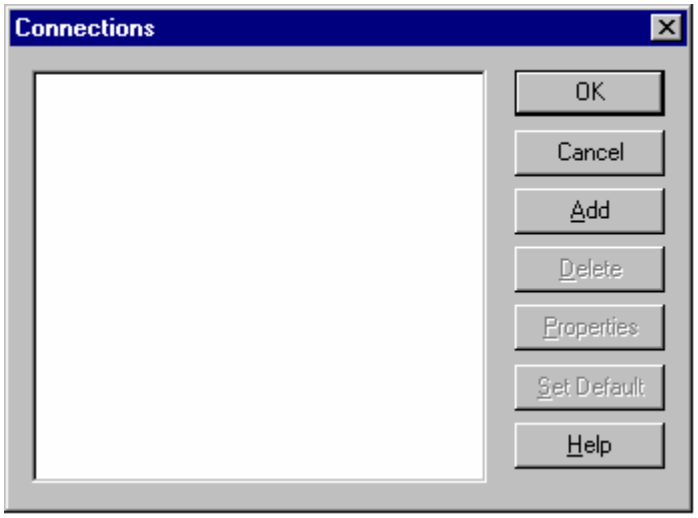
A check mark preceding **Status bar** indicates that the Status bar is displayed.

Server Menu Commands

The **Server** commands menu contains commands to edit the server configuration, specify server start-up and event logging options, and clear the event log view and communication statistics.

Connections Command

Use this command to edit the SCADAServer Host configuration. When selected this command opens the Connections dialog.



The Connections dialog specifies the communication connections the SCADAServer Host makes. The SCADAServer Host communicates with controllers using these connections. When a connection added the connection type and access path is displayed in the list.

The **OK** button closes the dialog and saves any changes. This is the default button.

The **Cancel** button closes the dialog and discards any changes.

The **Add** button opens the Add Connection dialog (see below) to add a new connection to the list.

The **Delete** button deletes the currently selected connection from the list. This button is grayed if no connection is selected.

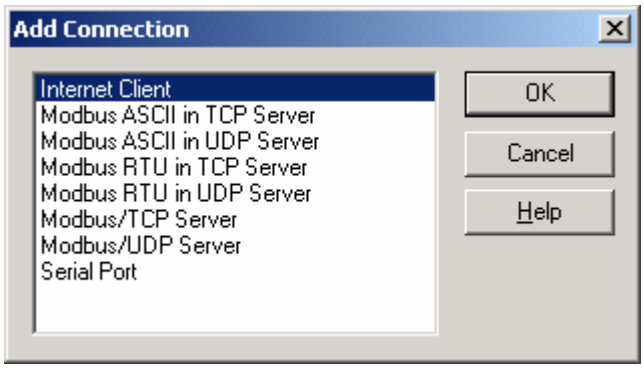
The **Properties** button opens the Connect Properties dialog for the currently selected connection. This button is grayed if no connection is selected.

The **Set Default** button makes the currently selected connection the default connection. The default connection will appear in bold type. This connection is used if an OPC Client item's Access Path is blank or set to default. This button is grayed if no connection is selected.

The **Help** button displays the on-line help for the dialog.

Add Connection Dialog

The Add Connection dialog specifies the type of connection to add to the Connections list.



The list box shows the available types of connections. The available connections are:

- Internet Client
- Modbus ASCII in TCP Server
- Modbus ASCII in UDP Server
- Modbus RTU in TCP Server
- Modbus RTU in UDP Server
- Modbus/TCP Server
- Modbus/UDP Server
- Serial Port

One connection of type Modbus/TCP Server, Modbus/UDP Server, Modbus RTU in TCP Server, Modbus RTU in UDP Server, Modbus ASCII in TCP Server, Modbus ASCII in UDP Server and Serial Port may be added to the connection list.

Multiple connections of type Internet Client may be added to the list.

Multiple connections of type Serial Port, to the maximum number of serial ports on the PC, may be added to the list.

The **OK** button accepts the selected connection and adds it to the connection list. The dialog is then closed.

The **Cancel** button closes the dialog without making any changes.

The **Help** button displays the on-line help for the dialog.

Internet Client Properties Dialog

The Server Properties dialog edits the settings for the following Internet Client connections:

- Modbus/TCP
- Modbus/UDP
- Modbus RTU in TCP
- Modbus RTU in UDP
- Modbus ASCII in TCP
- Modbus ASCII in UDP

Modbus/TCP is an extension of serial Modbus, which defines how Modbus messages are encoded within and transported over TCP/IP-based networks. The Modbus/TCP protocol uses a custom Modbus protocol layer on top of the TCP protocol. Its request and response messages are prefixed by six bytes. These six bytes consist of three fields: transaction ID field, protocol ID field and length field. The encapsulated Modbus message has exactly the same layout and meaning, from the function code to the end of the data portion, as other Modbus messages. The Modbus 'CRC-16' or 'LRC' check fields are not used in Modbus/TCP. The TCP/IP and link layer (e.g. Ethernet) checksum mechanisms instead are used to verify accurate delivery of the packet.

Modbus/UDP communication mode is similar to Modbus/TCP communication mode. It has the same message format with the Modbus/TCP. The only difference between them is one uses TCP protocol and another uses UDP protocol.

Modbus RTU in TCP message format is exactly same as that of the Modbus RTU compatible TeleBUS RTU protocol. The main difference is that Modbus RTU in TCP protocol communicates with a controller through the Internet and TeleBUS protocol through the serial port. The Modbus RTU in TCP protocol does not include a six-byte header prefix, as with the Modbus\TCP, but does include the Modbus 'CRC-16' or 'LRC' check fields. The Modbus RTU in TCP message format supports Modbus RTU message format.

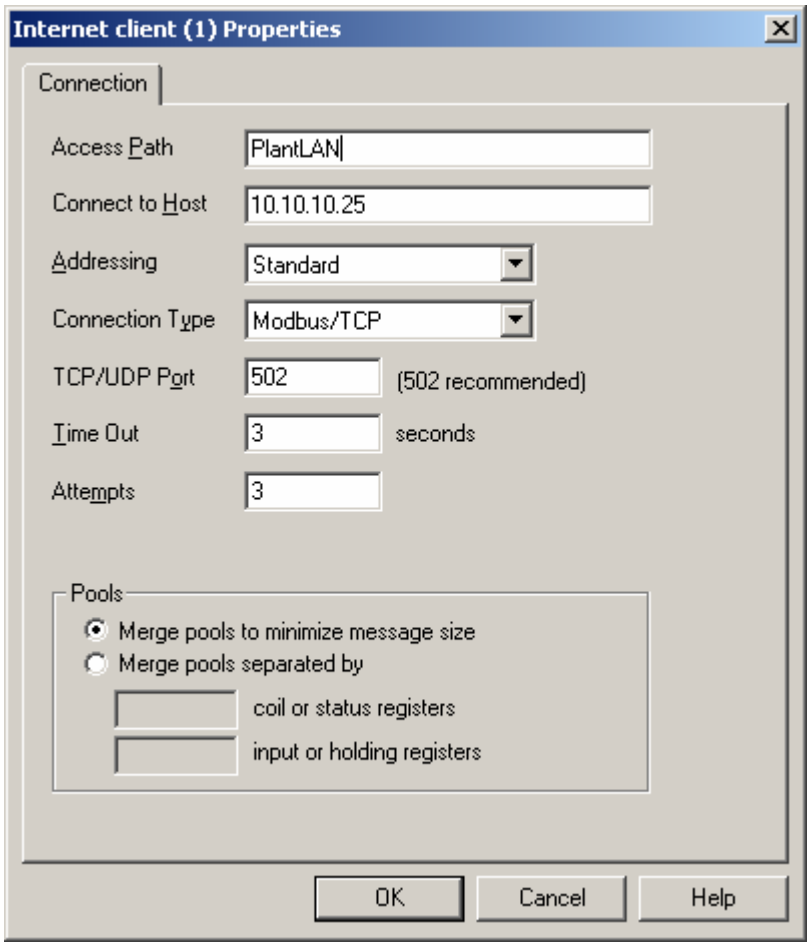
Modbus RTU in UDP communication mode is similar to Modbus RTU in TCP communication mode. It has the same message format as the RTU in TCP message. The only difference between them is one uses TCP protocol and another uses UDP protocol.

Modbus ASCII in TCP message format is exactly same as that of the Modbus ASCII compatible TeleBUS ASCII protocol. The main difference is that Modbus ASCII in TCP protocol communicates with a controller through the Internet and TeleBUS protocol through the serial port. The Modbus ASCII in TCP protocol does not include a six-byte header prefix, as with the Modbus\TCP, but does include the Modbus 'CRC-16' or 'LRC' check fields. The Modbus ASCII in TCP message format supports Modbus ASCII message format.

Modbus ASCII in UDP communication mode is similar to Modbus ASCII TCP communication mode. It has the same message format as the Modbus ASCII in TCP. The only difference between them is one uses TCP protocol and another uses UDP protocol.

Internet Client Connection Property Page

The dialog is a tabbed property sheet. Clicking on a tab displays the selected property page. An Internet Connection displays the ***Connection*** page.



The **Access Path** edit-box specifies the Internet Client connection name. It is a 16-character string. For a Modbus/TCP connection to a physical device the Access Path is the connection name.

The **Connect to Host** edit-box specifies the IP services of controllers that SCADAServer wants to connect to or listen to. This can be a name, like "WORKSTATION123" or an IP address, like "123.45.67.89", where all of the numbers range from 0 to 255.

The **Addressing** edit box selects standard or extended Modbus addressing. Standard addressing allows 255 stations and is compatible with standard Modbus devices. Extended addressing allows 65534 stations, with stations 1 to 254 compatible with standard Modbus devices.

The Connection Type edit-box specifies the Internet Client connection. It is one of the following:

- Modbus/TCP
- Modbus/UDP
- Modbus RTU in TCP
- Modbus RTU in UDP
- Modbus ASCII in TCP
- Modbus ASCII in UDP

The **TCP/UDP Port** edit-box sets the port number of the slave that SCADAServer wants to connect to. It is an integer in the range 1 to 65535.

- Modbus/TCP recommended port 502
- Modbus/UDP recommended port 502
- Modbus RTU in TCP recommended port 49152
- Modbus RTU in UDP recommended port 49152
- Modbus ASCII in TCP recommended port 49153
- Modbus ASCII in UDP recommended port 49153

The **Time Out** edit-box sets the length of time, in seconds, to wait for a response to a command. It is an integer in the range 1 to 255 seconds. The default value is 3.

The **Attempts** edit-box sets the number of communication attempts before a message is aborted. It is an integer in the range 1 to 20. The default value is 3.

The **Pools** area of the dialog provides control of how pools are merged. SCADAServer Host organizes data to be polled on each connection into blocks of data called pools. A pool contains sequential Modbus registers of a specific data type; coil status, input status, holding register and input register. SCADAServer updates the data in the pools by using standard Modbus commands to read the data. The idea of merging pools is that it may be more efficient to merge non-sequential pools together into one poll command that includes registers in the gap between the non-sequential pools rather than send two Modbus commands. The following table shows the maximum gap allowed based on connection type.

Connection Type	Register type	Maximum Gap
Modbus TCP	Coil status, Input status	1046
	Holding register, Input register	64
Modbus UDP	Coil status, Input status	854
	Holding register, Input register	52
Modbus RTU in TCP	Coil status, Input status	1030
	Holding register, Input register	63
Modbus RTU in UDP	Coil status, Input status	790
	Holding register, Input register	48
Modbus ASCII in TCP	Coil status, Input status	550
	Holding register, Input register	33
Modbus ASCII in UDP	Coil status, Input status	454
	Holding register, Input register	27

The **Merge pools to minimize message size** radio button automatically selects the gap size using the table above.

The **Merge pools separated by** radio button enables entry of user defined gap size.

- For **coil or status registers** valid entries are from 0 to 2000.
- For **input and holding registers** valid entries are from 0 to 125.

The Pools setting is applied, and the pools adjusted, when items are added to the pools or removed from them.

Server Properties Dialog

The Server Properties dialog edits the settings for the following server connections:

- Modbus ASCII in TCP Server
- Modbus ASCII in UDP Server

- Modbus RTU in TCP Server
- Modbus RTU in UDP Server
- Modbus/TCP Server
- Modbus/UDP Server

Modbus ASCII in TCP message format is exactly same as that of the Modbus ASCII compatible TeleBUS ASCII protocol. The main difference is that Modbus ASCII in TCP protocol communicates with a controller through the Internet and TeleBUS protocol through the serial port. The Modbus ASCII in TCP protocol does not include a six-byte header prefix, as with the Modbus\TCP, but does include the Modbus 'CRC-16' or 'LRC' check fields. The Modbus ASCII in TCP message format supports Modbus ASCII message format.

Modbus ASCII in UDP communication mode is similar to Modbus ASCII TCP communication mode. It has the same message format as the Modbus ASCII in TCP. The only difference between them is one uses TCP protocol and another uses UDP protocol.

Modbus RTU in TCP message format is exactly same as that of the Modbus RTU compatible TeleBUS RTU protocol. The main difference is that Modbus RTU in TCP protocol communicates with a controller through the Internet and TeleBUS protocol through the serial port. The Modbus RTU in TCP protocol does not include a six-byte header prefix, as with the Modbus\TCP, but does include the Modbus 'CRC-16' or 'LRC' check fields. The Modbus RTU in TCP message format supports Modbus RTU message format.

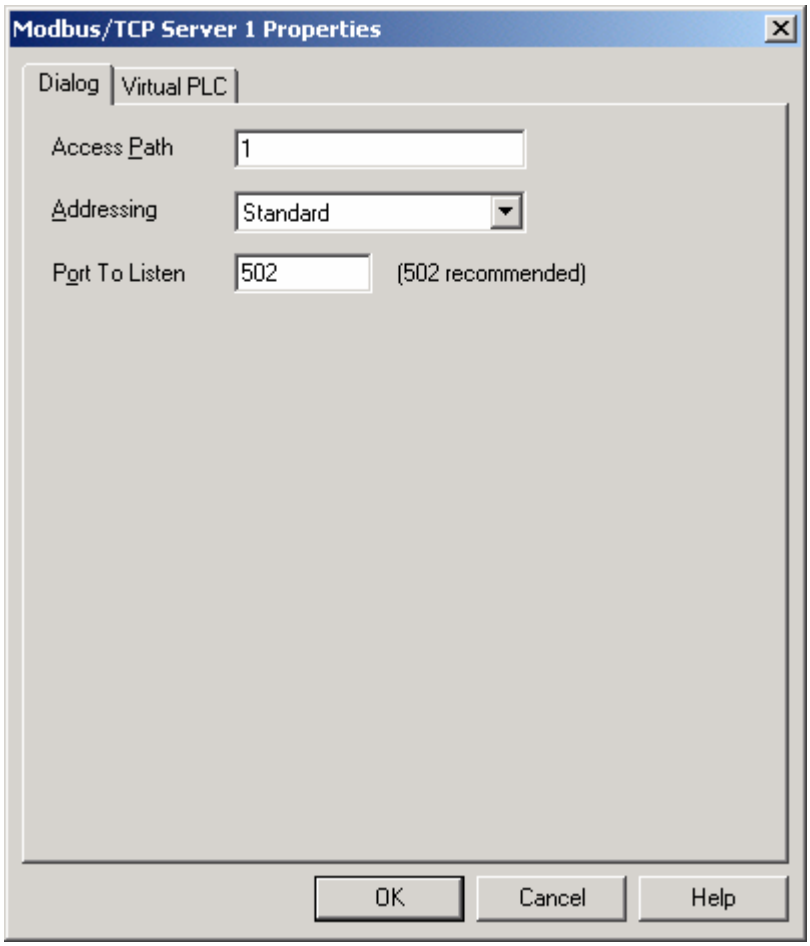
Modbus RTU in UDP communication mode is similar to Modbus RTU in TCP communication mode. It has the same message format as the RTU in TCP message. The only difference between them is one uses TCP protocol and another uses UDP protocol.

Modbus/TCP is an extension of serial Modbus, which defines how Modbus messages are encoded within and transported over TCP/IP-based networks. The Modbus/TCP protocol uses a custom Modbus protocol layer on top of the TCP protocol. Its request and response messages are prefixed by six bytes. These six bytes consist of three fields: transaction ID field, protocol ID field and length field. The encapsulated Modbus message has exactly the same layout and meaning, from the function code to the end of the data portion, as other Modbus messages. The Modbus 'CRC-16' or 'LRC' check fields are not used in Modbus/TCP. The TCP/IP and link layer (e.g. Ethernet) checksum mechanisms instead are used to verify accurate delivery of the packet.

Modbus/UDP communication mode is similar to Modbus/TCP communication mode. It has the same message format with the Modbus/TCP. The only difference between them is one uses TCP protocol and another uses UDP protocol.

Server Connection Property Page

The dialog is a tabbed property sheet. Clicking on a tab displays the selected property page. A Modbus/TCP connection displays the ***Dialog***, and ***Virtual PLC*** pages.



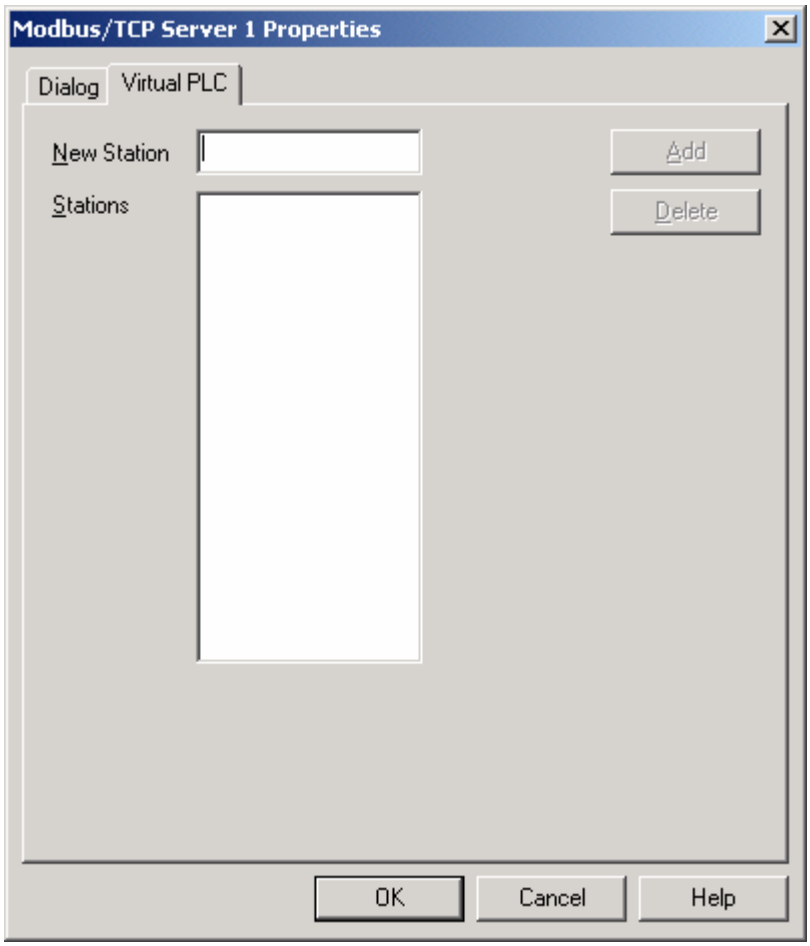
The **Access Path** edit-box specifies the Server connection name. It is a 16-character string.

The **Addressing** edit box selects standard or extended Modbus addressing. Standard addressing allows 255 stations and is compatible with standard Modbus devices. Extended addressing allows 65534 stations, with stations 1 to 254 compatible with standard Modbus devices.

The **Port To Listen Number** edit-box sets port number of master that SCADAServer wants to listen to. It is an integer in the range 1 to 65535. The default value is 49152 for Modbus RTU in TCP or in UDP servers, 49153 for Modbus ASCII in TCP or in UDP servers and 502 for Modbus/TCP or Modbus/UDP servers.

Server Connection Properties Virtual PLC Page

The Server Properties Virtual PLC page defines the virtual PLCs that are provided by the connection.



The **New Station** edit-box specifies the station number to add. The valid range is 1 to 255 if standard addressing is configured and 1 to 65534 if extended addressing is configured.

The **Add** button adds the station to the list. Duplicate addresses will not be accepted.

The **Delete** button removes the selected station from the list. This button is grayed if no station is selected.

Serial Port Properties Dialog

The Serial Port Properties dialog edits the settings for a Serial Port connection. The dialog is a tabbed property sheet. Clicking on a tab displays the selected property page. A serial port connection displays the ***Port***, ***Type, Phone Number*** and ***Virtual PLC*** pages.

The **OK** button closes the dialog and saves any changes. This is the default button.

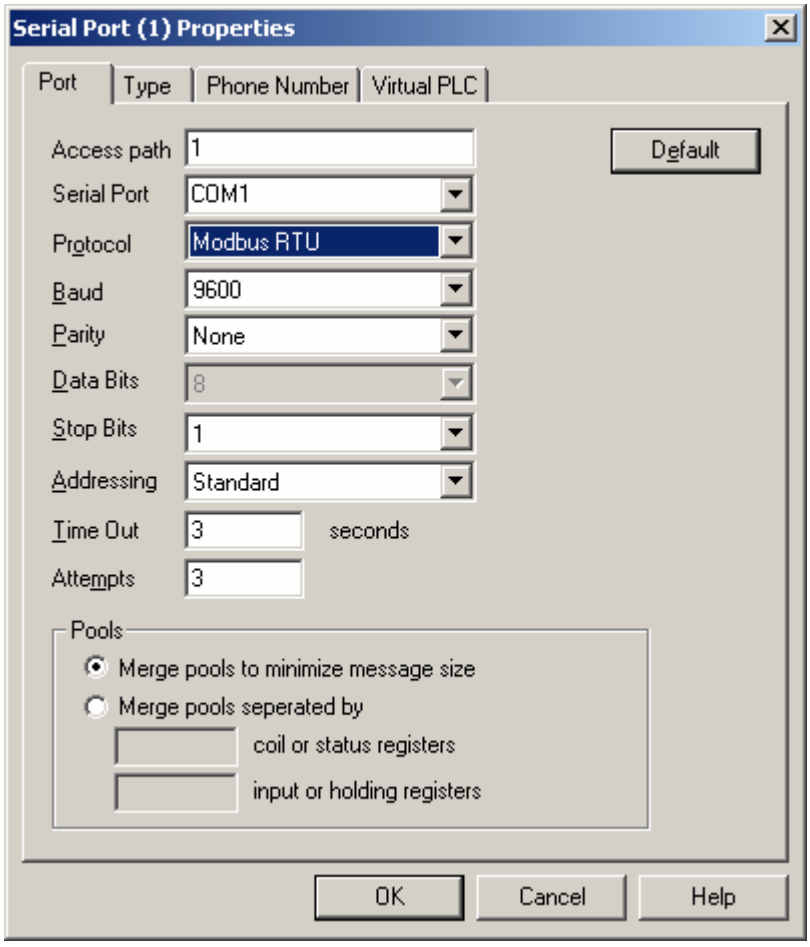
The **Cancel** button closes the dialog and discards any changes.

The **Help** button displays the on-line help for the dialog.

The dialog title shows the name of the connection being edited.

Serial Port Properties Port Page

The Serial Port Properties Port page defines the serial port communication parameters.



The **Access path** combo-box defines the connection name. This is a string with a maximum size of 16-characters.

The **Serial Port** combo box selects the serial port on the PC to use for the connection.

The **Protocol** combo box selects the communication protocol. It is one of Modbus RTU or Modbus ASCII. The default value is Modbus RTU.

The **Baud** combo box selects the baud rate. It is one of 300, 600, 1200, 2400, 4800, 9600, 19200, 38400 or 57600. The default value is 9600 baud.

The **Parity** combo box selects the number of parity bits. It is one of none, odd or even. The default value is none.

The **Data bits** combo box selects the number of data bits. It is one of 7 or 8. The default value is 8. If the protocol is Modbus RTU, this value is forced to 8 and the edit box is grayed.

The **Stop bits** combo box selects the number of stop bits. It is one of 1 or 2. The default value is 1.

The **Addressing** combo box selects the type of Modbus addressing. It is one of standard or extended. The default value is standard.

The **Time Out** edit-box sets the length of time, in seconds, to wait for a response to a command. It is an integer in the range 1 to 255 seconds. The default value is 3.

The **Attempts** edit box sets the number of communication attempts before a message is aborted. It is an integer in the range 1 to 20. The default value is 3.

The **Pools** area of the dialog provides control of how pools are merged. SCADAServer Host organizes data to be polled on each connection into blocks of data called pools. A pool contains sequential Modbus registers of a specific data type; coil status, input status, holding register and input register. SCADAServer updates the data in the pools by using standard Modbus commands to read the data. The idea of merging pools is that it may be more efficient to merge non-sequential pools together into one poll command that includes registers in the gap between the non-sequential pools rather than send two Modbus commands. The following table shows the maximum gap allowed based on connection type.

Connection Type	Register type	Maximum Gap
Modbus RTU	Coil status, Input status	118
	Holding register, Input register	6
Modbus ASCII	Coil status, Input status	122
	Holding register, Input register	6

The **Merge pools to minimize message size** radio button automatically selects the gap size using the table above.

The **Merge pools separated by** radio button enables entry of user defined gap size.

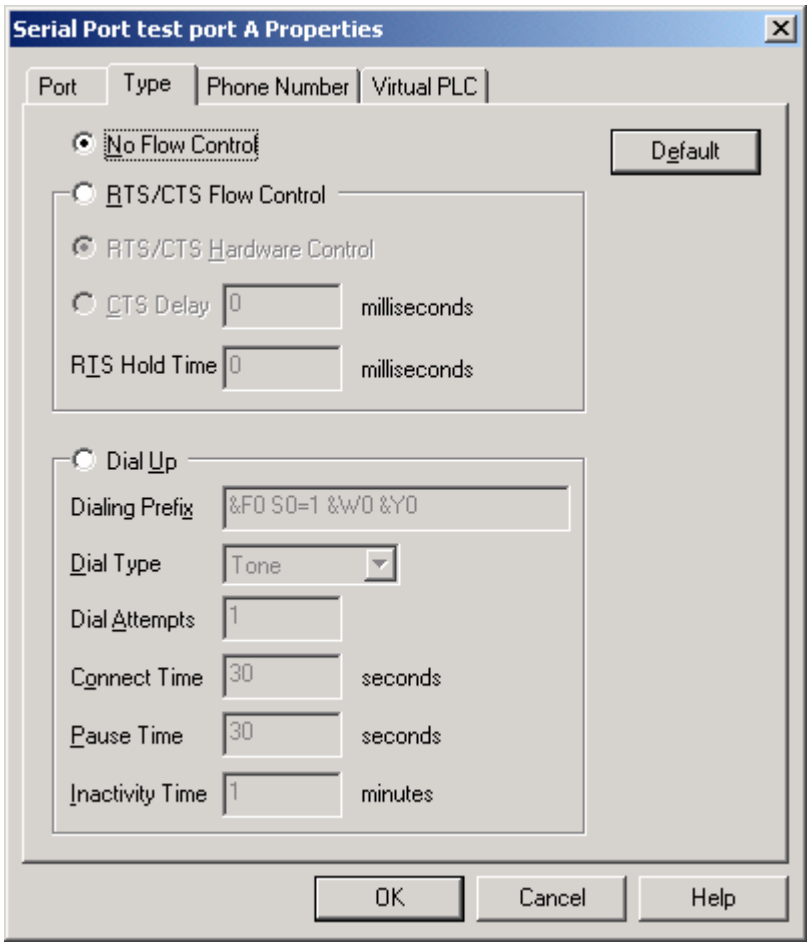
- For **coil or status registers** valid entries are from 0 to 2000.
- For **input and holding registers** valid entries are from 0 to 125.

The Pools setting is applied, and the pools adjusted, when items are added to the pools or removed from them.

The **Default** button sets all parameters to default values.

Serial Port Properties Type Page

The Serial Port Properties Type page defines the type of flow control needed for the network to be connected to the serial port.



The **No Flow Control** radio button specifies an RS-232 or RS-485 connection that does not require the hardware control lines on the serial ports. This is the default connection.

The **RTS/CTS Flow Control** radio button specifies a half-duplex connection requiring the use of the Request to Send (RTS) and Clear to Send (CTS) hardware control lines to control the flow of data. This connection is used with radios and dedicated telephone line modems.

When configured for RTS/CTS handshaking, the SCADAServer Host turns on the RTS signal when it wants to transmit. The modem or other device then turns on CTS when it is ready to transmit. SCADAServer Host transmits the data, then turns off the RTS signal.

All controls in the section are grayed if the *RTS/CTS Flow Control* radio button is not selected.

- The **RTS/CTS Hardware Control** radio button is used with a device that generates a CTS signal in response to the RTS signal. The SCADAServer Host will wait for CTS before proceeding.
- The **CTS Delay Time** radio button is used with a device that cannot generate a CTS signal. The SCADAServer Host will assert RTS then wait the specified delay time before proceeding.

- The **RTS Hold Time** edit box is used to specify the delay time if the device ends transmission immediately when RTS is turned off, and you need a delay after transmitting the last character. The SCADAServer Host holds RTS on for this time after the last character is sent.

The **Dial-Up** radio button specifies a dial-up modem connection. These modems are used on the public switched telephone network. The SCADAServer Host connects to each controller as required by the update rate for the items in the controller. Once a connection to the controller is made the SCADAServer Host reads or writes all items for the controller, regardless of their update rate. This is done because making the connection to the controller takes much more time than communication generally. The connection can be maintained or terminated if a single phone connection has been configured. If multiple phone connections have been configured, the SCADAServer Host will disconnect, after polling the controller, and establish the next configured connection.

The dial-up modem connection is capable of accepting incoming calls. This is used in report-by-exception systems. When the phone rings, the SCADAServer Host answers the call and waits for incoming Modbus messages. These messages are directed to the virtual PLCs in the server. The dial-up modem connection will not accept incoming calls if there are no virtual PLCs created in the server.

Note:	The SCADAServer Host does not support Dial-up connections for TelePACE, TelePACE Firmware Loader, ISaGRAF, SCADALog and RealFLO clients.

The **Dialing Prefix** edit box specifies the commands sent to the modem before dialing. This is a 32-character string. All characters are valid.

The **Dial Type** combo box specifies the dialing type. Valid values are Pulse and Tone. The default value is Tone.

The **Dial Attempts** edit box specifies how many dialing attempts will be made. Valid values are 1 to 10. The default value is 1.

The **Connect Time** edit box specifies the time, in seconds, the modem will wait for a connection. Valid values are 6 to 300. The default value is 30.

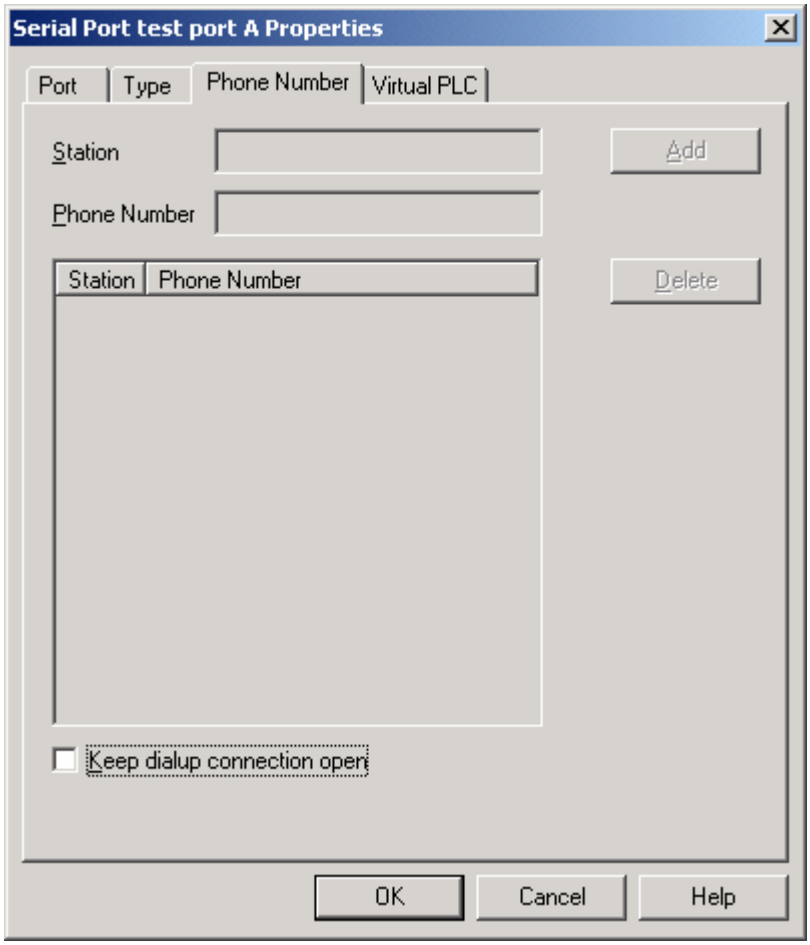
The **Pause Time** edit box specifies the time, in seconds, between dialing attempts. Valid values are 6 to 600. The default value is 30

The SCADAServer Host will automatically terminate an incoming connection after a period of inactivity. The **Inactivity Time** edit box specifies the time, in seconds, after which the connection will be terminated. Activity is defined as valid communication messages being received on the port. The range is 1 to 255 seconds. The default value is 1.

The **Default** button sets all parameters to default values.

Serial Port Properties Phone Number Page

The Serial Port Properties Phone Number page defines the phone numbers for each station on a serial port. The edit boxes on this page are available only if Dial-Up is selected on the Type page.



The **Station** edit box specifies the station number to add. The valid range is 1 to 255 if standard addressing is configured and 1 to 65534 if extended addressing is configured.

The **Phone Number** edit box specifies the phone number string to add. The string is 32 characters long.

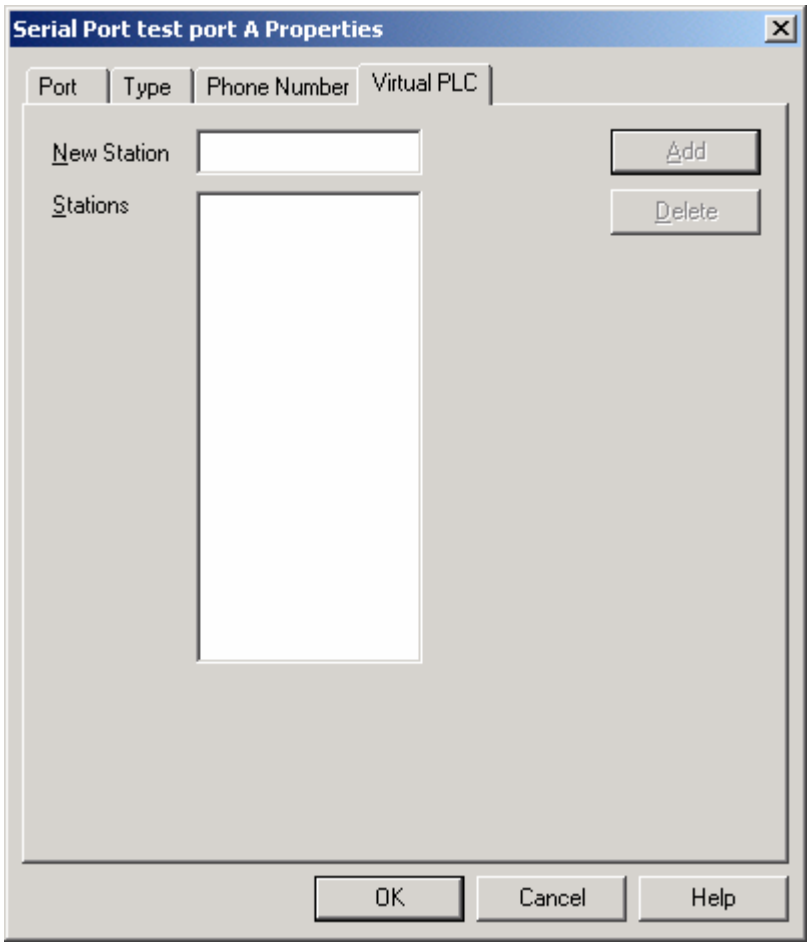
The **Add** button adds the station and phone number to the list. If the station is already in the list it is replaced with the new value. This button is grayed until there is data in both the Station and Phone Number edit boxes.

The **Delete** button removes the selected station from the list. This button is grayed if no station is selected. Only one station may be selected for deletion at a time.

The **Keep Dialup Connection Open** checkbox specifies if the modem connection is kept open when data transfer is completed. When two differing phone numbers are entered, the checkbox is grayed out.

Serial Port Properties Virtual PLC Page

The Serial Port Properties Virtual PLC page defines the virtual PLCs that are provided by the connection.



The **New Station** edit-box specifies the station number to add. The valid range is 1 to 255 if standard addressing is configured and 1 to 65534 if extended addressing is configured.

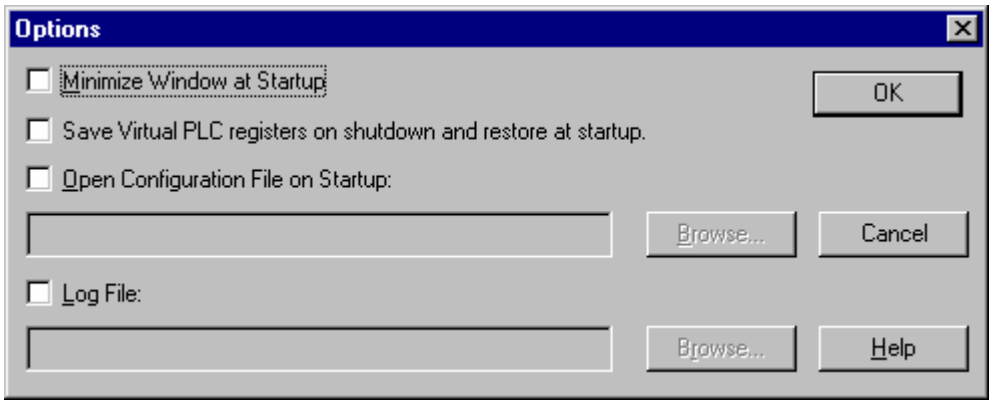
The **Add** button adds the station to the list. Duplicate addresses will not be accepted.

The **Delete** button removes the selected station from the list. This button is grayed if no station is selected.

Options Command

Use this command to specify server start-up and logging options. The command opens the Options dialog.

The Options dialog specifies the configuration file that is used when the SCADAServer Host starts and enables or disables logging of errors.



The **Minimize Window at Start-up** check box specifies if the server window should be minimized at SCADAServer Host start-up.

The **Save Virtual PLC registers on shutdown and restore at startup** checkbox specifies whether the values contained in any configured Virtual PLC database are to be saved when the SCADAServer is shutdown and restored when the SCADAServer is started up again.

The **Open Configuration File on Startup** check box specifies that a configuration file is opened at SCADAServer Host start-up.

- The edit box specifies the name of a configuration file.
- The **Browse** button opens the standard Windows file open dialog. Selecting a file from the dialog inserts the filename into the *Open Configuration File on Start-up* edit box. The *Files of Type* filter on the open dialog is set to SCADAServer Host Configuration (*.SSC).

If the file specified in the *Open Configuration File on Start-up* edit box does not open upon SCADAServer Host start-up an error message will be generated.

The edit box and Browse button are grayed if the *Open Configuration File on Start-up* check-box is not selected.

The **Log File** check box specifies that error and status messages displayed in the main windows be logged to disk.

- The edit box specifies the name of a file where data will be logged.
- The **Browse** button opens the standard Windows file open dialog. Selecting a file from the dialog inserts the filename into the *Log File* edit box. The *Files of Type* filter on the open dialog is set to Text Files (*.TXT).

The edit box and Browse button are grayed if the **Log File** check box is not selected.

The **OK** button checks the settings and closes the dialog. If the file name is not valid, an error message is displayed.

The **Cancel** button closes the dialog and discards any changes.

The **Help** button displays the on-line help for the dialog.

Clear Log Command

Use this command to clear the event log view. The command opens the Clear Log message box.

The **Yes** button deletes all events in the log view. The log file on the disk is not affected.

The **No** button closes the dialog with no further action.

Clear Statistics Command

Use this command to clear the communication statistics. The command opens the Clear Statistics message box.

The **Yes** button to resets all communication counters to zero.

The **No** button closes the dialog with no further action.

Window Menu Commands

The Window commands menu contains commands to set the number and type of open windows when viewing data.

New Window Command

Use this command to create a copy of the currently selected window. You can change the view in the copy of the window so you can look at more than one view at a time.

Cascade Command

Use this command to arrange all open, non-minimized windows, so they stack upon each other with an offset so the title bar of each window is visible. All minimized windows are collected at the bottom of the main window.

Tile Command

Use this command to arrange all open, non-minimized windows, so that all are visible. All minimized windows are collected at the bottom of the main window.

Arrange All Command

Use this command to arrange all icons (minimized windows) at the bottom of the main window.

Open Window List

Use the numbers and filenames listed at the bottom of the Window menu to switch to any open window. Choose the number that corresponds with the window you want to activate.

If there are more than nine open windows, the last item in the open window list will be the command *More Windows*. This will open a dialog with a list box showing all open windows.

Help Commands

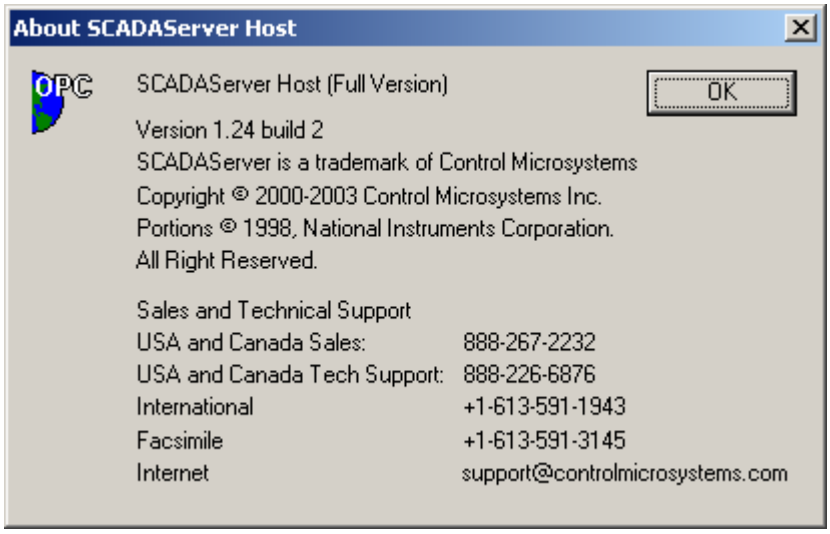
Contents Command

Use this command to open the SCADAServer Host file using the Windows Help program. The Contents page of the help file is displayed.

The help file has a general description of how the SCADAServer Host program operates and how to use the SCADAServer Host program.

About Command

The **About** dialog displays program and copyright information about the SCADAServer Host program.



- The **OK** button closes the dialog.

SCADAServer Client Examples

This section of the manual contains information on configuring various OPC clients for use with the SCADAServer Host.

TelePACE Client

TelePACE allows for the creating, editing, reading and writing of ladder logic code and the writing of C applications to SCADAPack and Micro 16 controllers. Implementing TelePACE as an OPC client allows the user to connect to any SCADAPack and Micro 16 controller through the SCADAServer Host OPC Server. The OPC client connection to the SCADAServer Host is configured in the *PC Communication Settings* dialog in TelePACE.

Configuring PC Communication Settings

The PC Communication Settings dialog defines the communication parameters used for all communication between the target controller and TelePACE. TelePACE supports no flow control, hardware flow control, dial-up modem and SCADAServer Host connections with a target controller. Only the SCADAServer Host connection configuration will be explained here. Refer to the TelePACE Reference and User manual for more information on all of the connection types.

To configure the PC Communication settings to use the SCADAServer Host connection:

- Open TelePACE and select **PC Communication Settings** from the **Communications** menu**.**
- On the left side of the PC Communication Settings dialog select the **SCADAServer** radio button.

The SCADAServer radio button specifies a SCADAServer Host connection. TelePACE will act as an OPC client and route all programming commands through the SCADAServer Host to the SCADAPack or Micro 16 field device. The type of connection to the field device, no flow control, hardware flow control or dial-up modem is configured in the SCADAServer Host itself.

- The **Access Path** edit box specifies the access path of the connection on the server. This is the name of any valid access path in the server. Leaving the edit box blank specifies the path configured as Default in the SCADAServer Host.
- The **Station** edit-box specifies the station address of the target controller. The valid range is 1 to 65534. This control is grayed if the SCADAServer Host radio button is not selected.
- Click on the **OK** button to save the setting and close the dialog.

ISaGRAF Client

Control Microsystems IEC 61131-3 implementation enables the programming of SCADAPack and Micro16 controllers using the IEC 61131-3 programming languages. The programming environment uses the ISaGRAF Workbench to create, load and debug IEC 61131-3 application programs. Implementing ISaGRAF as an OPC client allows the user to connect to any SCADAPack or Micro 16 controller through the SCADAServer Host. The OPC client connection to the SCADAServer Host is configured in the ISaGRAF *PC Communication Settings* dialog.

Configuring PC Communication Settings

The *PC Communication Settings* dialog defines the communication parameters used for all communication on the TeleBUS Driver. The TeleBUS driver supports no flow control, hardware flow control, dial-up modem and SCADAServer Host connections with a target controller. Only the SCADAServer Host connection configuration will be explained here. Refer to the IEC1131 Reference and User manual for more information on all of the connection types.

To configure the PC Communication settings to use the SCADAServer Host connection:

- Open a new or existing program in ISaGRAF.
- Select **Link Setup** from the **Debug.**
- Select **TeleBUS Driver** in the **Communication port:** selection.
- Click the **Setup** button.
- On the left side of the PC Communication Settings dialog select the **SCADAServer** radio button.

The **SCADAServer Host** radio button specifies a SCADAServer Host connection. ISaGRAF will act as an OPC client and route all programming commands through the SCADAServer Host to the SCADAPack or Micro 16 field device. The type of connection to the field device, no flow control, hardware flow control or dial-up modem is configured in the SCADAServer Host itself.

- The **Access Path** edit box specifies the access path of the connection on the server. This is the name of any valid access path in the server, but is usually a serial port name such as COM1. Leaving the edit box blank specifies the path configured as Default in the SCADAServer Host.
- The **Station** edit-box specifies the station address of the target controller. The valid range is 1 to 65534. This control is grayed if the SCADAServer Host radio button is not selected.
- The ***Remote Computer*** check box specifies that the server is on a remote computer. The edit-box specifies the name of the remote computer. Refer to the *Using a Remote Computer Connection* section of the manual for more information.
- Click on the **OK** button to save the setting and close the dialog.

TelePACE Firmware Loader Client

The TelePACE Firmware Loader allows the downloading of SCADAPack and Micro 16 firmware to flash memory-equipped controllers. It also provides controller information to the user. Implementing the TelePACE Firmware Loader as an OPC client allows the user to connect to any SCADAPack or Micro 16 controller through the SCADAServer Host. The OPC client connection to the SCADAServer Host is configured in the Firmware Loader *PC Communication Settings* dialog.

Configuring PC Communication Settings

The *PC Communication Settings* dialog defines the communication parameters used for all communication. The TelePACE Firmware Loader supports no flow control, hardware flow control, dial-up modem and SCADAServer Host connections with a target controller. Only the SCADAServer Host connection configuration will be explained here. Refer to the TelePACE Firmware Loader Reference and User manual for more information on all of the connection types.

To configure the PC Communication settings to use the SCADAServer Host connection:

- Open the Firmware Loader and select the **PC Communication Settings** button.
- On the left side of the PC Communication Settings dialog select the **SCADAServer** radio button.

The **SCADAServer Host** radio button specifies a SCADAServer Host connection. The Firmware Loader will act as an OPC client and route all programming commands through the SCADAServer Host to the SCADAPack or Micro 16 field device. The type of connection to the field device, no flow control, hardware flow control or dial-up modem is configured in the SCADAServer Host itself.

- The **Access Path** edit box specifies the access path of the connection on the server. This is the name of any valid access path in the server, but is usually a serial port name such as COM1. Leaving the edit box blank specifies the path configured as Default in the SCADAServer Host.
- The **Station** edit-box specifies the station address of the target controller. The valid range is 1 to 65534. This control is grayed if the SCADAServer Host radio button is not selected.
- The **Remote Computer** check box specifies that the server is on a remote computer. The edit-box specifies the name of the remote computer. Refer to the *Using a Remote Computer Connection* section of this manual for more information.
- Click on the **OK** button to save the setting and close the dialog.

RealFLO Client

RealFLO is a Man-Machine Interface to the Control Microsystems Dual Run Flow Computer. RealFLO allows editing of the Flow Computer configuration parameters with configuration dialogs for process inputs, contract specifications, density calculations and flow calculations for each meter run. The operator may write configuration data to the Dual Run Flow Computer or read it back.

Implementing RealFLO as an OPC client allows the user to connect to any SCADAPack or Micro 16 controller through the SCADAServer Host.

Only one RealFLO client or host program (HMI) may communicate with a specific RealFLO Flow Computer in the controller at a time. The command interface provided by the RealFLO Flow Computer allows only one host.

The OPC client connection to the SCADAServer Host is configured in the RealFLO *PC Communication Settings* dialog.

Configuring PC Communication Settings

The *PC Communication Settings* dialog defines the communication parameters used for all communication. RealFLO supports no flow control, hardware flow control, dial-up modem and SCADAServer Host connections with a target controller. Only the OPC Server connection configuration will be explained here. Refer to the RealFLO User manual for more information on all of the connection types.

To configure the PC Communication settings to use the SCADAServer Host connection:

- Open a new or existing RealFLO configuration and select **PC Communication Settings** from the **Communications** menu**.**
    - On the left side of the PC Communication Settings dialog select the **SCADAServer** radio button.

The **SCADAServer Host** radio button specifies a SCADAServer Host connection. RealFLO will act as an OPC client and route all programming commands through the OPC Server to the SCADAPack or Micro 16 field device. The type of connection to the field device, no flow control, hardware flow control or dial-up modem is configured in the OPC Server itself.

- The **Access Path** edit box specifies the access path of the connection on the server. This is the name of any valid access path in the server, but is usually a serial port name such as COM1. Leaving the edit box blank specifies the path configured as Default in the OPC Server.
- The **Station** edit-box specifies the station address of the target controller. The valid range is 1 to 65534. This control is grayed if the OPC Server radio button is not selected.
    - The **Remote Computer** check box specifies that the server is on a remote computer. The edit-box specifies the name of the remote computer. Refer to the *Using a Remote Computer Connection* section of the manual for more information.
- Click on the **OK** button to save the setting and close the dialog.

SCADALog Client

The SCADALog application allows the uploading, monitoring and saving of data logs from SCADAPack and Micro 16 controllers. Implementing SCADALog as an OPC client allows the user to connect to any SCADAPack or Micro 16 controller through the SCADAServer Host. The OPC client connection to the SCADAServer Host is configured in the SCADALog *PC Communication Settings* dialog.

Configuring PC Communication Settings

The *PC Communication Settings* dialog defines the communication parameters used for all communication. SCADALog supports no flow control, hardware flow control, dial-up modem and SCADAServer Host connections with a target controller. Only the OPC Server connection configuration will be explained here. Refer to the SCADALog User manual for more information on all of the connection types.

To configure the PC Communication settings to use the SCADAServer Host connection:

- Open a new or existing SCADALog program and select **PC Communication Settings** from the **Communications** menu**.**
- On the left side of the PC Communication Settings dialog select the **SCADAServer** radio button.

The **SCADAServer Host** radio button specifies a SCADAServer Host connection. SCADALog will act as an OPC client and route all programming commands through the OPC Server to the SCADAPack or Micro 16 field device. The type of connection to the field device, no flow control, hardware flow control or dial-up modem is configured in the OPC Server itself.

- The **Access Path** edit box specifies the access path of the connection on the server. This is the name of any valid access path in the server, but is usually a serial port name such as COM1. Leaving the edit box blank specifies the path configured as Default in the OPC Server.
- The **Station** edit-box specifies the station address of the target controller. The valid range is 1 to 65534. This control is grayed if the OPC Server radio button is not selected.

- The **Remote Computer** check box specifies that the server is on a remote computer. The edit-box specifies the name of the remote computer. Refer to the *Using a Remote Computer Connection* section of the manual for more information.

- Click on the **OK** button to save the setting and close the dialog.

LOOKOUT 4.0.1 Client

The LOOKOUT 4.0.1 software package is a human-machine-interface that allows reading data from and writing data to the IO database of a Control Microsystems controller. Implementing LOOKOUT 4.0.1 as an OPC client allows the user to connect to any SCADAPack or Micro 16 controller through the SCADAServer Host.

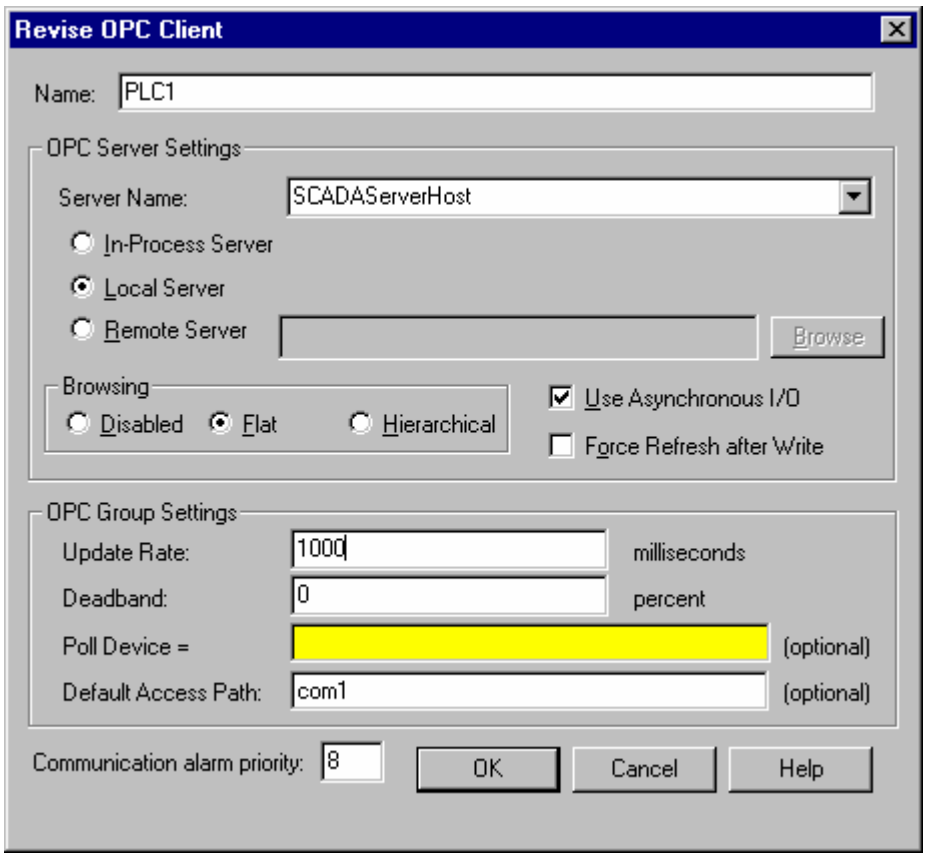
The OPC client connection to the OPC Server is configured in the *OPCClient* object class.

Configuring OPCClient Object

To configure the OPCClient object,

1. Switch to the *Edit Mode* using the *CTRL-Spacebar* toggle or by selecting the *Edit* | *Edit Mode* control. A yellow status line at the bottom of the LOOKOUT screen indicates the Edit Mode.
2. Select *Object* | *Create* and choose *OPCClient* from the *Drivers* folder.
3. Edit the *Name* edit box if a different object name is desired.
4. Select S*CADAServerHost* from the *Server Name* edit box list.
5. Select the *Local Server* radio button.
6. Select the *Flat Browsing* radio button.
7. Edit the *Update Rate* edit box to reflect the desired rate of data updating. The default is 100 milliseconds.
8. Enter the access path in the Default Access path edit box. If this is left blank, the access path defined as *default* in the SCADAPack OPC Server will be used.

The completed object configuration screen should resemble the following diagram:

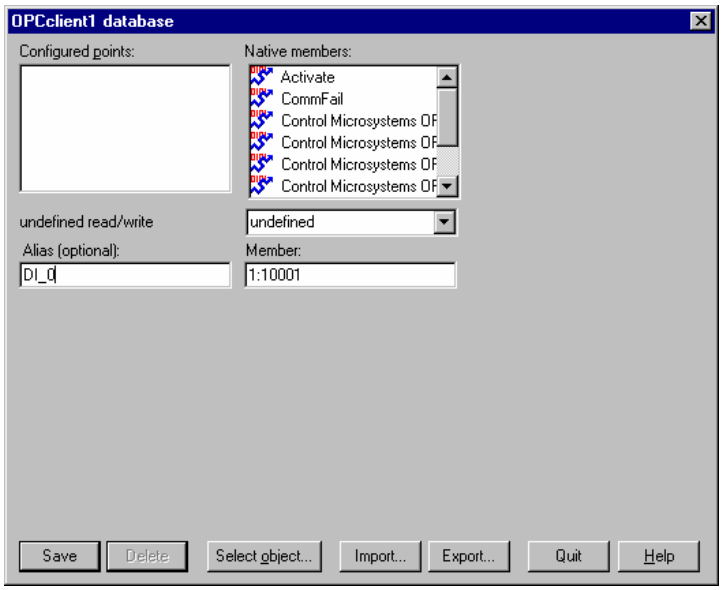


Configuring OPCClient Database

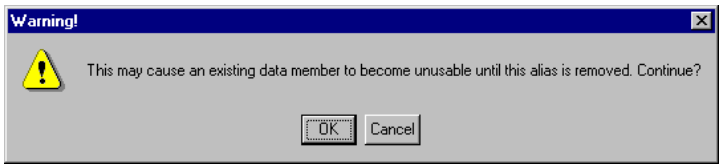
1. Select *Edit* | *Object Database* and open the OPC Client object that was named in the previous instructions; in this example, “OPCclient1”. The *OPCclient1 Database* dialog should appear.
2. In the *Member* edit box, enter the Item ID in the form XXX:YYYYY where XXX is the controller Station Address and YYYYY *is the Modbus Address*. For example, to configure a digital input, Modbus Address 10001, from a controller with a station address of 1, enter 1:10001.

   or

   Select the desired data type from the *Native Members* list. This operation loads a template into the *Member* edit box. Replace ‘Station_ID with the controller station number and replace 0XXXX’, 1XXXX’, 3XXXX’ or 4XXXX’ with the Modbus register.
3. If desired, enter an alias name in the Alias edit box, e.g. DI_01. This alias name will be associated with the item and may be used elsewhere in LOOKOUT to facilitate item identification. The *OPCclient1 Database* dialog should resemble the following diagram:



4. Select the data type from the *undefined read/write* edit box.
5. Select *Save*. The following message dialog appears:



6. Select *OK* to close the dialog.
7. Repeat steps 2 to 4 for additional items.
8. When all items have been completed, select *Quit* to close the *OPCclient1 Database* dialog.

Items can now be displayed on the LOOKOUT panel using standard LOOKOUT procedures. This includes creating display and control objects and connecting them to items programmed in the previous steps, or inserting expressions that include the aforementioned items.

Wonderware® OPCLink and Wonderware Client

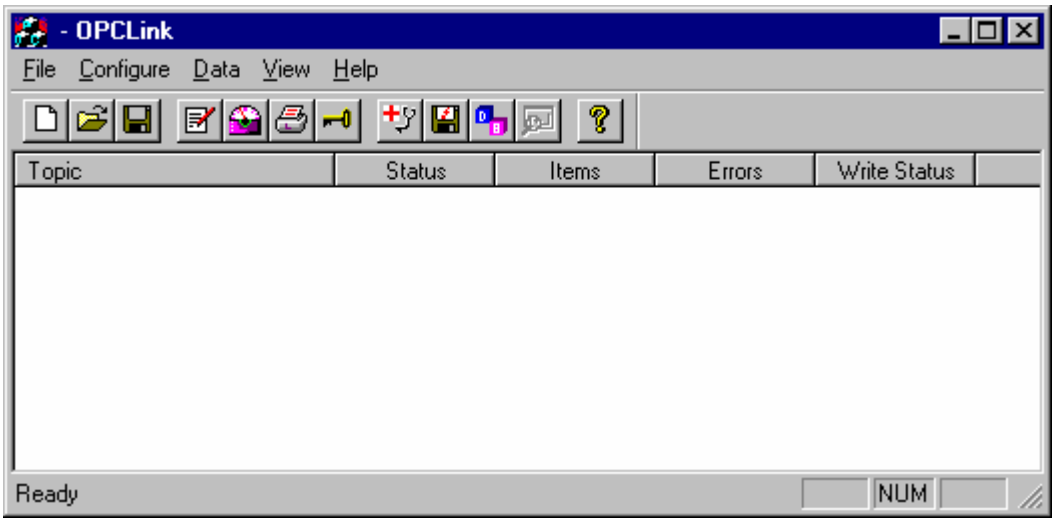
The Wonderware Factory Suite® CLIENT version 7, 0, 0, 1 and Wonderware Factory Suite® OPCLink version 7, 2, 0, 8 software packages comprise a human-machine-interface that allows for the reading of data from and writing data to the IO database of a Control Microsystems controller. Implementing these software packages allows the user to connect to any SCADAPack or Micro 16 controller through the SCADAServer Host.

To configure the Wonderware Factory Suite® CLIENT and Wonderware Factory Suite® OPCLink software packages to use the SCADAServer Host connection:

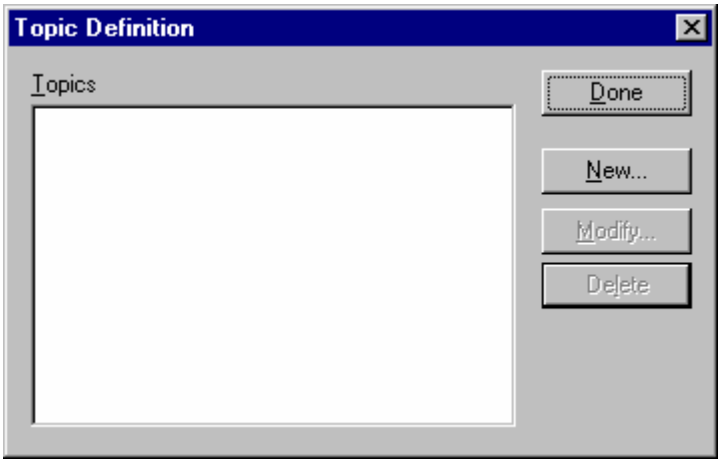
Configure OPCLink

Wonderware's OPCLink allows Wonderware applications such as Wonderware Client to receive data from OPC servers such as SCADAServer Host. The following steps will demonstrate how to create an OPC connection to SCADAServer Host.

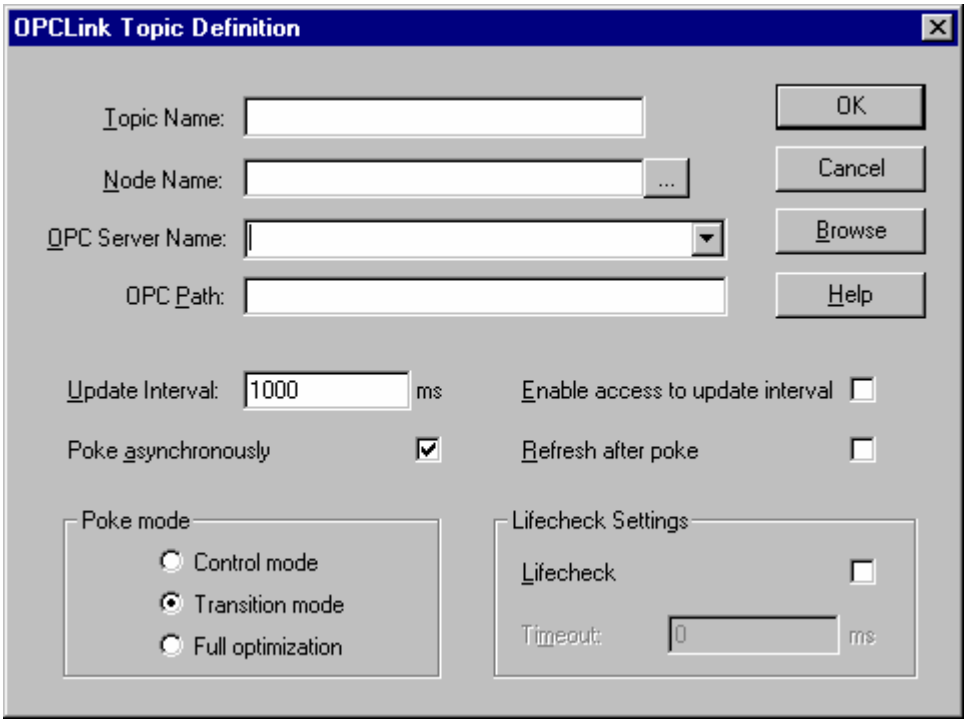
1. Open OPCLink.
2. Select File | New.



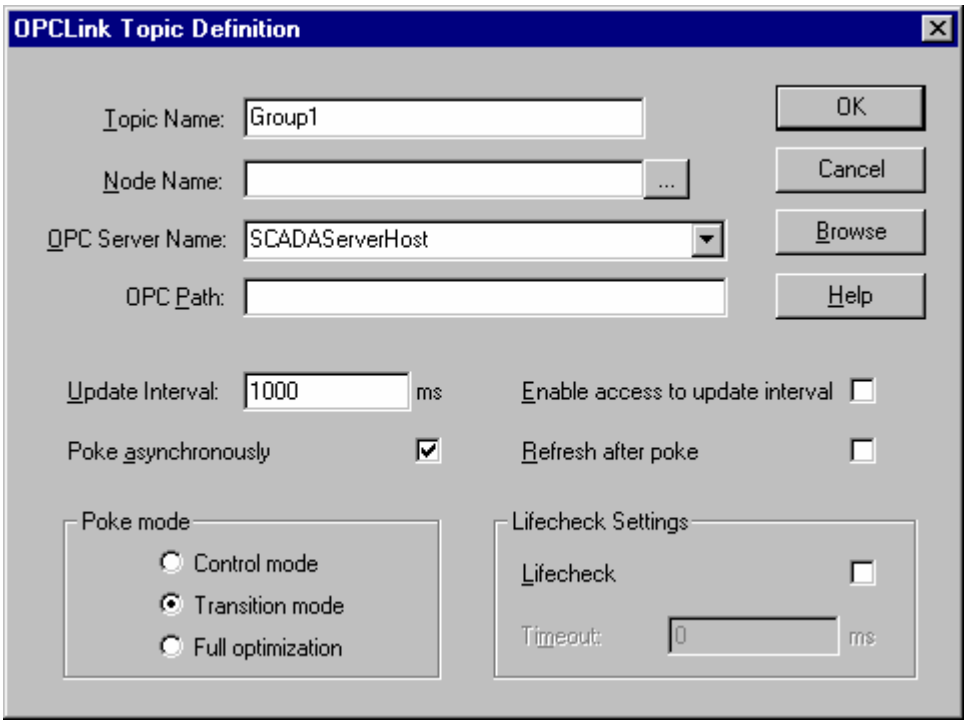
3. Select Configure | Topic Definition.



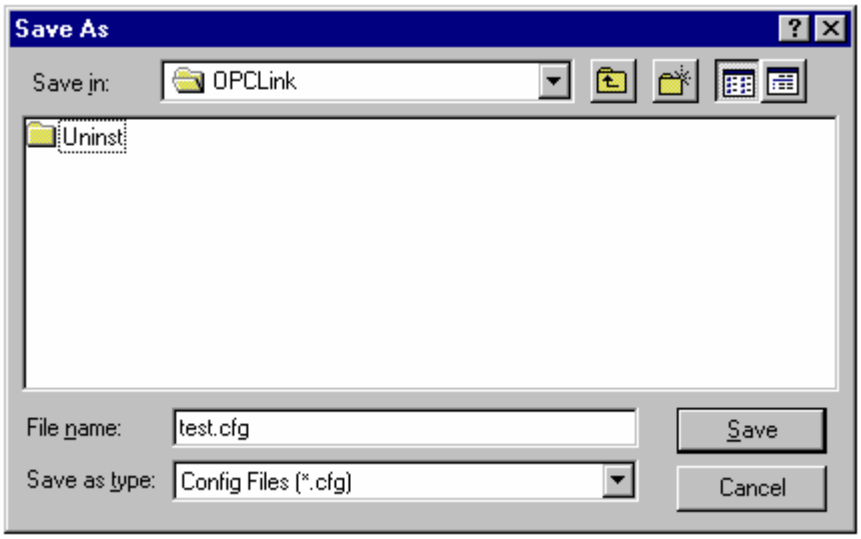
4. Select **New…** to define a new OPCLink topic.



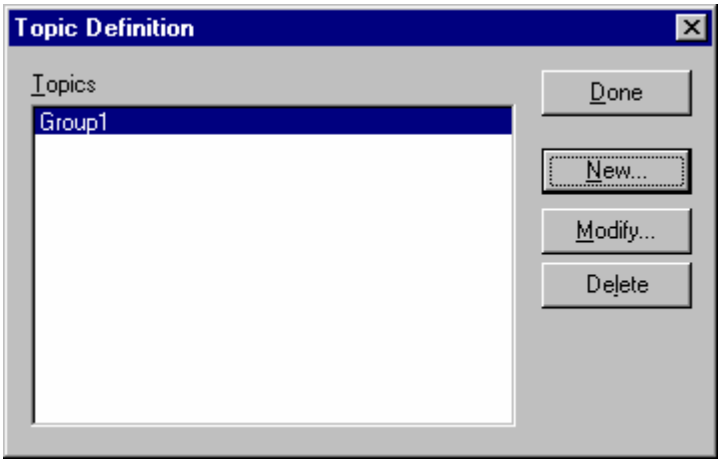
5. In the **Topic Name** field, add a suitable topic name, e.g. *Group1*.
6. If the SCADAServer Host resides on the same computer as OPCLink, leave ***the Node Name*** field blank. If the SCADAServer Host resides on a remote computer, accessible via the computer network, select the remote computer from the **Node Name** drop down menu. In this example, the **Node Name** will be left blank.
7. In the **OPC Server Name** drop down menu, select *SCADAServerHost*.
8. Leave the **OPC Path** field blank.



9. Select **OK** to close the dialog. The OPCLink configuration file *Save As* dialog will appear the first time **OK** is selected.



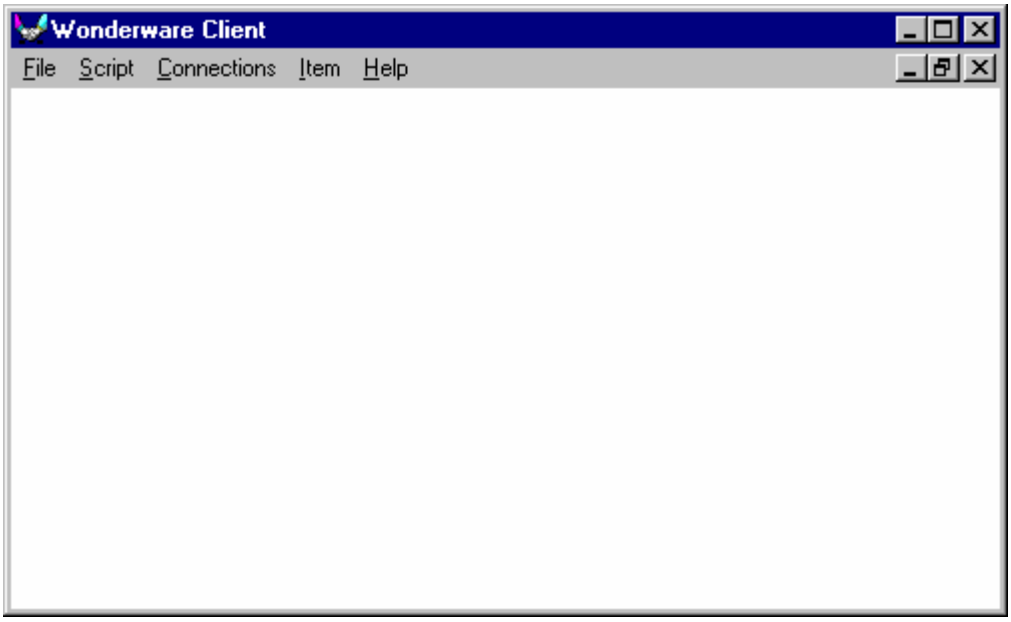
10. Enter a valid name for the configuration file, e.g. t*est.cfg*.
11. Select **Save** to save the configuration and close the dialog.



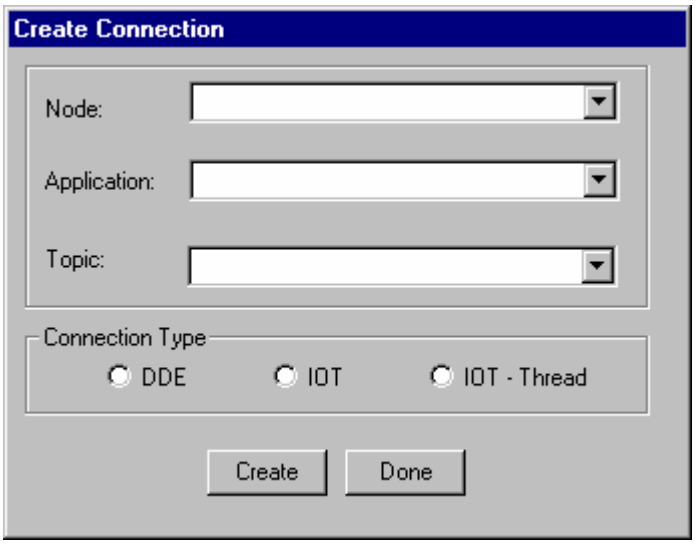
12. Select **Done** to close the Topic Definition dialog.

Connect to SCADAServer Host with Wonderware Client

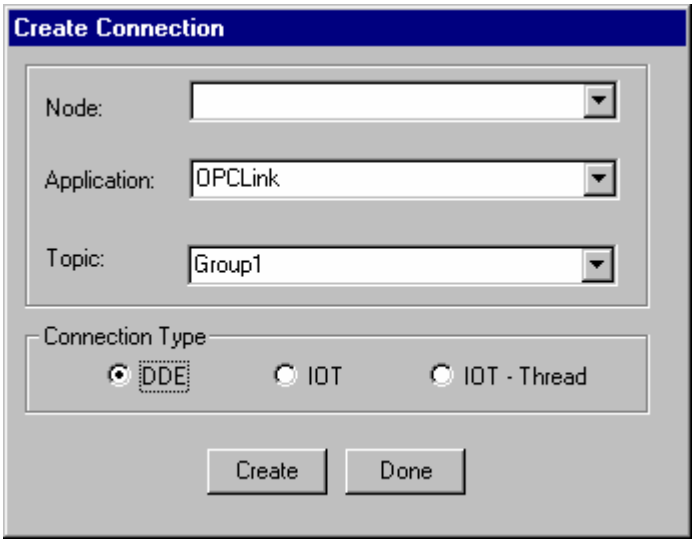
1. Open the *Wonderware Client* application.



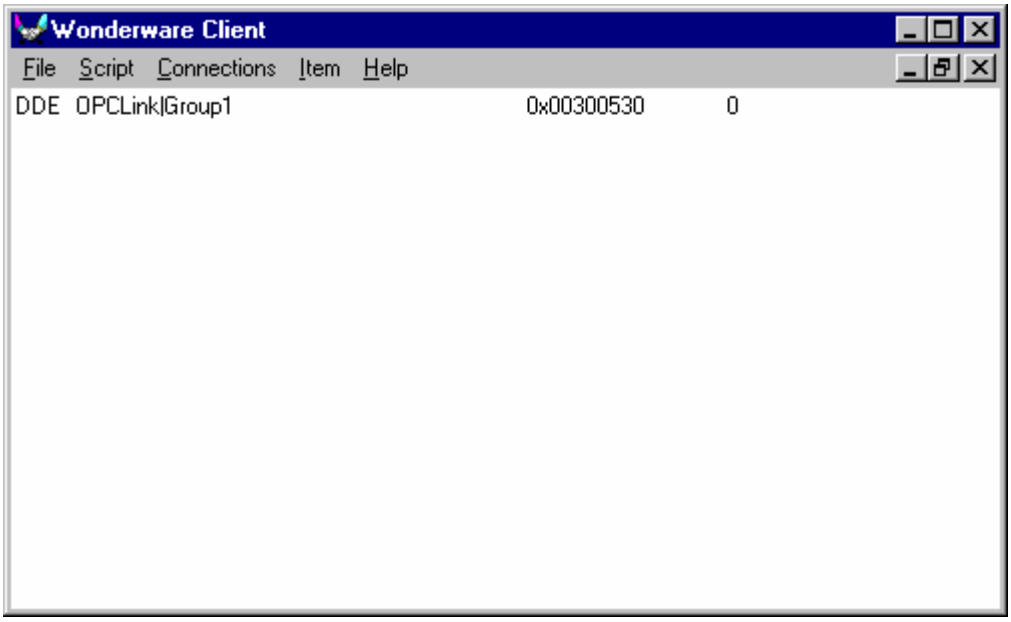
2. Select Connections | Create.



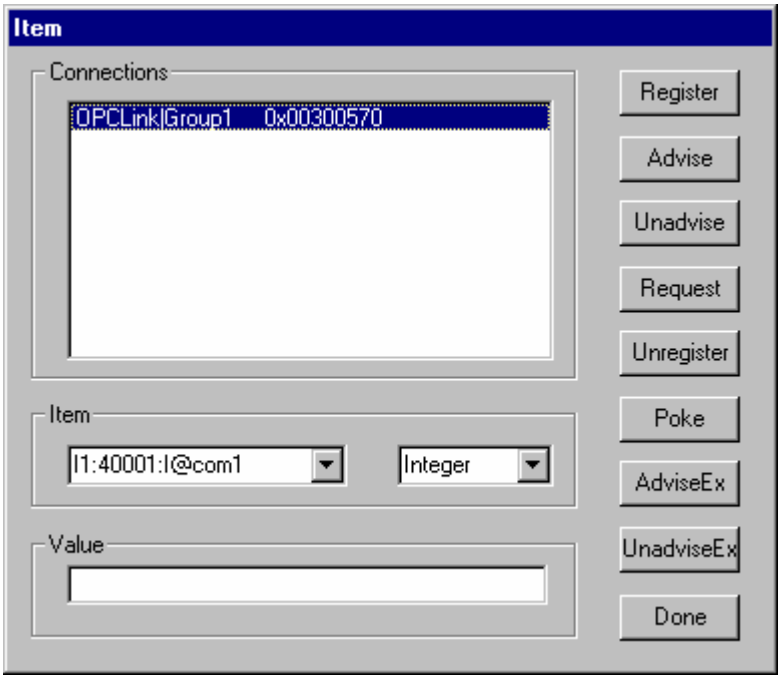
3. Leave the *Node* field blank.
4. Enter "*OPCLink*" (omit quotations) in the **Application** field.
5. Enter the topic name used in step 6 in the **Topic** field, e.g. *Group1*.
6. Select the **DDE** radio button.



7. Select **Create**. The SCADAServer Host application will start up in the background. This may take up to 10 seconds.
8. While the SCADAServer Host is running, select **Done** on the Create Connection dialog to close this dialog. The Wonderware Client should now appear as:

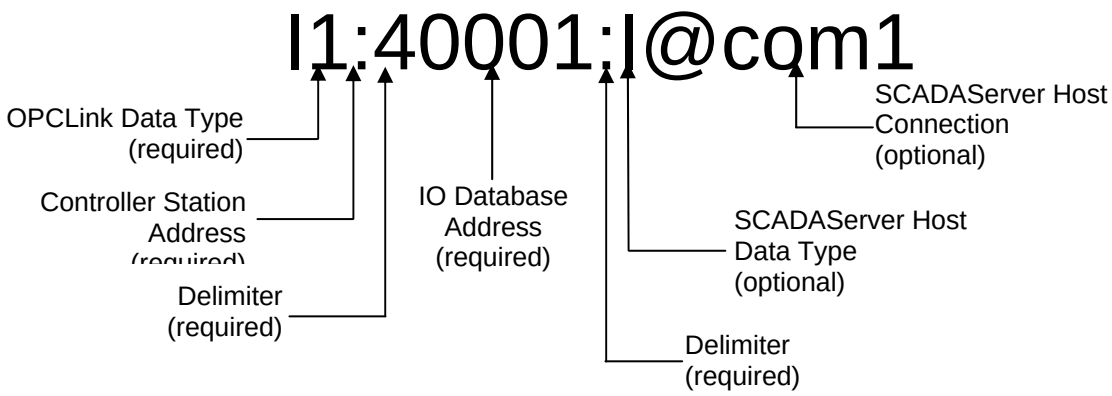


9. Select **Item**.



10. In the **Item** field, enter the full path of the item. Refer to the following diagram for the **Item** field syntax and available SCADAServer Host data types.

Item field Syntax:



OPCLink Data Type: The single letter that precedes the *Item* name tells OPCLink what type of data to request from a server. This letter is required in the field. The four OPCLink data types are:

- D discrete value
- I integer value
- R real value
- M message

Controller Station Address: The station address of the destination controller is placed here. Valid entries are 1 to 255. This address is required in the field. Control Microsystems Extended Addressing is not supported by OPCLink.

Delimiter: The colon (:) is required in the field.

IO Database Address: The destination controller IO Database register address is placed here. Valid entries are 00001 to 04096 for Coil Registers, 10001 to 14096 for Status Registers, 30001 to 31024 for Input registers and 40001 to 49999 for Holding Registers. This address is required by the field.

Delimiter: The colon (:) is required in the field.

SCADAServer Host Data Type: The single or double letter indicates the data type that is requested from the SCADAServer Host. This is required, for non-discrete data types, by the SCADAServer Host and is to be used in addition to the OPCLink Data Type, which is required by OPCLink. The valid SCADAServer Host data types are:

- I 16-bit signed integer (default if no data type is specified)
- U 16-bit unsigned integer
- D 32-bit signed integer (low word in 1st register)
- L 32-bit signed integer (high word in 1st register)
- UD 32-bit unsigned integer (low word in 1st register)
- UL 32-bit unsigned integer (high word in 1st register)
- F 32-bit IEEE754 Floating Point (high word in 1st register)

SCADAServer Host Connection: The connection type, preceded by the @ symbol, is required to specify which SCADAServer Host-configured connection to use. If this is not used, the connection configured as "default" in the SCADAServer Host is used. When in doubt as to which connection has been specified as default in SCADAServer Host, enter the specific connection type.

Examples:

Read unsigned analog input 30100 from slave controller #32 through SCADAServer Host connection com2:

I32:30100:U@com2

Read signed double registers 42000 (low word) and 42001 (high word) from slave controller #12 through SCADAServer Host connection com1:

I12:42000:D@com1

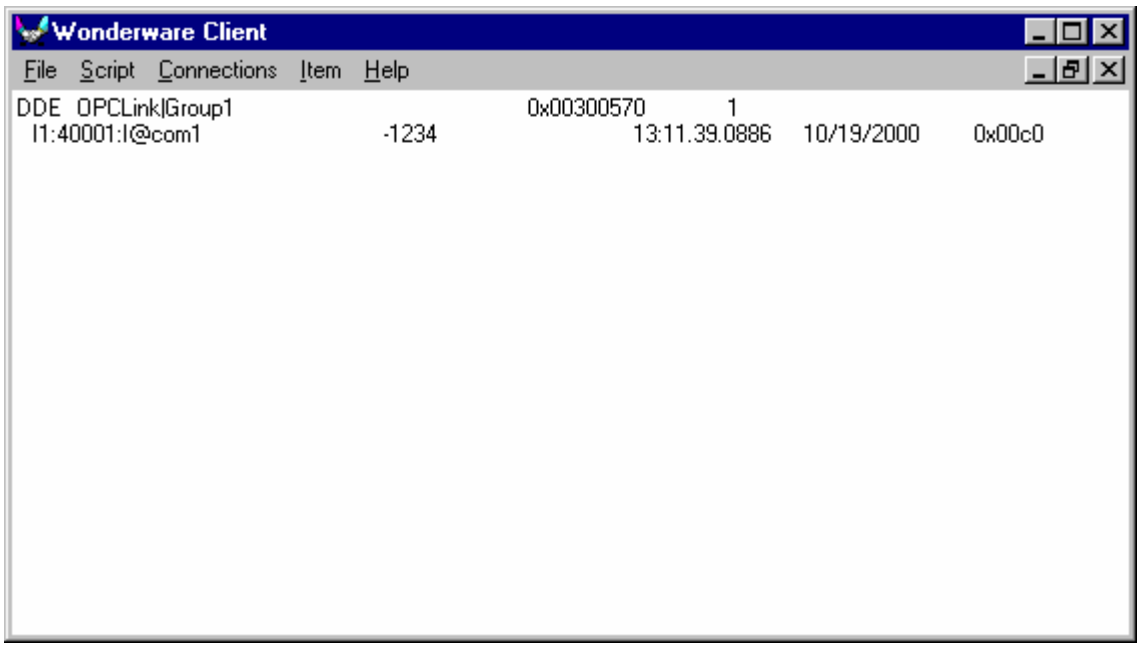
Read digital input register 10025 from slave controller #3 through default SCADAServer Host connection:

D3:10025

Read floating point registers 43123 (high word) and 43125 (low word) from controller #26 through default SCADAServer Host connection:

R26:43123:F

Select **AdviseEx** (Advise by Exception). The Item and item value will appear in the Wonderware Client as follows:


Wonderware Client
File Script Connections Item Help
DDE OPCLink|Group1 0x00300570 1
I1:40001:I@com1 -1234 13:11.39.0886 10/19/2000 0x00c0

PollNow Feature

SCADAServer Host allows the user to perform a demand scan on items (register addresses) in a list by way of its PollNow feature. In order to use the PollNow feature, follow these steps:

1. In the HMI, create a tag name called "PollNow". Ensure its initial value is set to "0" or FALSE.
2. When you want SCADAServer Host to go out and ‘grab’ data from a controller, set “PollNow” from a “0” to a “1” (or TRUE).
3. SCADAServer Host will poll the controller and set the “PollNow” tag back to a “0” (or FALSE).
4. SCADAServer Host will continue to poll based on the HMI's 'Update Interval' time or until "PollNow" is enabled again.

Using a Remote Computer Connection

In certain situations, it may not be possible or practical to have OPC Clients situated in the same computer as the SCADAServer. If this is the case, a Remote Computer Connection may provide a solution to the problem.

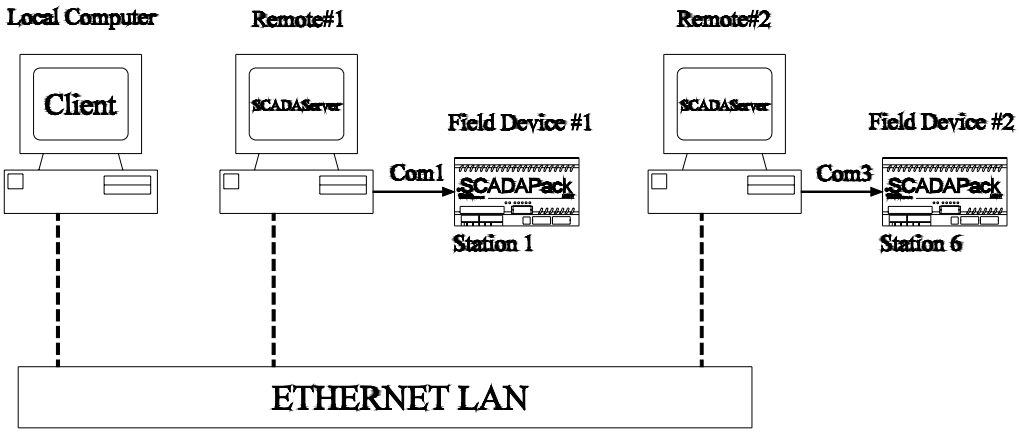
A Remote Computer Connection allows all Control Microsystems OPC clients, with the exception of TelePACE, installed on a local computer, to connect to a SCADAServer Host installed on a remote computer. The two computers must both reside on an ETHERNET Local Area Network (LAN). Once the network connection has been established, the local client will behave exactly as if it is running in the remote computer.

To use a remote computer connection, follow these steps:

1. Configure the SCADAServer Host in the remote computer to reflect the serial connections at the remote location.
2. Configure the Control Microsystems OPC client in the local computer.
    - For RealFLO, SCADALog, ISaGRAF and Firmware Loader clients, open the **PC Communication Settings** dialog. Note that TelePACE does not support remote SCADAServer connection.
    - Select the **SCADAServer** radio button.
    - Enter the **Access Path** and **Station Address** into their respective edit boxes. These parameters should be the same as for a client that is located in the remote computer.
    - Place a checkmark in the **Remote Computer** checkbox.
    - Enter the remote computer's network identification name into the **Remote Computer** edit box. The computer network identification name is the name given to a computer for network identification purposes. On Windows XP O/S, this can be located from the **Start | Settings | Control Panel | System | Computer Name** dialog.

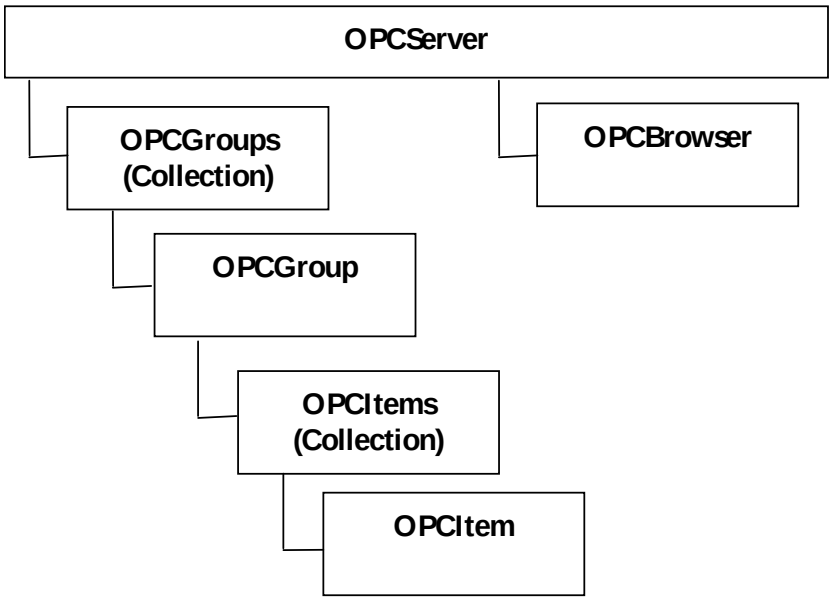
The client will now function as if it is located in the remote computer.

The following diagram illustrates a typical local/remote computer system. The Local Computer and two Remote Computers are all branched off of the same ETHERNET LAN. Each Remote Computer has its own SCADAServer Host configured for connection to its own field device. For Remote#1, the connection is com1. For Remote#2, the connection is com3. In order for the local client to communicate with Field Device #1, it must be configured for Access Path = com1, Station = 1 and Remote Computer = Remote#1. In order for the local client to communicate with Field Device #2, it must be re-configured for Access Path = com3, Station = 6 and Remote Computer = Remote#2.



SCADAServer OLE Automation Reference

The OPC Automation interface allows clients such as Visual BASIC and Delphi applications to access the OPC server. This section is divided into an overview of the automation objects and then a detailed description of the properties and methods for each object. Included with each description is a Visual Basic example of how to use the properties or method.



OPCServer An instance of an OPC Server. You must create an OPCServer object before you can get references to other objects. It contains the OPCGroups Collection and creates OPCBrowser objects.

OPCGroups An Automation collection containing all of the OPCGroup objects this client has created within the scope of the OPCServer that the Automation Application has connected to via the OPCServer.Connect()

OPCGroup An instance of an OPCGroup object. The purpose of this object is to maintain state information and provide the mechanism to provide data acquisition services for the OPCItem Collection object that the OPCGroup object references.

OPCItems An Automation collection containing all of the OPCItem objects this client has created within the scope of the OPCServer, and corresponding OPCGroup object that the Automation Application has created.

OPCItem An automation object that maintains the item's definition, current value, status information, last update time. Note the Custom Interface does not provide a separate Item Object.

OPCBrowser An object that browses item names in the server's configuration. There exists only one instance of an OPCBrowser object per instance of an OPC Server object.

OPCServer Object

Description A client creates the OPCServer Automation object. The client then 'connects' it to an OPC Data Access Custom Interface (see the 'Connect' method). The OPCServer object can now be used to obtain general information about an OPC server and to create and manipulate the collection of OPCGroup objects.'

Syntax OPCServer

Remarks The WithEvents syntax enables the object to support the declared events for the particular object. For the OPCServer, the only event defined is the ServerShutDown. The OPCGroup (described later) has all the events associated with DataChange and the events as required to support the Asynchronous methods of the OPCGroup object.

Example
```
Dim WithEvents AnOPCServer As OPCServer
Set AnOPCServer = New OPCServer
```

Summary of Properties

StartTime	CurrentTime	LastUpdateTime
MajorVersion	MinorVersion	BuildNumber
VendorInfo	ServerState	LocaleID
Bandwidth	OPCGroups	PublicGroupNames
ServerName	ServerNode	ClientName

Summary of Methods

GetOPCServers	Connect	Disconnect
CreateBrowser	GetErrorString	QueryAvailableLocaleIDs
QueryAvailableProperties	GetItemProperties	LookupItemIDs

Summary of Events

ServerShutDown		

OPCServer Properties

StartTime

Description (Read-only) Returns the time the server started running. This is the start time of the server that the client has specified to connect to. Multiple Clients connecting to the same server can be assured that each client will read the same value from the server for this property.

Syntax StartTime As Date

Remarks The automation server is expected to use the custom interface GetStatus () to obtain the values for this property as well as many of the other properties defined as properties of the OPCServer. An error occurs if the client has not connected to a Data Access Server via the Connect method.

Example

```
Dim AnOPCServerTime As Date
AnOPCServerTime = AnOPCServer.StartTime
```

CurrentTime

Description (Read-only) Returns the current time from the server. When you access this property, you will get the value that the automation server has obtained from the custom server via the GetStatus () interface.

Syntax CurrentTime As Date

Remarks An error occurs if the client has not connected to a Data Access Server via the Connect method.

Example

```
Dim AnOPCServerTime As Date
AnOPCServerTime = AnOPCServer.CurrentTime
```

LastUpdateTime

Description (Read-only) Returns the last update time from the server. When you access this property, you will get the value that the automation server has obtained from the custom server via the GetStatus() interface.

Syntax LastUpdateTime As Date

Remarks Returns the last time data was sent from the server to a client application. An error occurs if the client has not connected to a Data Access Server via the Connect method.

Example

```
Dim AnOPCServerTime As Date
AnOPCServerTime = AnOPCServer.LastUpdateTime
```

MajorVersion

Description (Read-only) Returns the major part of the server version number (e.g. the “1” in version 1.32). When you access this property, you will get the value that the automation server has obtained from the custom server via the GetStatus() interface.

Syntax MajorVersion As Integer

Remarks An error occurs if the client has not connected to a Data Access Server via the Connect method.

Example

```
Dim AnOPCServerMajorVersion As String
AnOPCServerMajorVersion = Str(AnOPCServer.MajorVersion)
```

MinorVersion

Description (Read-only) Returns the minor part of the server version number (e.g. the “32” in version 1.32). When you access this property, you will get the value that the automation server has obtained from the custom server via the GetStatus () interface.

Syntax MinorVersion As Integer

Remarks An error occurs if the client has not connected to a Data Access Server via the Connect method.

Example

```
Dim AnOPCServerMinorVersion As String
AnOPCServerMinorVersion = Str(AnOPCServer.MinorVersion)
```

BuildNumber

Description (Read-only) Returns the build number of the server. When you access this property, you will get the value that the automation server has obtained from the custom server via the GetStatus () interface.

Syntax BuildNumber As Integer

Remarks An error occurs if the client has not connected to a Data Access Server via the Connect method.

Example

```
Dim BuildNumber as Integer
BuildNumber = AnOPCServer.BuildNumber
```

VendorInfo

Description (Read-only) Returns the vendor information string for the server. When you access this property, you will get the value that the automation server has obtained from the custom server via the GetStatus () interface.

Syntax VendorInfo As String

Remarks An error occurs if the client has not connected to a Data Access Server via the Connect method.

Example

```
Dim info As String
info = AnOPCServer.VendorInfo
```

ServerState

Description (Read-only) Returns the server's state, which will be one of the OPCServerState values:

Syntax ServerState As Long

OPC_STATUS_RUNNING	The server is running normally. This is the usual state for a server
OPC_STATUS_FAILED	A vendor specific fatal error has occurred within the server. The server is no longer functioning. The recovery procedure from this situation is vendor specific. An error code of E_FAIL should generally be returned from any other server method.
OPC_STATUS_NOCONFIG	The server is running but has no configuration information loaded and thus cannot function normally. Note this state implies that the server needs configuration information in order to function. Servers which do not require configuration information should not return this state.
OPC_STATUS_SUSPENDED	The server has been temporarily suspended via some vendor specific method and is not getting or sending data. Note that Quality will be returned as OPC_QUALITY_OUT_OF_SERVICE.

OPC_STATUS_TEST	The server is in Test Mode. The outputs are disconnected from the real hardware but the server will otherwise behave normally. Inputs may be real or may be simulated depending on the vendor implementation. Quality will generally be returned normally.

Remarks These are the server states that are described in the *OPC Data Access Custom Interface Specification*, and returned by an OPC server via the custom interface. Refer to the *OPC Data Access Custom Interface Specification* IOPCServer::GetStatus() for more details. When you access this property, you will get the value that the automation server has obtained from the custom server via the GetStatus () interface.
An error occurs if the client has not connected to a Data Access Server via the Connect method..

Example

```
Dim ServerState As Long
ServerState = AnOPCServer.ServerState
```

LocaleID

Description (Read/Write) This property identifies the locale, which may be used to localize strings returned from the server. . This LocaleID will be used by the GetErrorString method on this interface

Syntax LocaleID As Long

Remarks It should also be used as the 'default' LocaleID by any other server functions that are affected by LocaleID.
An error occurs if the client has not connected to a Data Access Server via the Connect method.

Example

```
'(getting the property)::
Dim LocaleID As Long
LocaleID = AnOPCServer.LocaleID
 '(setting the property):
 AnOPCServer.LocaleID = LocaleID
```

Bandwidth

Description (Read-only) This is server specific. The suggested use is the server's bandwidth as a percentage of available bandwidth. This value will be hFFFFFFFF when the server cannot calculate a bandwidth. When you access this property, you will get the value that the automation server has obtained from the custom server via the GetStatus () interface.

Syntax Bandwidth As Long

Remarks An error occurs if the client has not connected to a Data Access Server via the Connect method.

Example

```
Dim Bandwidth As Long
Bandwidth = AnOPCServer.Bandwidth
```

OPCGroups

Description (Read only) A collection of OPCGroup objects. This is the default property of the OPCServer object.

Syntax OPCGroups As OPCGroups

Example

```
'(explicit property specification):
Dim groups As OPCGroups
Set groups = AnOPCServer.OPCGroups
 '(using the default specification):
Dim groups As OPCGroups
Set groups = AnOPCServer
```

PublicGroupNames

Description (Read-only) Returns the names of this server's Public Groups. These names can be used in ConnectPublicGroup. The names are returned as an array of strings.

Syntax PublicGroupNames As Variant

Remarks An error occurs if the client has not connected to a Data Access Server via the Connect method. An empty list is returned if the underlying server does not support the Public Groups interface, or if there are no public groups defined.

Example

```
Dim AllPublicGroupNames As Variant
AllPublicGroupNames = AnOPCServer.PublicGroupNames
```

ServerName

Description (Read-only) Returns the server name of the server that the client connected to via Connect().

Syntax ServerName As String

Remarks When you access this property, you will get the value that the automation server has cached locally.
The ServerName is empty if the client is not connected to a Data Access Server.

Example

```
Dim info As String
info = AnOPCServer.ServerName
```

ServerNode

Description (Read-only) Returns the node name of the server that the client connected to via Connect(). When you access this property, you will get the value that the automation server has cached locally.

Syntax ServerNode As String

Remarks The ServerNode is empty if the client is not connected to a Data Access Server.
The ServerNode will be empty if no host name was specified in the Connect method.

Example

```
Dim info As String
info = AnOPCServer.ServerNode
```

ClientName

Description (Read/Write) This property allows the client to optionally register a client name with the server. This is included primarily for debugging purposes. The recommended behavior is that the client set his Node name and EXE name here.

Syntax ClientName As String

Remarks Recommended to put NodeName and ClientName in the string, separated by a semi-colon (;). Refer to the example below for suggested syntax

Example

```
‘(getting the property):
Dim info As String
info = AnOPCServer.ClientName
 ‘(setting the property):
 AnOPCServer.ClientName =
“NodeName;c:\programfiles\vendor\someapplication.exe”
```

OPCServer Methods

GetOPCServers

Description Returns the names (ProgID’s) of the registered OPC Servers. Use one of these ProgIDs in the Connect method. The names are returned as an array of strings.

Syntax GetOPCServers(Optional Node As Variant) As Variant

Part Description

Node The Node name provides the mechanism to specify the remote node where you want the automation server to give you the list of all the registered OPC servers.

Remarks Refer to the OPC Data Access Custom Interface Standard for specific registry requirements for the custom servers.

Node is optional. The use of a node name makes use of DCOM to access another computer. Acceptable node names are UNC names (“Server”), or DNS names (“server.com”, “www.vendor.com”, or “180.151.19.75”).

Example

```
‘ getting the registered OPC Servers (the real OPC servers and
adding them to a standard VB listbox).

Dim AllOPCServers As Variant
AllOPCServers = AnOPCServer.GetOPCServers
For i = LBound(AllOPCServers) To UBound(AllOPCServers)
    listbox.AddItem AllOPCServers(i)
Next i
```

Connect

Description Must be called to establish connection to an OPC Data Access Server (that implements the custom interface).

Syntax Connect (ProgID As String, Optional Node As Variant)

ProgID The ProgID is a string that uniquely identifies the registered real OPC Data Access Server (that implements the custom interface).

Node The Node name can specify another computer to connect using DCOM.

Remarks Each instance of an OPC Automation Server is "connected" to an OPC Data Access Server (which implements the custom interface).

Node is optional. The use of a node name makes use of DCOM to access another computer. Acceptable node names are UNC names ("Server"), or DNS names ("server.com", "www.vendor.com", or "180.151.19.75").

Calling this function will result in the automation wrapper calling CoCreateInstanceEx to create a Data Access Custom(specified by the ProgID)server on the specified machine(StrNodeName).

If this function is called a second time without calling explicitly calling disconnect the automation wrapper will automatically disconnect the existing connection.

See Also Use the GetOPCServers method to find the legal ProgIDs.

Example

```
' Connect to the first registered OPCServer returned from the
GetOPCServers
Dim AllOPCServers As Variant
AllOPCServers = AnOPCServer.GetOPCServers
AnOPCServer.Connect(AllOPCServers(1))
 'Connect to a specific server on some remote node
Dim ARealOPCServer As String
Dim ARealOPCNodeName As String
ARealOPCServer = "VendorX.DataAccessCustomServer"
ARealOPCNodeName = "SomeComputerNodeName"
AnOPCServer.Connect (ARealOPCServer, ARealOPCNodeName)
```

Disconnect

Description Disconnects from the OPC server.

Syntax Disconnect()

Remarks This allows you to disconnect from a server and then either connect to another server, or remove the object. It is it is good programming practice for the client application to explicitly remove the objects that it created (including all OPCGroup(s), and OPCItem(s) using the appropriate automation method. Calling this function will remove all of the groups and release all references to the underlying OPC Custom Server.

Example

```
AnOPCServer.Disconnect
```

CreateBrowser

Description Creates an OPCBrowser object

Syntax CreateBrowser() As OPCBrowser

Remarks The OPC Browse interface is an optional interface that is not required to be supported by an OPC Custom interface server. Therefore, an OPCBrowser object will not be returned for OPC Custom interface servers that do not implement the browse interface.

Example

```
Dim ARealOPCServer As String
Dim ARealOPCNodeName As String
ARealOPCServer = "VendorX.DataAccessCustomServer"
ARealOPCNodeName = "SomeComputerNodeName"
AnOPCServer.Connect(ARealOPCServer, ARealOPCNodeName)
Dim AnOPCServerBrowserObject As OPCBrowser
Set AnOPCServerBrowserObject = AnOPCServer.CreateBrowser
```

GetErrorString

Description Converts an error number to a readable string. The server will return the string in the Locale that is specified in the server level LocaleID property. Refer to the properties of the OPC Server for setting and getting the LocaleID property.

Syntax GetErrorString(ErrorCode As Long) As String

ErrorCode Server specific error code that the client application had returned from an interface function from the server, and for which the client application is requesting the server's textual representation.

Example

```
Dim AnOPCServerErrorString As String
' for this sample, assume while adding some items, we detected
that one of the items was 'invalid.  Not all code included for
clarity reasons.

AnOPCItemCollection.Add AddItemCount, AnOPCItemIDs,
AnOPCItemServerHandles, AnOPCItemErrors
'Get the error string and display it to tell the user why the
item could not be added
AnOPCServerErrorString =
AnOPCServer.GetErrorString(AnOPCItemErrors (index))
'and more code
ErrorBox.Text = AnOPCServerErrorString
'and more code
```

QueryAvailableLocaleIDs

Description Return the available LocaleIDs for this server/client session. The LocaleIDs are returned as an array of longs.

Syntax QueryAvailableLocaleIDs () As Variant

Example

```
Dim LocaleID As Variant
```

```
Dim AnOPCTextSting as String
AnOPCServerLocaleID = AnOPCServer.QueryAvailableLocaleIDs()
For i = LBound(LocaleID) To UBound(LocaleID)
  AnOPCTextSting = LocaleIDToString(LocaleID(i))
    listbox.AddItem AnOPCTextSting
Next i
```

QueryAvailableProperties

Description Return a list of ID codes and Descriptions for the available properties for this ItemID. This list may differ for different ItemIDs. This list is expected to be relatively stable for a particular ItemID. That is, it could be affected from time to time by changes to the underlying system's configuration.

Syntax QueryAvailableProperties (ItemID As String, ByRef Count As Long, ByRef PropertyIDs() as Long, ByRef Descriptions() As String, ByRef DataTypes() As Integer)

ItemID	The ItemID for which the caller wants to know the available properties
Count	The number of properties returned
PropertyIDs	DWORD ids for the returned properties. These IDs can be passed to GetItemProperties or LookupItemIDs
Descriptions	A brief vendor supplied text Description of each Property. NOTE LocalID does not apply to Descriptions.
DataTypes	The datatype which will be returned for this Property by GetItemProperties.

Example

```
' Get the available properties
Dim OPCItemID As String
Dim ItemCount As Long
Dim PropertyIDs() As Long
Dim Descriptions() As String
Dim DataTypes() As Integer
Dim AnOPCTextSting As String
OPCItemID = "SomeOPCDataAccessItem"
AnOPCServer.QueryAvailableProperties (OPCItemID, ItemCount,
PropertyIDs, Descriptions, DataTypes)
For i = 1 To ItemCount
  AnOPCTextSting = Str(PropertyIDs(i) + " " + Descriptions(i)
    listbox.AddItem AnOPCTextSting
Next I
```

GetItemProperties

Description Return a list of the current data values for the passed ID codes.

Syntax GetItemProperties (ItemID As String, Count As Long, ByRef PropertyIDs() as Long, ByRef PropertyValues() As Variant, ByRef Errors() As Long)

ItemID	The ItemID for which the caller wants to read the list of properties
Count	The number of properties passed

PropertyIDs DWORD ids for the requested properties. These Ids were returned by QueryAvailableProperties or obtained from the fixed list described earlier.

PropertyValues An array of size Count VARIANTS returned by the server, which contain the current values of the requested properties.

Errors Error array indicating whether each property was returned.

Example

```
Dim OPCItemID as String
Dim ItemCount As Long
Dim PropertyIDs(3) as Long
Dim Data() as Variant
Dim Errors() as Long
Dim AnOPCTextSting As String
‘ Set values for ItemCount and PropertyIDs…
AnOPCServer.GetItemProperties (OPCItemID, ItemCount,
PropertyIDs, Data, Errors)
For i = 1 To ItemCount
   AnOPCTextSting = Str(PropertyIDs(i) + “ “ + Data(i)
     listbox.AddItem AnOPCTextSting
Next i
```

LookupItemIDs

Description Return a list of ItemIDs (if available) for each of the passed ID codes. These indicate the ItemID, which could be added to an OPCGroup and used for more efficient access to the data corresponding to the Item Properties. An error within the error array may indicate that the passed Property ID is not defined for this item.

Syntax LookupItemIDs (ItemID As String, Count As Long, PropertyIDs() as Long, ByRef NewItemIDs() As String, ByRef Errors () As Long)

ItemID The ItemID for which the caller wants to lookup the list of properties

Count The number of properties passed

PropertyIDs DWORD ids for the requested properties. These IDs were returned by QueryAvailableProperties

NewItemIDs The returned list of ItemIDs.

Errors Error array indicating whether each New ItemID was returned.

Example

```
Dim OPCItemID as String
Dim Count As Long
Dim PropertyIDs(1) as Long
Dim NewItemIDs () as String
Dim Errors() as Long
Dim AnOPCTextSting As String
OPCItemID = “SomeOPCDataAccessItem”
Count = 1
PropertyIDs(1) = 5;
AnOPCServer.LookupItemIDs (OPCItemID, Count, PropertyIDs,
NewItemIDs, Errors)
For i = 1 To Count
```

```
    AnOPCTextSting = Str(PropertyIDs(i) + " " + NewItemIDs(i)
      listbox.AddItem AnOPCTextSting
Next i
```

OPCServer Events

ServerShutDown

Description The ServerShutDown event is fired when the server is planning on shutting down and wants to tell all the active clients to release any resources. The client provides this method so that the server can request that the client disconnect from the server. The client should remove all groups and items.

Syntax ServerShutDown (Reason As String)

ServerReason An optional text string provided by the server indicating the reason for the shutdown.

Example

```
Dim WithEvents AnOPCServer As OPCServer
Dim ARealOPCServer As String
Dim ARealOPCNodeName As String
Set AnOPCServer = New OPCServer ‘ note we need to specify an
example to facilitate creating an object that is
‘dimensioned with events
ARealOPCServer = “VendorX.DataAccessCustomServer”
ARealOPCNodeName = “SomeComputerNodeName”
AnOPCServer.Connect(ARealOPCServer, ARealOPCNodeName)
Private Sub AnOPCServer_ServerShutDown(ByRef aServerReason As
String)
‘ write your client code here to let go of the server
End Sub
```

OPCBrowser Object

Description The OPCBrowser object is a collection of branch or item names that exist in the server. Browsing is optional. If the server does not support browsing, CreateBrowser will not create this object.

Syntax OPCBrowser

Remarks The properties Filter, DataType, and AccessRights affect the collection at the time a method such as ShowLeafs is called. These properties let the client request a subset of the address space.
If the user is browsing names of items to write data to, then the AccessRights property should be set to OPCWritable before calling ShowLeafs.
Servers can have either a flat or hierarchical name space. When the namespace is flat, calling the method ShowLeafs fills the collection with the entire set of names in the server.
Hierarchical browsing is a two step process. First, the browse position is set using a Move method, then the names are put into the collection using the Show methods. Calling ShowBranches fills the collection with the branches below the current position. Calling MoveDown with one of these branch names moves the position to that branch. Calling MoveUp moves up one level. Calling MoveToRoot moves all the way to the top level. From any position, branches and leafs can be browsed.

ShowBranches and ShowLeafs should not be called from inside a For Each loop.

The reason for this restriction is when in a For Each loop or a loop to Count the items, basically you would be changing the contents of the collection, and the next item has no meaning.

Basically, you should not call ShowBranches and ShowLeafs while looping through the Browse object's collection. It is legal to call ShowLeafs while in a loop on some other collection.

Example

```
Dim WithEvents AnOPCServer As OPCServer
Dim ARealOPCServer As String
Dim ARealOPCNodeName As String
Dim AnOPCServerBrowser As OPCBrowser
Dim SomeName As Variant
Set AnOPCServer = New OPCServer
ARealOPCServer = "VendorX.DataAccessCustomServer"
ARealOPCNodeName = "SomeComputerNodeName"
AnOPCServer.Connect(ARealOPCServer, ARealOPCNodeName)
Set browser = AnOPCServer.CreateBrowser
AnOPCServerBrowser.ShowBranches

AnOPCServerBrowser.MoveDown(AnOPCServerBrowser.Item(1) )
AnOPCServerBrowser.DataType = vbInteger
AnOPCServerBrowser.ShowLeafs
' 1st method for getting all names
For I = 1 To AnOPCServerBrowser.Count
   name = AnOPCServerBrowser.Item( I )
' Or…
   name = AnOPCServerBrowser ( I )
   listBox.Add name
Next I
' 2nd method for getting all names
For Each name In AnOPCServerBrowser
   listBox.Add name
Next name
```

Summary of Properties

Organization	Filter	DataType
AccessRights	CurrentPosition	Count

Summary of Methods

Item	ShowBranches	ShowLeafs
MoveUp	MoveToRoot	MoveDown
MoveTo	GetItemID	GetAccessPaths

OPCBrowser Properties

Organization

Description (Read-only) Returns either OPCHierarchical or OPCFlat.

Syntax Organization As Long

Remarks If the organization is OPCFlat, then calling ShowBranches or any Move method has no effect. All names will be available after a single call to ShowLeafs.

Example

```
Dim TheOrganization As Long
Set AnOPCServerBrowser = AnOPCServer.CreateBrowser
TheOrganization = AnOPCServerBrowser.Organization
```

Filter

Description (Read/Write) The filter that applies to ShowBranches and ShowLeafs methods. This property defaults to “” (no filtering). Servers may use this filter to narrow the list of names. Servers are recommended to support wildcards such as “*”.

Syntax Filter As String

Example

```
Dim TheFilter As String
TheFilter = AnOPCServerBrowser.Filter
VB Syntax Example (setting the property):
Dim TheFilter As String
AnOPCServerBrowser.Filter =  “FIC*”
```

DataType

Description (Read/Write) The requested data type that applies to ShowLeafs methods. This property defaults to VT_EMPTY, which means that any data type is acceptable.

Syntax DataType As Integer

Remarks Any legal Variant type can be passed as a requested data type. The server responds with names that are compatible with this data type (may be none). This property provides the mechanism such that the client only gets the leafs that are of a certain DataType.

Example

```
VB Syntax Example (getting the property):
Dim TheDataType As Long
TheDataType = AnOPCServerBrowser.DataType
VB Syntax Example (setting the property):
Dim TheDataType As Long
AnOPCServerBrowser.DataType =  vbInteger
```

AccessRights

Description (Read/Write) The requested access rights that apply to the ShowLeafs methods. This property defaults to OPCReadable OR’d with OPCWritable

(that is, everything). This property applies to the filtering, i.e. you only want the leafs with these AccessRights.

Syntax AccessRights As Long

Example

```
VB Syntax Example (getting the property):
Dim TheAccessRights As Long
TheAccessRights = AnOPCServerBrowser.AccessRights
VB Syntax Example (setting the property):
Dim TheAccessRights As Long
AnOPCServerBrowser.AccessRights =  OPCWritable
```

CurrentPosition

Description (Read-only) Current position in the tree. This string will be “” (i.e. the "root") initially. It will always be “” if Organization is OPCFlat.

Syntax CurrentPosition As String

Remarks Current Position returns the absolute position and is equivalent to calling GetItemID on a branch (see also the Custom Interface Spec).

Example

```
VB Syntax Example (getting the property):
Dim ACurrentPosition As String
AnOPCServerBrowser.MoveDown(“level_1”)
Set ACurrentPosition = AnOPCServerBrowser.CurrentPosition
```

Count

Description (Read-only) Required property for collections. Returns the number of items in the collection.

Syntax Count As Long

Example

```
VB Syntax Example (getting the property):
Dim AnOPCCount As Long
AnOPCCount = AnOPCServerBrowser.Count
```

OPCBrowser Methods

Item

Description Required property for collections. Returns a name indexed by ItemSpecifier. The name will be a branch or leaf name, depending on previous calls to ShowBranches or ShowLeafs. Item is the default for the OPCBrowser.

Syntax Item(ItemSpecifier As Variant) As String

ItemSpecifier 1-based index into the collection

Example

```
AnOPCServerBrowser.ShowBranches
’ 1st method for getting all names
For I = 1 To AnOPCServerBrowser.Count
   SomeName = AnOPCServerBrowser.Item( I )
```

```
   listBox.Add SomeName
Next I
' 2nd  method for getting all names
For I = 1 To AnOPCServerBrowser.Count
   SomeName = AnOPCServerBrowser( I )
   listBox.Add SomeName
Next I
' 3rd  method for getting all names
For Each SomeName In browser
   listBox.Add SomeName
Next SomeName
```

ShowBranches

Description Fills the collection with names of the branches at the current browse position.

Syntax ShowBranches()

Example

AnOPCServerBrowser.ShowBranches

ShowLeafs

Description Fills the collection with the names of the leafs at the current browse position. Default for Flat is FALSE.

Syntax ShowLeafs(Optional Flat As Variant)

Flat	Defines what the collection should contain.
Settings	The Settings for Flat are:
True	the collection is filled with all leafs at the current browse position, as well as all the leafs that are below the current browse position. Basically we are treated from the current position on down as a flat name space.
False	the collection is filled with all leafs at the current browse position

Remarks The names of leafs in the collection should match the filter criteria defined by DataType, AccessRights, and Filter. Default for Flat is FALSE.

Example

```
AnOPCServerBrowser.MoveDown (“Floor1_Mixing”)
AnOPCServerBrowser.ShowLeafs
```

MoveUp

Description Move up one level in the tree.

Syntax MoveUp()

Example

```
AnOPCServerBrowser.MoveUp
```

MoveToRoot

Description Move up to the first level in the tree.

Syntax MoveToRoot()

Example

```
AnOPCServerBrowser.MoveToRoot
```

MoveDown

Description Move down into this branch.

Syntax MoveDown(Branch As String)

Example

```
AnOPCServerBrowser.MoveDown (“Floor1_Mixing”)
```

MoveTo

Description Move to an absolute position.

Syntax MoveTo(Branches() As String)

Branches Branches are an array of branch names from the root to a particular position in the tree.

Remarks This method is equivalent to calling MoveToRoot, followed by MoveDown for each branch name in the array.

Example

```
Dim branches(3) As String
AnOPCServerBrowser.MoveToRoot
branches(1) = “node”
branches(2) = “device”
branches(3) = “group”
browser.MoveTo branches
Set ACurrentPosition = AnOPCServerBrowser.CurrentPosition
‘ACurrentPosition is now “node.device.group”
```

GetItemID

Description Given a name, returns a valid ItemID that can be passed to OPCItems Add method.

Syntax GetItemID(Leaf As String) As String

Leaf The name of a BRANCH or LEAF at the current level.

Remarks The server converts the name to an ItemID based on the current “position” of the browser. It will not correctly translate a name if MoveUp, MoveDown, etc. has been called since the name was obtained.

Example

```
AnOPCServerBrowser.ShowLeafs
For I = 1 To AnOPCServerBrowser.Count
Set AnOPCItemID = AnOPCServerBrowser.GetItemID(I)
Next I
‘ or
AnOPCServerBrowser.MoveDown “Mixing”
Set AnOPCItemID = AnOPCServerBrowser.GetItemID(“FIC101.PV”)
```

GetAccessPaths

Description Returns the strings that are legal AccessPaths for this ItemID. May be Null if there are no AccessPaths for this ItemID or the server does not support them.

Syntax GetAccessPaths(ItemID As String) As Variant

ItemID Fully Qualified ItemID

Remarks AccessPath is the “how” for the server to get the data specified by the ItemID (the what). The client uses this function to identify the possible access paths for the specified ItemID.

Example

```
AnOPCServerBrowser.ShowLeafs
For I = 1 To AnOPCServerBrowser.Count
Set AnAccessPath =
AnOPCServerBrowser.GetAccessPaths(“FIC101.PV)
Next I
‘ or
AnOPCServerBrowser.MoveDown “Mixing”
Set AnAccessPath =
AnOPCServerBrowser.GetAccessPaths(“FIC101.PV)
```

OPCGroups Object

Description OPCGroups is a collection of OPCGroup objects, and the methods that create, remove, and manage them.

This object also has properties for OPCGroup defaults. When OPCGroups are added, the DefaultGroupXXXX properties set its initial state. The defaults can be changed to add OPCGroups with different initial states. Changing the defaults does not affect groups that have already been created. Once an OPCGroup is added, its properties can be modified. This reduces the number of parameters required to call the Add method.

Syntax OPCGroups

Summary of Properties

Parent	DefaultGroupIsActive	DefaultGroupUpdateRate
DefaultGroupDeadband	DefaultGroupLocaleID	DefaultGroupTimeBias
Count		

Summary of Methods

Item	Add	GetOPCGroup
Remove	RemoveAll	ConnectPublicGroup
RemovePublicGroup		

Summary of Events

GlobalDataChange		

Example Syntax Base The following sample code is necessary for the subsequent Visual Basic Examples to be operational. This code is referred to as OPCGroupsObjectBase.

```
Dim WithEvents AnOPCServer As OPCServer
Dim ARealOPCServer As String
```

```
Dim ARealOPCNodeName As String
Dim AnOPCServerBrowser As OPCBrowser
Dim MyGroups As OPCGroups
Dim DefaultGroupUpdateRate As Long
Dim OneGroup As OPCGroup
Dim AnOPCItemCollection As OPCItems
Dim AnOPCItem As OPCItem
Dim ClientHandles(100) As Long
Dim AnOPCItemIDs(100) As String
Dim AnOPCItemServerHandles() As Long
Dim AnOPCItemServerErrors() As Long
Set AnOPCServer = New OPCServer
ARealOPCServer = “VendorX.DataAccessCustomServer”
ARealOPCNodeName = “SomeComputerNodeName”
AnOPCServer.Connect(ARealOPCServer, ARealOPCNodeName)
Set MyGroups = AnOPCServer.OPCGroups
MyGroups.DefaultGroupIsActive = True
Set OneGroup = MyGroups.Add( “AnOPCGroupName” )
Set AnOPCItemCollection = OneGroup.OPCItems
```

OPCGroups Properties

Parent

Description (Read-only) Returns reference to the parent OPCServer object.

Syntax Parent As OPCServer

DefaultGroupIsActive

Description (Read/Write) This property provides the default active state for OPCGroups created using Groups.Add.

Syntax DefaultGroupIsActive As Boolean

Remarks This property defaults to True.

Example

```
VB Syntax Example (getting the property):
Dim DefaultGroupIsActive As Boolean
DefaultGroupIsActive = MyGroups.DefaultGroupIsActive
VB Syntax Example (setting the property):
MyGroups.DefaultGroupIsActive = FALSE
```

DefaultGroupUpdateRate

Description (Read/Write) This property provides the default update rate (in milliseconds) for OPCGroups created using Groups.Add. This property defaults to 1000 milliseconds (1 second).

Syntax DefaultGroupUpdateRate As Long

Example

```
VB Syntax Example (getting the property):
Dim DefaultGroupUpdateRate As Long
DefaultGroupUpdateRate = MyGroups.DefaultGroupUpdateRate
VB Syntax Example (setting the property):
MyGroups.DefaultGroupUpdateRate = 250
```

DefaultGroupDeadband

Description (Read/Write) This property provides the default deadband for OPCGroups created using Groups.Add. A deadband is expressed as percent of full scale (legal values 0 to 100).

Syntax DefaultGroupDeadband As Single

Remarks This property defaults to 0. Error would be generated if value > 100 or less than 0.

Example

```
VB Syntax Example (getting the property):
Dim DefaultGroupDeadband As Single
DefaultGroupDeadband = MyGroups.DefaultGroupDeadband
VB Syntax Example (setting the property):
MyGroups.DefaultGroupDeadband =  10
```

DefaultGroupLocaleID

Description (Read/Write) This property provides the default locale for OPCGroups created using Groups.Add.

Syntax DefaultGroupLocaleID As Long

Remarks This property defaults to the Servers LocaleID..

Example

```
VB Syntax Example (getting the property):
Dim DefaultGroupLocaleID As Long
DefaultGroupLocaleID = MyGroups.DefaultGroupLocaleID
VB Syntax Example (setting the property):
MyGroups.DefaultGroupLocaleID =
ConvertLocaleIdStringToLocaleIdLong (“English”)
```

DefaultGroupTimeBias

Description (Read/Write) This property provides the default time bias for OPCGroups created using Groups.Add.

Syntax DefaultGroupTimeBias As Long

Remarks This property defaults to 0 minutes.

Example

```
VB Syntax Example (getting the property):
Dim DefaultGroupTimeBias As Long
DefaultGroupTimeBias = MyGroups.DefaultGroupTimeBias
VB Syntax Example (setting the property):
MyGroups.DefaultGroupTimeBias = 60
```

Count

Description (Read-only) Required property for collections.

Syntax Count As Long

Example

```
VB Syntax Example :
For index = 1 to MyGroups.Count
‘ some code here
```

```
Next index
```

OPCGroups Methods

Item

Description Returns an OPCGroup by ItemSpecifier. ItemSpecifier is the name or 1-based index into the collection. Use GetOPCGroup to reference by ServerHandle. Item is the default method for OPCGroups.

Syntax Item(ItemSpecifier As Variant) As OPCGroup

Example

```
VB Syntax Example:
Dim AnOPCGroup As OPCGroup
Set AnOPCGroup = MyGroups.Item(3)
‘Or
Set AnOPCGroup = MyGroups(“Group3”)
```

Add

Description Creates a new OPCGroup object and adds it to the collection. The properties of this new group are determined by the current defaults in the OPCServer object. After a group is added, its properties can also be modified.

Syntax Add(Optional Name As Variant) As OPCGroup

Name Name of the group. The name must be unique among the other groups created by this client. If no name is provided, The server-generated name will also be unique relative to any existing groups.

Remarks If the optional name is not specified, the server generates a unique name. This method will fail if a name is specified but it is not unique. A failure in this case results in the OPCGroup object not being created, and Visual Basic will generate an error when attempting to use the object that has not been set. Refer to Appendix A - OPC Automation Error Handling for information on OPC Automation errors and Exceptions.

Example

```
MyGroups.DefaultGroupIsActive = True
Set OneGroup = MyGroups.Add( “AnOPCGroupName” )
```

GetOPCGroup

Description Returns an OPCGroup by ItemSpecifier.

Syntax GetOPCGroup (ItemSpecifier As Variant) As OPCGroup

ItemSpecifier ItemSpecifier is either the OPCGroup’s ServerHandle, or the name of an OPCGroup. Use Item to reference by index.

Example

```
‘ If  “AnOPCGroupName” has already been Added
Set OneGroup = MyGroups.GetOPCGroup( “AnOPCGroupName” )
```

Remove

Description Removes an OPCGroup by Key.

Syntax Remove(ItemSpecifier As Variant)

ItemSpecifier ItemSpecifier is either the OPCGroup's ServerHandle, or the name of an OPCGroup. Use Item to reference by index.

Remarks This method will fail if the group is a public group.

Example

```
Set OneGroup = MyGroups.Add( "AnOPCGroupName" )
'   some more code here
MyGroups.Remove( "AnOPCGroupName" )
'or
Set OneGroup = MyGroups.Add( "AnOPCGroupName" )
'   some more code here
MyGroups.Remove(OneGroup.ServerHandle )
```

RemoveAll

Description Removes all current OPCGroup and OPCItem objects to prepare for server shutdown.

Syntax RemoveAll()

Remarks This is designed to make thorough sub-object cleanup much easier for clients to ensure all objects are released when the Server object is released. It is equivalent to calling Remove on all remaining OPCItem and OPCGroup objects. OPCBrowser objects are not sub-objects of the server, and they are not removed by this method.

Example

```
Set OneGroup = MyGroups.Add( "AnOPCGroupName" )
Set OneGroup = MyGroups.Add( "AnOPCGroupName1" )
Set OneGroup = MyGroups.Add( "AnOPCGroupName2" )
'   some more code here
MyGroups.RemoveAll
```

ConnectPublicGroup

Description Public Groups are pre-existing groups in a server. These groups can be connected rather than added..

Syntax ConnectPublicGroup (Name As String) As OPCGroup

Name Name of group to be connected.

Remarks This method will fail if the server does not support public groups or the name is not valid

Example

```
Set OneGroup = MyGroups.ConnectPublicGroup (
"AnOPCServerDefinedPublicGroup"" )
```

RemovePublicGroup

Description Removes the OPCGroup specified by ItemSpecifier.

Syntax RemovePublicGroup (ItemSpecifier As Variant)

ItemSpecifier The ServerHandle returned by ConnectPublicGroup, or the name of a Public OPCGroup.

Remarks This method will fail if the server does not support public groups, or if the group has not been connected to via ConnectPublicGroup.

Example

```
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
MyGroups.RemovePublicGroup ( “AnOPCGroupName” )
‘or
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
MyGroups.RemovePublicGroup (OneGroup.ServerHandle )
```

OPCGroups Events

GlobalDataChange

Description The GlobalDataChangeevent is an event to facilitate one event handler being implemented to receive and process data changes across multiple groups.

Syntax GlobalDataChange (TransactionID As Long, GroupHandle As Long, NumItems As Long, ClientHandles() As Long, ItemValues() As Variant, Qualities() As Long, TimeStamps() As Date)

TransactionID The client specified transaction ID. A non-0 value for this indicates that this call has been generated as a result of an AsyncRefresh. A value of 0 indicates that this call has been generated as a result of normal subscription processing.

GroupHandle ClientHandle of the OPCGroup Object the changed data corresponds to.

NumItems The number of items returned

ClientHandles Array of client item handles for the items

ItemValues Array of values.

Qualities Array of Qualities for each item's value.

TimeStamps Array of UTC TimeStamps for each item's value

Remarks NOTE – it is recommended that the event OnDataChange on the OPCGroup object be used normally. This event has been provided to facilitate one event handler being set up to process data changes for multiple OPCGroup objects. Normally, your application has an individual event handler for each group to receive and process the data changes. This allows you to have one event handler, and then using the GroupHandle, know which Group the event has been fired on behalf of.

This event will be invoked for each OPCGroup object that contains an item, whose value or state of the value has changed since the last time this event, was fired. The individual event on the OPCGroup object is also fired as well. Your application, when using both event handlers will receive the data value twice, once for the individual group event, and once for the AllGroupsDataChange Event.

Example

```
Dim WithEvents AnOPCGroupCollection As OPCGroups
Private Sub AnOPCGroupCollection_GlobalDataChange
(TransactionID As Long, GroupHandle As Long, MasterQuality As
```

```
Long, MasterError As Long, NumItems As Long, ClientHandles() As
Long, ItemValues() As Variant, Qualities() As Long,
TimeStamps() As Date)
‘ write your client code here to process the data change values
End Sub
```

OPCGroup Object

Description 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.

Syntax OPCGroup

Summary of Properties

Parent	Name	IsPublic
IsActive	IsSubscribed	ClientHandle
ServerHandle	LocaleID	TimeBias
DeadBand	UpdateRate	OPCItems

Summary of Methods

SyncRead	SyncWrite	AsyncRead
AsyncWrite	AsyncRefresh	AsyncCancel

Summary of Events

DataChange	AsyncReadComplete	AsyncWriteComplete
AsyncCancelComplete		

Example Syntax Base The following sample code is necessary for the subsequent Visual Basic Examples to be operational. This code is referred to as OPCGroupObjectBase.

```
Dim WithEvents AnOPCServer As OPCServer
Dim ARealOPCServer As String
Dim ARealOPCNodeName As String
Dim AnOPCServerBrowser As OPCBrowser
Dim MyGroups As OPCGroups
Dim DefaultGroupUpdateRate As Long
Dim WithEvents OneGroup As OPCGroup
Dim AnOPCItemCollection As OPCItems
Dim AnOPCItem As OPCItem
Dim ClientHandles(100) As Long
Dim AnOPCItemIDs(100) As String
Dim AnOPCItemServerHandles() As Long
Dim AnOPCItemServerErrors() As Long
Set AnOPCServer = New OPCServer
```

```
ARealOPCServer = “VendorX.DataAccessCustomServer”
ARealOPCNodeName = “SomeComputerNodeName”
AnOPCServer.Connect(ARealOPCServer, ARealOPCNodeName)
Set MyGroups = AnOPCServer.OPCGroups
MyGroups.DefaultGroupIsActive = True
Set OneGroup = MyGroups.Add( “AnOPCGroupName” )
Set AnOPCItemCollection = OneGroup.OPCItems
For  x = 1 To AddItemCount
  ClientHandles(x)  = x + 1
  AnOPCItemID(x) = “Register_”  &  x
Next x
AnOPCItemCollection.AddItems AddItemCount, AnOPCItemIDs,
AnOPCItemServerHandles, AnOPCItemServerErrors
```

OPCGroup Properties

Parent

Description (Read-only) Returns reference to the parent OPCServer object.

Syntax Parent As OPCServer

Name

Description (Read/Write) The name given to this group.

Syntax Name As String

Name Name of the group. The name must be a unique group name, with respect to the naming of other groups created by this client.

Remarks Naming a group is optional. This property allows the user to specify a name, therefore a unique name must be provided when setting the value for this property.
The server will generate a unique name for the group, if no name is specified, on the Add method of the OPCGroups object.

Example

```
VB Syntax Example (getting the property):
Dim CurrentValue As String
Set OneGroup = MyGroups.Add( “AnOPCGroupName” )
CurrentValue = OneGroup.Name
VB Syntax Example (setting the property):
Set OneGroup = MyGroups.Add( “AnOPCGroupName” )
OneGroup.Name = “aName”
```

IsPublic

Description (Read-only) Returns True if this group is a public group, otherwise False.

Syntax IsPublic As Boolean

Example

```
Dim CurrentValue As Boolean
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
Set CurrentValue = OneGroup.IsPublic ‘ to get the value
```

IsActive

Description (Read/Write) This property controls the active state of the group. A group that is active acquires data. An inactive group typically does not continue data acquisition except as required for read/writes.

Syntax IsActive As Boolean

Remarks Default value for this property is the value from the OPCGroups corresponding default value at time of the Add();

Example

```
VB Syntax Example (getting the property):
Dim CurrentValue As Boolean
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
Set CurrentValue = OneGroup.IsActive ‘ to get the value
VB Syntax Example (setting the property):
Dim CurrentValue As Boolean
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
OneGroup.IsActive = True
```

IsSubscribed

Description (Read/Write) This property controls asynchronous notifications to the group. A group that is subscribed receives data changes from the server.

Syntax IsSubscribed As Boolean

Remarks Default value for this property is the value from the OPCGroups corresponding default value at time of the Add();

Example

```
VB Syntax Example (getting the property):
Dim CurrentValue As Boolean
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
Set CurrentValue = OneGroup.IsSubscribed ‘ to get the value
VB Syntax Example (setting the property):
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
OneGroup.IsSubscribed = True ‘ to set the value
```

ClientHandle

Description (Read/Write) A Long value associated with the group. Its purpose is for the client to quickly locate the destination of data. The handle is typically an index, etc. This handle will be returned to the client along with data or status.

Syntax ClientHandle As Long

Example

```
VB Syntax Example (getting the property):
Dim CurrentValue As Long
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
Set CurrentValue  = OneGroup.ClientHandle ‘ to get the value
VB Syntax Example (setting the property):
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
OneGroup.ClientHandle = 1975 ‘ to set the value
```

ServerHandle

Description (Read-only) The server assigned handle for the group. The ServerHandle is a Long that uniquely identifies this group. The client must supply this handle to some of the methods that operate on OPCGroup objects (such as OPCGroups.Remove).

Syntax ServerHandle As Long

Example

```
VB Syntax Example (getting the property):
Dim CurrentValue As Long
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
Set CurrentValue  = OneGroup.ServerHandle ‘ to get the value
```

LocaleID

Description (Read/Write) This property identifies the locale, which may be used to localize strings returned from the server. This property’s default depends on the value set in the OPCGroups Collection..

Syntax LocaleID As Long

Example

```
VB Syntax Example (getting the property):
Dim CurrentValue As Long
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
Set CurrentValue = OneGroup.LocaleID
VB Syntax Example (setting the property):
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
OneGroup.LocaleID = StringToLocaleID(“English”)
```

TimeBias

Description (Read/Write). This property provides the information needed to convert the time stamp on the data back to the local time of the device.

Syntax TimeBias As Long

Example

```
VB Syntax Example (getting the property):
Dim CurrentValue As Long
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
Set CurrentValue = OneGroup.TimeBias
VB Syntax Example (setting the property):
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
OneGroup.TimeBias = 100
```

DeadBand

Description (Read/Write) A deadband is expressed as percent of full scale (legal values 0 to 100). This property’s default depends on the value set in the OPCGroups Collection.

Syntax DeadBand As Single

Example

```
VB Syntax Example (getting the property):
Dim CurrentValue As Single
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
Set CurrentValue = OneGroup.DeadBand
VB Syntax Example (setting the property):
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
OneGroup.DeadBand = 5
```

UpdateRate

Description (Read/Write) The fastest rate at which data change events may be fired. A slow process might cause data changes to fire at less than this rate, but they will never exceed this rate. Rate is in milliseconds. This property’s default depends on the value set in the OPCGroups Collection. Assigning a value to this property is a “request” for a new update rate. The server may not support that rate, so reading the property may result in a different rate (the server will use the closest rate it does support).

Syntax UpdateRate As Long

Example

```
VB Syntax Example (getting the property):
Dim CurrentValue As Long
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
DefaultGroupUpdateRate = OneGroup.UpdateRate
VB Syntax Example (setting the property):
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName” )
‘   some more code here
OneGroup.UpdateRate = 50
```

OPCItems

Description A collection of OPCItem objects. This is the default property of the OPCGroup object.

Syntax OPCItems As OPCItems

Example

```
VB Syntax Example (getting the property):
Dim AnOPCItemCollection As OPCItems
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup (“AnOPCGroupName”)
‘   some more code here
Set AnOPCItemCollection = OneGroup.OPCItems
```

OPCGroup Methods

SyncRead

Description This function reads the value, quality and timestamp information for one or more items in a group.

Syntax SyncRead(Source As Integer, NumItems As Long, ServerHandles() As Long, ByRef Values() As Variant, ByRef Errors() As Long, Optional ByRef Qualities As Variant, Optional ByRef TimeStamps As Variant)

Source The ‘data source’; OPC_DS_CACHE or OPC_DS_DEVICE

NumItems The number of items to be read.

ServerHandles Array of server item handles for the items to be read

Values Array of values.

Errors Array of Long’s indicating the success of the individual item reads. This indicates whether the read succeeded in obtaining a defined value, quality and timestamp. NOTE any FAILED error code indicates that the corresponding Value, Quality and Time stamp are UNDEFINED.

Qualities Variant containing an Integer Array of Qualities.

TimeStamps Variant containing a Date Array of UTC TimeStamps. If the device cannot provide a timestamp then the server will provide one.

Remarks The function runs to completion before returning. The data can be read from CACHE in which case it should be accurate to within the ‘UpdateRate’ and percent deadband of the group. The data can be read from the DEVICE in which case an actual read of the physical device is to be performed. The exact implementation of CACHE and DEVICE reads is not defined by this specification.
When reading from CACHE, the data is only valid if both the group and the item are active. If either the group or the item is inactive, then the Quality will indicate out of service (OPC_QUALITY_OUT_OF_SERVICE). Refer to the discussion of the quality bits later in this document for further information.
DEVICE reads are not affected by the ACTIVE state of the group or item.

Example

```
Private Sub ReadButton_Click()
```

```
Dim Source As Integer
Dim NumItems As Long
Dim ServerIndex As Long
Dim ServerHandles(10) As Long
Dim Values() As Variant
Dim Errors() As Long
Dim Qualities() As Variant
Dim TimeStamps() As Variant
Source = OPC_DS_DEVICE
NumItems = 10
For ServerIndex = 1 to NumItems
‘ set up which items to be read
ServerHandles(ServerIndex) =
AnOPCItemServerHandles(ServerIndex)
Next ServerIndex
OneGroup.SyncRead Source, NumItems, ServerHandles, Values,
Errors, Qualities, TimeStamps
For ServerIndex = 1 to NumItems
‘ process the values
TextBox(ServerIndex).Text = Values(ServerIndex)
Next ServerIndex
End Sub
```

SyncWrite

Description Writes values to one or more items in a group. The function runs to completion. The values are written to the DEVICE. That is, the function should not return until it verifies that the device has actually accepted (or rejected) the data.

Syntax SyncWrite(NumItems As Long, ServerHandles() As Long, Values() As Variant, ByRef Errors() As Long)

NumItems	Number of items to be written
ServerHandles	Array of server item handles for the items to be written
Values	Array of values.
Errors	Array of Long’s indicating the success of the individual item writes..

Remarks Writes are not affected by the ACTIVE state of the group or item.

Example

```
Private Sub WriteButton_Click()
Dim Source As Integer
Dim NumItems As Long
Dim ServerIndex As Long
Dim ServerHandles() As Long
Dim Values() As Variant
Dim Errors() As Long
NumItems = 10
For ServerIndex = 1 to NumItems
‘ set up which items to be written
ServerHandles(ServerIndex) =
AnOPCItemServerHandles(ServerIndex)
Values(ServerIndex) = ServerIndex * 2 ‘ any random value for
this example would suffice
Next ServerIndex
```

```
OneGroup.SyncWrite NumItems, ServerHandles, Values, Errors
For ServerIndex = 1 to NumItems
‘ process the Errors
TextBox(ServerIndex).Text = Errors(ServerIndex)
Next ServerIndex
End Sub
```

AsyncRead

Description Read one or more items in a group. The results are returned via the AsyncReadComplete event associated with the OPCGroup object.

Reads are from ‘DEVICE’ and are not affected by the ACTIVE state of the group or item.

Syntax AsyncRead(NumItems As Long, ServerHandles() As Long, ByRef Errors() As Long, TransactionID As Long, ByRef CancelID As Long)

NumItems The number of items to be read.

ServerHandles Array of server item handles for the items to be read

Errors Array of Long’s indicating the status of the individual items to be read.

TransactionID The client specified transaction ID. This is included in the ‘completion’ information provided in the Corresponding Event.

CancelID A Server generated transaction ID. This is provided to enable the client to cancel the “transaction”.

Remarks The AsyncRead requires the OPCGroup object to have been dimensioned with events (Dim WithEvents xxx As OPCGroup) in order for the results of the AsyncRead operation to be returned to the automation client application. The AsyncReadComplete event associated with the OPCGroup object will be fired (called) by the automation server with the results of the AsyncRead operation.

See Also IOPCAsyncIO2: Read from the OPC Data Access Custom Interface Specification.

Example

```
Private Sub AsyncReadButton_Click()
Dim NumItems As Long
Dim ServerIndex As Long
Dim ServerHandles(10) As Long
Dim Values() As Variant
Dim Errors() As Long
Dim ClientTransactionID As Long
Dim ServerTransactionID As Long
Dim Qualities() As Variant
Dim TimeStamps() As Variant
NumItems = 10
ClientTransactionID = 1975
For ServerIndex = 1 to NumItems
‘ set up which items to be read
ServerHandles(ServerIndex) =
AnOPCItemServerHandles(ServerIndex)
Next ServerIndex
OneGroup.AsyncRead NumItems, ServerHandles, Errors,
ClientTransactionID , ServerTransactionID
End Sub
```

AsyncWrite

Description Write one or more items in a group. The results are returned via the AsyncWriteComplete event associated with the OPCGroup object.

Syntax AsyncWrite(NumItems As Long, ServerHandles() As Long, Values() As Variant, ByRef Errors() As Long, TransactionID As Long, ByRef CancelID As Long)

NumItems The number of items to be written.

ServerHandles Array of server item handles for the items to be written.

Values Array of values.

Errors Array of Long's indicating the status of the individual items to be written.

TransactionID The client specified transaction ID. This is included in the 'completion' information provided in the Corresponding Event.

CancelID A Server generated transaction ID. This is provided to enable the client to cancel the "transaction".

Remarks The AsyncWrite requires the OPCGroup object to have been dimensioned with events (Dim WithEvents xxx As OPCGroup) in order for the results of the AsyncWrite operation to be returned to the automation client application. The AsyncWriteComplete event associated with the OPCGroup object will be fired (called) by the automation server with the results of the AsyncWrite operation.

See Also IOPCAsyncIO2:: Write from the OPC Data Access Custom Interface Specification.

Example

```
Private Sub AsyncWriteButton_Click()
Dim NumItems As Long
Dim ServerIndex As Long
Dim ServerHandles(10) As Long
Dim Values() As Variant
Dim Errors() As Long
Dim ClientTransactionID As Long
Dim ServerTransactionID As Long
NumItems = 10
For ServerIndex = 1 to NumItems
ClientTransactionID = 1957
‘ set up which items to be write
ServerHandles(ServerIndex) =
AnOPCItemServerHandles(ServerIndex)
Values(ServerIndex) = ServerIndex * 2  ‘any random value for
this example would suffice
Next ServerIndex
OneGroup.AsyncWrite NumItems, ServerHandles, Values, Errors,
ClientTransactionID , ServerTransactionID
End Sub
```

AsyncRefresh

Description Generate an event for all active items in the group (whether they have changed or not). Inactive items are not included in the callback. The results are returned via the DataChange event associated with the OPCGroup object,

as well as the GlobalDataChange event associated with the OPCGroups object.

Syntax AsyncRefresh(Source As Integer, TransactionID As Long,ByRef CancelID As Long)

Source The 'data source'; OPC_DS_CACHE or OPC_DS_DEVICE

TransactionID The client specified transaction ID. This is included in the 'completion' information provided in the Corresponding Event.

CancelID A Server generated transaction ID. This is provided to enable the client to cancel the "transaction".

Remarks The AsyncRefresh requires the OPCGroup object to have been dimensioned with events (Dim WithEvents xxx As OPCGroup) in order for the results of the refresh operation to be returned to the automation client application. The DataChange event associated with the OPCGroup object will be fired (called) by the automation server with the results of the refresh operation.
If the automation client application has dimensioned with events the OPCGroups Object (Dim WithEvents xyz as OPCGroups), then the GlobalDataChange event associated with the OPCGroups object will be fired (called) by the automation server with the results of the refresh operation.

See Also IOPCAsyncIO::Refresh from the OPC Data Access Custom Interface Specification.

Example

```
Dim MyGroups As OPCGroups
Dim DefaultGroupUpdateRate As Long
Dim WithEvents OneGroup As OPCGroup
Private Sub AsyncRefreshButton_Click()
Dim ServerIndex As Long
Dim Source As Long
Dim ClientTransactionID As Long
Dim ServerTransactionID As Long
ClientTransactionID = 2125
Source = OPC_DS_DEVICE
OneGroup.AsyncRefresh Source, ClientTransactionID
ServerTransactionID
End Sub
```

AsyncCancel

Description Request that the server cancel an outstanding transaction. An AsyncCancelComplete event will occur indicating whether or not the cancel succeeded.

Syntax AsyncCancel(CancelID As Long)

CancelID The Server generated CancelID that was previously returned by the AsyncRead, AsyncWrite or AsyncRefresh method that the client now wants to cancel.

See Also IOPCAsyncIO2::Cancel from the OPC Data Access Custom Interface Specification.

Remarks The AsyncCancel requires the OPCGroup object to have been dimensioned with events (Dim WithEvents xxx As OPCGroup) in order for the results of the AsyncCancel operation to be returned to the automation client application. The AsyncCancelComplete event associated with the OPCGroup object will

be fired (called) by the automation server with the results of the AsyncCancel operation. The client specified transaction ID (TransactionID) will be returned to the automation client application in the AsyncCancelComplete event.

Example

```
Private Sub AsyncCancelButton_Click()
Dim ServerIndex As Long
Dim CancelID As Long
CancelID = 1 ‘ some transaction id returned from one of the
async calls like read, write, or refresh.
OneGroup.AsyncCancel CancelID
End Sub
```

OPCGroup Events

DataChange

Description The DataChange event is fired when a value or the quality of a value for an item within the group has changed. Note the event will not fire faster than the update rate of the group. Therefore, item values will be held by the server and buffered until the current time + update rate is greater than the time of the previous update (event fired).
This is also affected by active states for both Group and Items. Only items that are active, and whose group is active will be sent to the client in an event.

Syntax DataChange (TransactionID As Long, NumItems As Long, ClientHandles() As Long, ItemValues() As Variant, Qualities() As Long, TimeStamps() As Date)

TransactionID	The client specified transaction ID. A non-0 value for this indicates that this call has been generated as a result of an AsyncRefresh. A value of 0 indicates that this call has been generated as a result of normal subscription processing.
NumItems	The number of items returned
ClientHandles	Array of client item handles for the items
ItemValues	Array of values.
Qualities	Array of Qualities for each item's value.
TimeStamps	Array of UTC TimeStamps for each item's value. If the device cannot provide a timestamp then the server will provide one.

Remarks If the item values are changing faster than the update rate, only the most recent value for each item will be buffered and returned to the client in the event.

Example

```
Dim WithEvents AnOPCGroup As OPCGroup
Private Sub AnOPCGroup_DataChange (TransactionID As Long,
NumItems As Long, ClientHandles() As Long, ItemValues() As
Variant, Qualities() As Long, TimeStamps() As Date)
‘ write your client code here to process the data change values
End Sub
```

AsyncReadComplete

Description This event fires when an AsyncRead is completed.

Syntax AsyncReadComplete (TransactionID As Long, NumItems As Long, ClientHandles() As Long, ItemValues() As Variant, Qualities() As Long, TimeStamps() As Date, Errors() As Long)

TransactionID The client specified transaction ID.

NumItems The number of items returned

ClientHandles Array of client item handles for the items

ItemValues Array of values.

Qualities Array of Qualities for each item's value.

TimeStamps Array of UTC TimeStamps for each item's value. If the device cannot provide a timestamp then the server will provide one.

Errors Array of Long's indicating the success of the individual item reads. This indicates whether the read succeeded in obtaining a defined value, quality and timestamp. NOTE any FAILED error code indicates that the corresponding Value, Quality and Time stamp are UNDEFINED.

Example

```
Dim WithEvents AnOPCGroup As OPCGroup
Private Sub AnOPCGroup_AsyncReadComplete (TransactionID As
Long, NumItems As Long, ClientHandles() As Long, ItemValues()
As Variant, Qualities() As Long, TimeStamps() As Date)
' write your client code here to process the data change values
End Sub
```

AsyncWriteComplete

Description This event fires when an AsyncWrite is completed.

Syntax AsyncWriteComplete (TransactionID As Long, NumItems As Long, ClientHandles() As Long, Errors() As Long)

TransactionID The client specified transaction ID.

NumItems The number of items returned

ClientHandles Array of client item handles for the items

Errors Array of Long's indicating the success of the individual item writes.

Example

```
Dim WithEvents AnOPCGroup As OPCGroup
Private Sub AnOPCGroup_AsyncWriteComplete (TransactionID As
Long, NumItems As Long, ClientHandles() As Long, ItemValues()
As Variant, Qualities() As Long, TimeStamps() As Date)
' write your client code here to process the errors
End Sub
```

AsyncCancelComplete

Description This event fires when an AsyncCancel is completed.

Syntax AsyncCancelComplete (TransactionID As Long)

TransactionID The client specified transaction ID. This is included in the 'completion' information provided in the Corresponding Event.

Example

```
Dim WithEvents AnOPCGroup As OPCGroup
Private Sub AnOPCGroup_AsyncCancelComplete (TransactionID As
Long)
‘ write your client code here to process the cancel
End Sub
```

OPCItems Object

Description This object also has properties for OPCItem defaults. When an OPCItem is added, the DefaultXXXX properties set its initial state. The defaults can be changed to add OPCItems with different initial states. Of course, once an OPCItem is added, its properties can be modified. This reduces the number of parameters required to call the Add method.

Syntax OPCItems

Summary of Properties

Parent	DefaultRequestedDataType	DefaultAccessPath
DefaultIsActive	Count	

Summary of Methods

Item	GetOPCItem	AddItem
AddItems	Remove	Validate
SetActive	SetClientHandles	SetDataTypes

Example Syntax Base The following sample code is necessary for the subsequent Visual Basic Examples to be operational. This code is referred to as OPCItemsObjectBase.

```
Dim AnOPCServer As OPCServer
Dim ARealOPCServer As String
Dim ARealOPCNodeName As String
Dim AnOPCServerBrowser As OPCBrowser
Dim MyGroups As OPCGroups
Dim DefaultGroupUpdateRate As Long
Dim OneGroup As OPCGroup
Dim AnOPCItemCollection As OPCItems
Dim AnOPCItem As OPCItem
Dim ClientHandles(100) As Long
Dim AnOPCItemIDs(100) As String
Dim AnOPCItemServerHandles(10) As Long
Dim AnOPCItemServerErrors() As Long
Set AnOPCServer = New OPCServer
ARealOPCServer = “VendorX.DataAccessCustomServer”
ARealOPCNodeName = “SomeComputerNodeName”
AnOPCServer.Connect(ARealOPCServer, ARealOPCNodeName)
Set MyGroups = AnOPCServer.OPCGroups
MyGroups.DefaultGroupIsActive = True
Set OneGroup = MyGroups.Add( “AnOPCGroupName” )
Set AnOPCItemCollection = OneGroup.OPCItems
```

OPCItems Properties

Parent

Description (Read-only) Returns reference to the parent OPCGroup object.

Syntax Parent As OPCGroup

DefaultRequestedDataType

Description (Read/Write) The requested data type that will be used in calls to Add. This property defaults to VT_EMPTY (which means the server sends data in the server canonical data type).

Syntax DefaultRequestedDataType As Integer

Remarks Any legal Variant type can be passed as a requested data type.

Example

```
VB Syntax Example (getting the property):
Dim CurrentValue As Integer
Dim SomeValue As Integer
CurrentValue = AnOPCItemCollection.DefaultRequestedDataType
VB Syntax Example (setting the property):
AnOPCItemCollection.DefaultRequestedDataType = SomeValue
```

DefaultAccessPath

Description (Read/Write) The default AccessPath that will be used in calls to Add. This property defaults to “”.

Syntax DefaultAccessPath As String

Example

```
VB Syntax Example (getting the property):
Dim CurrentValue As String
Dim SomeValue As String
CurrentValue = AnOPCItemCollection.DefaultAccessPath
VB Syntax Example (setting the property):
AnOPCItemCollection.DefaultAccessPath = SomeValue
```

DefaultIsActive

Description (Read/Write) The default active state that will be used in calls to Add. This property defaults to True.

Syntax DefaultIsActive As Boolean

Example

```
VB Syntax Example (getting the property):
Dim CurrentValue As Boolean
Dim SomeValue As Boolean
CurrentValue = AnOPCItemCollection.DefaultIsActive
VB Syntax Example (setting the property):
AnOPCItemCollection.DefaultIsActive = SomeValue
```

Count

Description (Read-only) Required property for collections.

Syntax Count As Long

Example

```
VB Syntax Example (ge0ting the property):
Dim CurrentValue As Long
Dim SomeValue As Long
CurrentValue = AnOPCItemCollection.Count
```

OPCItems Methods

Item

Description Required property for collections.

Syntax Item (ItemSpecifier As Variant) As OPCItem

ItemSpecifier Returns an OPCItem by ItemSpecifier. ItemSpecifier is the 1-based index into the collection

Remarks Returns an OPCItem by ItemSpecifier. ItemSpecifier is the 1-based index into the collection. Use GetOPCItem to reference by ServerHandle.

NOTE: do not confuse the automation 'Item' property with the OPCItem object. The automation 'Item' is a special reserved property used in a generic way by automation collections to refer to the items they contain. The OPCItem is an OPC Automation specific object type that can reside in an 'OPCItems' collection.

GetOPCItem

Description Returns an OPCItem by ServerHandle returned by Add. Use the Item property to reference by index.

Syntax GetOPCItem (ServerHandle As Long) As OPCItem

ServerHandle ServerHandle is the OPCItem's ServerHandle
Use Item to reference by index.

Example

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
```

AddItem

Description Creates a new OPCItem object and adds it to the collection. The properties of this new OPCItem are determined by the current defaults in the OPCItems collection object. After an OPCItem is added, its properties can also be modified.

Syntax AddItem (ItemID As String, ClientHandle As Long)

ItemID Fully Qualified ItemID

ClientHandle Client handle that will be returned with the

Remarks This method is intended to provide the mechanism to add one item to the collection at a time. For adding multiple items use the AddItems method, rather than repetitively calling AddItem for each object to be added.

Example

```
Dim AnOPCItemID as String
Dim AnClientHandle as Long
AnOPCItemID = “N7:0”
AnClientHandle = 1975
AnOPCItemCollection.AddItem AnOPCItemID AnClientHandle
```

AddItems

Description Creates OPCItem objects and adds them to the collection. The properties of each new OPCItem are determined by the current defaults in the OPCItems collection object. After an OPCItem is added, its properties can also be modified.

Syntax AddItems (Count As Long, ItemIDs() As String, ClientHandles() As Long, ByRef ServerHandles() As Long, ByRef Errors() As Long, Optional RequestedDataTypes As Variant, Optional AccessPaths As Variant)

Count The number of items to be affected

ItemIDs Array of Fully Qualified ItemID’s

ClientHandles Array of client item handles for the items processed

ServerHandles Array of server item handles for the items processed

Errors Array of Long’s indicating the success of the individual items operation.

RequestedDataTypes Optional Variant containing an integer array of Requested DataTypes.

AccessPaths Optional Variant containing a string array of Access Path’s.

Example

```
Dim addItemCount as long
Dim AnOPCItemIDs() as String
Dim AnOPCItemServerHandles as long
Dim AnOPCItemServerErrors as long
Dim AnOPCRequestedDataTypes as variant
Dim AnOPCAccessPathss as variant
For  x = 1 To AddItemCount
  ClientHandles(x)  = x + 1
  AnOPCItemID(x) = “Register_”  &  x
Next x
AnOPCItemCollection.AddItems AddItemCount, AnOPCItemIDs,
ClientHandles,  AnOPCItemServerHandles, AnOPCItemServerErrors,
AnOPCRequestedDataTypes, AnOPCAccessPathss
‘ add code to process any errors that are returned from ‘the
method, individual errors are reported in the Errors array
```

Remove

Description Removes an OPCItem

Syntax Remove (Count As Long, ServerHandles() As Long, ByRef Errors() As Long)

Count The number of items to be removed

ServerHandles Array of server item handles for the items processed

Errors Array of Long's indicating the success of the individual items operation.

Example

```
AnOPCItemCollection.Remove  AnOPCItemServerHandles,
AnOPCItemServerErrors
' add code to process any errors that are returned from 'the
method, individual errors are reported in the Errors array
```

Validate

Description Determines if one or more OPCItems could be successfully created via the Add method (but does not add them).

Syntax Validate (Count As Long, ItemIDs() As String, ByRef Errors() As Long, Optional RequestedDataTypes As Variant, Optional AccessPaths As Variant)

Count The number of items to be affected

ItemIDs Array of Fully Qualified ItemID's

Errors Array of Long's indicating the success of the individual items operation.

RequestedDataTypes Variant containing an integer array of Requested DataTypes.

AccessPaths Variant containing a string array of Access Path's.

Example

```
Dim addItemCount as long
Dim AnOPCItemIDs() as String
Dim AnOPCItemServerHandles as long
Dim AnOPCItemServerErrors as long
Dim AnOPCRequestedDataTypes as variant
Dim AnOPCAccessPathss as variant

For  x = 1 To AddItemCount
  ClientHandles(x)  = x + 1
  AnOPCItemID(x) = "Register_"  &  x
Next x
AnOPCItemCollection.Validate AddItemCount, AnOPCItemIDs,
AnOPCItemServerErrors, AnOPCRequestedDataTypes,
AnOPCAccessPathss

' add code to process any errors that are returned from 'the
method, individual errors are reported in the Errors array
```

SetActive

Description Allows Activation and deactivation of individual OPCItem's in the OPCItems Collection

Syntax SetActive (Count As Long, ServerHandles() As Long, ActiveState As Boolean, ByRef Errors() As Long)

Count The number of items to be affected

ServerHandles Array of server item handles for the items processed

ActiveState TRUE if items are to be activated. FALSE if items are to be deactivated.

Errors Array of Long's indicating the success of the individual items operation.

Example

```
'set items to active (TRUE)
AnOPCItemCollection.SetActive ItemCount,
AnOPCItemServerHandles, TRUE, AnOPCItemServerErrors
' add code to process any errors that are returned from 'the
method, individual errors are reported in the Errors array
```

SetClientHandles

Description Changes the client handles or one or more Items in a Group.

Syntax SetClientHandles (Count As Long, ServerHandles() As Long, ClientHandles() As Long, ByRef Errors() As Long)

Count The number of items to be affected

ServerHandles Array of server item handles for the items processed

ClientHandles Array of new Client item handles to be stored. The Client handles do not need to be unique.

Errors Array of Long's indicating the success of the individual items operation.

Example

```
For  x = 1 To ItemCount
  ClientHandles(x)  = x + 1975
Next x
AnOPCItemCollection. SetClientHandles ItemCount,
AnOPCItemServerHandles, ClientHandles, AnOPCItemServerErrors
```

SetDataTypes

Description Changes the requested data type for one or more Items

Syntax SetDataTypes (Count As Long, ServerHandles() As Long, RequestedDataTypes() As Long, ByRef Errors() As Long)

Count The number of items to be affected

ServerHandles Array of server item handles for the items processed

RequestedDataTypes Array of new Requested DataTypes to be stored.

Errors Array of Long's indicating the success of the individual items operation.

Example

```
Dim RequestedDataTypes(100) As Long
For  x = 1 To ItemCount
  RequestedDataTypes (x)  = "some vbinteger"
Next x
AnOPCItemCollection.SetDataTypes ItemCount,
AnOPCItemServerHandles, RequestedDataTypes,
AnOPCItemServerErrors
```

OPCItem Object

Description An OPC Item represents a connection to data sources within the server. 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.

Syntax OPCItem

Summary of Properties

Parent	ClientHandle	ServerHandle
AccessPath	AccessRights	ItemID
IsActive	RequestedDataType	Value
Quality	TimeStamp	CanonicalDataType
EUType	EUInfo	

Summary of Methods

Read	Write	

OPCItem Properties

Parent

Description (Read-only) Returns reference to the parent OPCGroup object.

Syntax Parent As OPCGroup

ClientHandle

Description (Read/Write) A Long value associated with the OPCItem. Its purpose is for the client to quickly locate the destination of data. The handle is typically an index, etc. This handle will be returned to the client along with data or status changes by OPCGroup events.

Syntax ClientHandle As Long

Example

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
VB Syntax Example (getting the property):
Dim CurrentValue As Long
Dim SomeValue As Long
CurrentValue = AnOPCItem.ClientHandle
VB Syntax Example (setting the property):
AnOPCItem.ClientHandle = SomeValue
```

ServerHandle

Description (Read-only) The server assigned handle for the AnOPCItem. The ServerHandle is a Long that uniquely identifies this AnOPCItem. The client must supply this handle to some of the methods that operate on OPCItem objects (such as OPCItems.Remove).

Syntax ServerHandle As Long

Example

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
VB Syntax Example (getting the property):
Dim CurrentValue As Long
Dim SomeValue As Long
CurrentValue = AnOPCItem.ServerHandle
```

AccessPath

Description (Read-only) The access path specified by the client on the Add function..

Syntax AccessPath As String

Example

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
VB Syntax Example (getting the property):
Dim CurrentValue As String
Dim SomeValue As String
CurrentValue = AnOPCItem.AccessPath
```

AccessRights

Description (Read-only) Returns the access rights of this item.

Syntax AccessRights As Long

Remarks Indicates if this item is read only, write only or read/write.

Example

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
VB Syntax Example (getting the property):
Dim CurrentValue As Long
Dim SomeValue As Long
CurrentValue = AnOPCItem.AccessRights
```

ItemID

Description (Read-only) The unique identifier for this item.

Syntax ItemID As String

Example

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
VB Syntax Example (getting the property):
Dim CurrentValue As String
Dim SomeValue As String
CurrentValue = AnOPCItem.ItemID
```

IsActive

Description (Read/Write) State of the Data Acquisition for this item.

Syntax IsActive As Boolean

Remarks FALSE if the item is not currently active, TRUE if the item is currently active

Example

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
VB Syntax Example (getting the property):
Dim CurrentValue As Boolean
Dim SomeValue As Boolean
CurrentValue = AnOPCItem.IsActive
VB Syntax Example (setting the property):
AnOPCItem.IsActive = SomeValue
```

RequestedDataType

Description (Read/Write) The data type in which the item's value will be returned. Note that if the requested data type was rejected the OPCItem will be invalid(failed), until the RequestedDataType is set to a valid value.

Syntax RequestedDataType As Integer

Example

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
VB Syntax Example (getting the property):
Dim CurrentValue As Integer
Dim SomeValue As Integer
CurrentValue = AnOPCItem.RequestedDataType
VB Syntax Example (setting the property):
AnOPCItem.RequestedDataType = SomeValue
```

Value

Description (Read-only) Returns the latest value read from the server. This is the default property of AnOPCItem.

Syntax Value As Variant

Example

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
VB Syntax Example (getting the property):
Dim CurrentValue As Variant
Dim SomeValue As Variant
CurrentValue = AnOPCItem.Value
```

Quality

Description (Read-only) Returns the latest quality read from the server.

Syntax Quality As Long

Example

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
VB Syntax Example (getting the property):
Dim CurrentValue As Long
Dim SomeValue As Long
CurrentValue = AnOPCItem.Quality
```

TimeStamp

Description (Read-only) Returns the latest timestamp read from the server.

Syntax TimeStamp As Date

Example

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
VB Syntax Example (getting the property):
Dim CurrentValue As Date
Dim SomeValue As Date
CurrentValue = AnOPCItem.TimeStamp
```

CanonicalDataType

Description (Read-only) Returns the native data type in the server.

Syntax CanonicalDataType As Integer

Example

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
VB Syntax Example (getting the property):
Dim CurrentValue As Integer
Dim SomeValue As Integer
CurrentValue = AnOPCItem.CanonicalDataType
```

EUType

Description (Read-only) Indicate the type of Engineering Units (EU) information (if any) contained in EUInfo.

Syntax EUType As Integer

See Also OPCITEMATTRIBUTES in the OPC Data Access Custom Interface Specification.

Remarks 0 - No EU information available (EUInfo will be VT_EMPTY)
1 - Analog - EUInfo will contain a SAFEARRAY of exactly two doubles (VT_ARRAY | VT_R8) corresponding to the LOW and HI EU range.
2 - Enumerated - EUInfo will contain a SAFEARRAY of strings (VT_ARRAY | VT_BSTR) which contains a list of strings (Example: “OPEN”, “CLOSE”, “IN TRANSIT”, etc.) corresponding to sequential numeric values (0, 1, 2, etc.)

Example

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
VB Syntax Example (getting the property):
Dim CurrentValue As Integer
Dim SomeValue As Integer
CurrentValue = AnOPCItem.EUType
```

EUInfo

Description (Read-only) Variant that contains the Engineering Units information

Syntax EUInfo As Variant

See Also OPCITEMATTRIBUTES in the OPC Data Access Custom Interface Specification.

Example

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
VB Syntax Example (getting the property):
Dim CurrentValue As Variant
Dim SomeValue As Variant
CurrentValue = AnOPCItem.EUInfo
```

OPCItem Methods

Read

Description Read makes a blocking call to read this item from the server.
Read can be called with only a source (either OPCCache or OPCDevice) to refresh the item's value, quality and timestamp properties. If the value, quality and timestamp must be in sync, this method's optional parameters return values that were acquired together.

Syntax Read (Source As Integer, Optional ByRef Value As Variant, Optional ByRef Quality As Variant, Optional ByRef TimeStamp As Variant)

Source	The 'data source'; OPC_DS_CACHE or OPC_DS_DEVICE
Value	Returns the latest value read from the server
Quality	Returns the latest value read from the server
TimeStamp	Returns the latest timestamp read from the server.

Example

```
Private Sub ReadButton_Click()
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
Dim Source As Integer
Dim Value As Variant
Dim Quality As Variant
Dim TimeStamp As Variant
Source = OPC_DS_DEVICE
AnOPCItem.Read Source, ServerHandles, Value, Quality, TimeStamp
 ' process the values
TextBox.Text = Value
End Sub
```

Write

Description Write makes a blocking call to write this value to the server.

Syntax Write (Value As Variant)

Value	Value to be written to the data source.

Example

```
Private Sub WriteButton_Click()
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
Dim Value As Variant
Value = 1975
AnOPCItem.Write Value
End Sub
```