

Computing

OPC Server Interface

Manual

OPC Custom Interface

1

OPC Automation Interface

2

This manual is based on version 2.0 of the OPC specification from the OPC Foundation.

This manual is available only in English.

C79000–G7076–C225–01

Edition: 3

Safety Guidelines

This manual contains notices which you should observe to ensure your own personal safety, as well as to protect the product and connected equipment. These notices are highlighted in the manual by a warning triangle and are marked as follows according to the level of danger:



Danger

indicates that death, severe personal injury or substantial property damage **will** result if proper precautions are not taken.



Warning

indicates that death, severe personal injury or substantial property damage **can** result if proper precautions are not taken.



Caution

indicates that minor personal injury or property damage can result if proper precautions are not taken.

Note

draws your attention to particularly important information on the product, handling the product, or to a particular part of the documentation.

Qualified Personnel

Only **qualified personnel** should be allowed to install and work on this equipment. Qualified persons are defined as persons who are authorized to commission, to ground, and to tag circuits, equipment, and systems in accordance with established safety practices and standards.

Correct Usage

Note the following:



Warning

This device and its components may only be used for the applications described in the catalog or the technical descriptions, and only in connection with devices or components from other manufacturers which have been approved or recommended by Siemens.

This product can only function correctly and safely if it is transported, stored, set up, and installed correctly, and operated and maintained as recommended.

Trademarks

SIMATIC®, SIMATIC HMI® and SIMATIC NET® are registered trademarks of SIEMENS AG.

Some of other designations used in these documents are also registered trademarks; the owner's rights may be violated if they are used by third parties for their own purposes.

Copyright Siemens AG 1999 All rights reserved

The reproduction, transmission or use of this document or its contents is not permitted without express written authority. Offenders will be liable for damages. All rights, including rights created by patent grant or registration of a utility model or design, are reserved.

Siemens AG
Automation and Drives (A&D)
Industrial Automation Systems (AS)
Postfach 4848, D- 90327 Nürnberg

Disclaimer of Liability

We have checked the contents of this manual for agreement with the hardware and software described. Since deviations cannot be precluded entirely, we cannot guarantee full agreement. However, the data in this manual are reviewed regularly and any necessary corrections included in subsequent editions. Suggestions for improvement are welcomed.

© Siemens AG 1999
Technical data subject to change.

Contents

1	OPC Custom Interface	1-1
1.1	Creating and Using an OLE Object in C/C++	1-2
1.2	Additional Information about the Interface Description for the OPC Custom Interface	1-5
1.3	The “OPC Server” Object	1-6
1.4	Objects of the “OPC Group” Class	1-11
1.5	IDataObject Interface	1-16
2	OPC Automation Interface	2-1
2.1	Creating and Using an OLE Object in Visual Basic	2-2
2.2	Object Model for the Automation Interface	2-5
2.3	The “OPCServer” Object	2-6
2.4	The “OPCBrowser” Object	2-8
2.5	The “OPCGroups” Collection Object	2-10
2.6	The “OPCGroup” Object	2-12
2.7	The “OPCItems” Collection Object	2-15
2.8	The “OPCItem” Object	2-17
 Figures		
1-1	OPC Server Object	1-6
1-2	OPC Group Object	1-11
1-3	IAdviseSink (Client) and IDataObject (Server) Interfaces	1-16
2-1	Activating the Reference for the Automation Interface	2-2
2-2	Object Model for the Automation Interface	2-5
 Tables		
1-1	Objects and Interfaces of the OPC Custom Interface	1-5
2-1	Properties of the “OPCServer” Object	2-6
2-2	Properties of the “OPCBrowser” Object	2-8
2-3	Properties of the “OPCGroups” Object	2-10
2-4	Properties of the “OPCGroup” Object	2-12
2-5	Properties of the “OPCItems” Collection Object	2-15
2-6	Properties of the “OPCItem” Object	2-17

OPC Custom Interface

Chapter Overview

This chapter shows how to use the OPC custom interface. It also lists the interfaces and methods of the OPC customer interface. This is not a detailed interface description but contains supplementary information and notes relating specifically to the Computing software.

There is now an extended version of the OPC custom interface, specification 2.0. Version 2.0 supplements the existing OPC custom interface in several aspects particularly those simplifying the handling of asynchronous communication.

The interfaces of Version 1.0 of the OPC interface are completely upwards compatible with Version 2.0.

Section	Description	Page
1.1	Creating and Using an OLE Object in C/C++	1-2
1.2	Additional Information about the Interface Description for the OPC Custom Interface	1-5
1.3	The “OPC Server” Object	1-6
1.4	Objects of the “OPC Group” Class	1-11
1.5	IDataObject Interface	1-16

1.1 Creating and Using an OLE Object in C/C++

The following sections illustrate step-by-step how you can call the methods of an instance of an OLE class in C++. Note the difference between the term “Class” in OLE and in C++:

- OLE Classes: A Windows object is an instance of an OLE class. The term OLE class differs from the class in C++.
- C++ Classes: A class in C++ is a type definition. An OLE class is, however, an object description and does not contain types.

Class Identification Code

Each OLE class can be identified uniquely by a 128-bit long identification code, the CLSID. This is used by the operating system for the unique assignment of a DLL or EXE file that implements this class. A client that wishes to use an object of a class requires only the CLSID.

ProgID

To simplify the identification of OPC servers, there is normally a readable name, the ProgID assigned in the CLSIDs. While a CLSID is always unique due to the algorithm for compilation, it is possible that a ProgID exists more than once. Just like the CLSID, the ProgID is specified by the vendor of an OPC server.

The ProgID for the OPC server of Computing is: `OPCServer.WinAC`

Creating a COM Object

A COM object is created in five steps:

1. Initialize COM.
2. Query the CLSID.
3. Create an Object.
4. Call an OPC function.
5. Release the interfaces used.

Step 1: Initialize COM

Before you can use the functions of COM, the COM library must be initialized with the following call:

```
HRESULT r1;  
r1 = CoInitialize(NULL);
```

Step 2: Query the CLSID

If the name of an object is known, the CLSID can be queried using the OLE function "CLSIDFromProgID".

Example: The following program section illustrates how to query the CLSID for the OPC server for Computing.

```
CLSID clsid; // Get the CLSID from the Name
r1 = CLSIDFromProgID(L"OPCServer.WinAC",&clsid);
```

Step 3: Create an Object

If a client wants to use an object, it transfers the CLSID to the operating system and requests an object instance. Regardless of where the server is located, the object request is always directed to COM.

The "CoCreateInstance" function creates an object belonging to the required class. This function includes certain intermediate steps via the "IClassFactory" interface. Creating an object using IClassFactory is more efficient when several objects of a class must be created.

Example: The following lines show how an object of the class "OPC server" with reference to the "IUnknown" interface can be created.

```
IUnknown * pOPCUnknown;
r1 = CoCreateInstance (clsid, NULL, CLSCTX_LOCAL_SERVER,
IID_IUnknown, (void**) &pOPCUnknown );
```

Step 4: Call an OPC Function

In this step, a method of the "IOPCServer" interface of the created object will be used to learn the status of the server. First, a pointer to the "IOPCServer" interface is made available via "IUnknown". Finally, the "GetStatus" method is called.

Example: The program shown below outputs the status of the server and the vendor information. If the "GetStatus" method is called successfully, the OPC server allocates the memory areas for the return information via the "IMalloc" interface. The user must release these memory areas again.

```
IOPCServer *pOPCServer;
OPCSERVERSTATUS *pss;
r1 = pOPCUnknown->QueryInterface(IID_IOPCServer,
(void**)&pOPCServer);
r1 = pOPCServer ->GetStatus(&pss);
printf("Status.szVendorInfo = %ls\n", pss-> szVendorInfo);
// Remember to release the memory returned by the method
pIMalloc->Free (pss->szVendorInfo);
pIMalloc->Free (pss);
```

Step 5: Release the Interfaces Used

Objects include reference counters to detect when the object is no longer required and can remove itself from memory. Each time the “QueryInterface” function is called, the reference counter is incremented. To release the object, the counter must be reset.

Example: Using the following commands, the reference counter for the interfaces “IUnknown” and “IOPCServer” is reset.

```
pOPCServer->Release();  
pOPCUnknown->Release();
```


1.2 Additional Information about the Interface Description for the OPC Custom Interface

A comprehensive description of the OPC interfaces is beyond the scope of this manual. This is supplied as a file along with this product as an original English document from the OPC Foundation. The documents are located in the "DOC" directory in the product directory of the OPC server for Computing.

The following supplementary information about the interfaces lists the objects of OPC, their interfaces and the methods defined in these interfaces and points out particular characteristics of the OPC server for Computing.

Return Values

All the listed methods return a result of the type HRESULT.

Overview of the Objects and Interfaces

Table 1-1 Objects and Interfaces of the OPC Custom Interface

Object	Interface
OPCServer	IOPCServer
	IOPCServerPublicGroups (optional)
	IOPCBrowseServerAddressSpace (optional)
	IOPCItemProperties (new with V 2.0)
	IConnectionPointContainer (new with V 2.0)
	IOPCCommon (new with V 2.0)
	IPersistFile (optional)
OPCGroup	IOPCGroupStateMgt
	IOPCPublicGroupStateMgt (optional)
	IOPCASyncIO2 (new with V 2.0)
	IOPCASyncIO (no longer necessary with V2.0)
	IOPCItemMgt
	IConnectionPointContainer (new with V 2.0)
	IOPCSyncIO
	IDataObject (no longer necessary with V2.0)
EnumOPCItemAttributes	IEnumOPCItemAttributes

1.3 The “OPC Server” Object

The OPC server class has various attributes that contain information about the status, the version etc. of an OPC server object. The OPC server class also has methods with which a client can manage the objects of the OPC group class. A client application addresses only an object of this class directly using COM mechanisms. The other objects are created by corresponding OPC methods.

The methods of the IOPCServer interface are used to manage the objects in the OPC group class. Using the methods of the IOPCBrowseServerAddressSpace interface, it is possible to investigate the address area of the server.

Figure 1-1 illustrates the “OPC server” object with its interfaces.

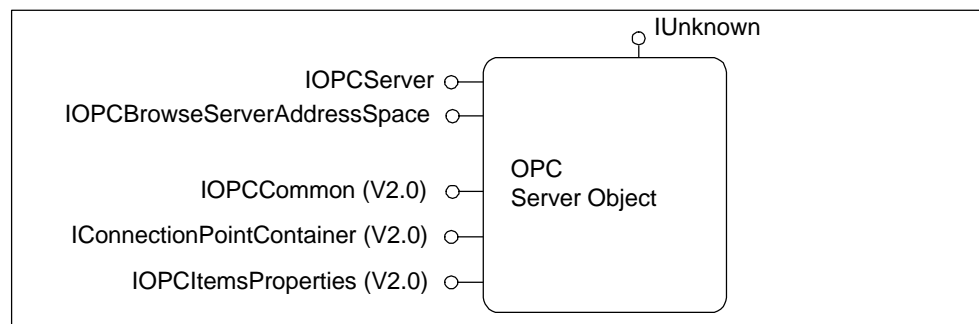


Figure 1-1 OPC Server Object

IOPCServer Interface

This interface contains methods to manage groups within a server object. It is also possible to obtain information about the current status of the server.

```
AddGroup ( szName, bActive, dwRequestedUpdateRate, hClientGroup,
pTimeBias, pPercentDeadband, dwLCID, phServerGroup,
pRevisedUpdateRate, riid, ppUnk )
```

Creates a group in the server object.

Notes:

- The "pTimeBias" parameter is not evaluated by the OPC server for Computing.
- "LCID" is irrelevant for the OPC server for Computing.
- The "UpdateRate" is specified by the configuration parameter "Minimum Update Rate" as a multiple of the configuration value.
- The "pPercentDeadband" parameter is only effective for variables of the real (VT_R4) type.
- If the "szName" parameter is empty, a name is generated beginning with the underscore character (for example, "_123456"). User-defined names should therefore not begin with the underscore character.

CreateGroupEnumerator (dwScope, riid, ppUnk)

Creates various enumerators for the group.

Note: Since there are no public groups in the OPC server for Computing, the return values for the parameter "dwScope" "...PRIVATE" and "...PUBLIC" are identical.

GetErrorString (dwError, dwLocale, ppString)

Supplies the error message for a specific error code.

Note: The OPC server for Computing supports German and English error texts. Errors detected by the Windows operating system are explained in the language in which the operating system was installed.

GetGroupByName (szName, riid, ppUnk)

Supplies an additional interface pointer for the name of a private group, in other words the reference counter is incremented.

GetStatus (ppServerStatus)

Supplies the status information of the server.

Note: The return value of the OPC server for Computing is the name and the version of the OPC server.

RemoveGroup (hServerGroup, bForce)

Deletes a group on the server.

Note: The OPC server for Computing does not support the use of the "bForce" parameter. It is not possible to delete groups to which references are still active.

IOPCBrowseServerAddressSpace Interface

This interface contains methods with which the address area of the server can be queried. The address area contains all the OPC items known to the server.

BrowseAccessPaths (szItemID, ppIEnumString)

This provides the possibility of querying the access path of an ItemID.

Note: Not required with the OPC server for Computing.

BrowseOPCItemIDs (dwBrowseFilterType, szFilterCriteria, vtDataTypeFilter, dwAccessRightsFilter, ppIEnumString)

Supplies a string of the type "IEnumString" whose content is specified by the call parameters. The position from which the list is created can be set using the "ChangeBrowsePosition" method.

Notes:

- "BRANCH" excludes the filters for Type and AccessRights.
- The rules for creating a filter are as follows:
 - Asterisk (*) Any character string, including empty strings
 - Plus (+) Any character string, however at least one character
 - Question marks (?) Any single character
 - Square brackets ([]) One single character from the specified set
- To use one of the filter characters, this must be preceded by a back slash (\).

ChangeBrowsePosition (dwBrowseDirection, szString)

Allows you to browse through the address area. You can change to the higher level or to a branch.

GetItemID (szItemDataID, szItemID)

Creates a complete ItemID in the hierarchical address area. This function is necessary since browsing itself only provides the designations below the current node.

Note: The description of GetItemID in OPC specification is inconsistent with the description of ChangeBrowsePosition. With ChangeBrowsePosition, it is not possible to specify a complete ItemID. For this reason, the OPC server for Computing only currently supports the command GetItemID for single leaves (LEAF).

QueryOrganization (pNamespaceType)

Supplies the structure of the address area. The address area can be organized with a flat or hierarchical structure.

Note: The structure of the address area of the OPC server for Computing is structured hierarchically.

IOPCCommon Interface (Version 2.0)

This interface of version 2.0 of the OPC Custom Interface contains methods allowing the language settings and the name of the client to be made known to the server.

SetLocaleID (dwLcid)

Sets the language code of the server. The language code specifies the language in which the server outputs text.

Note: The OPC server for Computing supports English and German.

GetLocaleID (pdwLcid)

Fetches the language code of the server.

Note: The OPC server for Computing supports English and German.

QueryAvailableLocaleIDs (pdwLcid)

Provides all the available language codes of the server.

Note: The OPC server for Computing supports English and German.

GetErrorString (dwError, ppString)

Provides the error text for a specific error code in the set language.

SetClientName (szName)

Transfers a descriptive text for the client to the server. The descriptive text can be used for any purpose by the server, for example for logging in trace files.

IConnectionPointContainer Interface

This interface is a standard COM interface for reporting asynchronous events via connection points. For more detailed information about using connection points, refer to the documentation of OLE/COM.

IOPCItemProperties (V 2.0) Interface

This interface of version 2.0 contains methods allowing specific server information to be queried about an item.

QueryAvailableProperties (szItemID, pdwCount, ppPropertyIDs, ppDescriptions, ppvtDataTypes)

Returns a list of available properties for an item.

```
GetItemProperties (szItemID, dwCount, pdwPropertyIDs, ppvData,  
ppErrors )
```

Provides the values of the properties of an item transferred in a list of PropertyIDs.

```
LookupItemIDs (szItemID, dwCount, pdwPropertyIDs, ppszNewItemIDs,  
ppErrors);
```

Provides (whenever possible for the propertyID) a list of ItemIDs for a list of PropertyIDs. These ItemIDs can be included in a group simplifying and speeding up access to the data.

Note: The OPC server for Computing does not support this function. The call is rejected with error message 0x8004001 (not implemented).

1.4 Objects of the “OPC Group” Class

The “OPC Group” class manages the individual process variables, the OPC items. Using these group objects, a client can form semantically meaningful units of OPC items and execute operations with them.

Figure 1-2 illustrates an object of the “OPC Group” class and its interfaces.

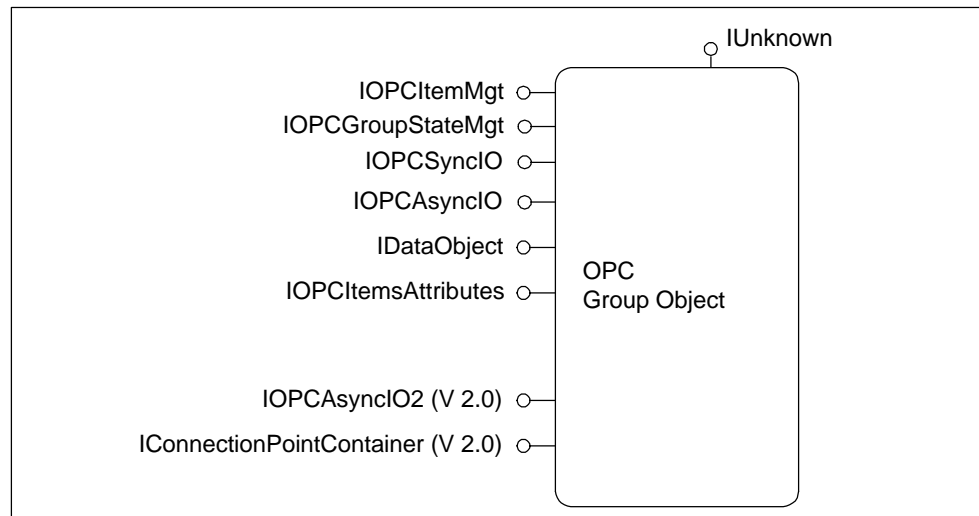


Figure 1-2 OPC Group Object

IOPCItemMgt Interface

This interface provides methods to manage more than one item in a group.

Time Stamp

With each value read, OPC supplies a time stamp. This indicates when this value was received or when it was changed. Since the SIMATIC systems do not use a time stamp, the time at which the value is received on the server is used as the time stamp.

AddItems (dwNumItems, pItemArray, ppAddResults, ppErrors)

Adds one or more items to a group.

Notes:

- A unique name of an item for the OPC server for Computing is as follows:

Examples MD0:Real
 EB0
 DB1.DB00

This name can be specified completely in the ItemID. The AccessPath must then be empty.

- As an alternative, the part of the name in square brackets can be included in the AccessPath.

Example: AccessPath: ""
 MD0:Real

- It is possible to add the same OPC item to the same group more than once. In this case each of these items nevertheless has its own server handle.

The server handles of the items are only unique within a group and not for all items of all groups.

- Valid data types are as follows: VT_UI1, VT_UI2, VT_UI4, VT_I1, VT_I2, VT_I4, VT_R4, VT_BOOL, VT_BSTR

CreateEnumerator (riid, ppUnk)

Creates an enumerator for the items of a group.

RemoveItems (dwNumItems, phServer, ppErrors)

Deletes one or more items from a group.

SetActiveState (dwNumItems, phServer, bActive, ppErrors)

Sets the active state of one or more items in a group.

SetClientHandles (dwNumItems, phServer, phClient, ppErrors)

Sets the client handle of one or more items in a group.

SetDatatypes (dwNumItems, phServer, pRequestedDatatypes, ppErrors)

Sets the requested data type of one or more items in a group.

Note: See AddItem


```
ValidateItems ( dwNumItems, pItemArray, bBlobUpdate,
ppValidationResults, ppErrors )
```

Checks the validity of an OPC item, for example whether it was added to a group without any error occurring, and supplies information such as the canonical data type.

Note: See AddItem

IOPCGroupStateMgt Interface

The IOPCGroupStateMgt interface provides methods with which groups can be managed. It is possible to edit group-specific parameters and to copy groups.

```
CloneGroup ( szName, riid, ppUnk )
```

Creates a copy of a group. All group attributes are copied except for the following:

- The active state is set to FALSE
- A new server handle is assigned

Note: The “szName” parameter can be empty. In this case a unique name is generated (see AddGroup).

```
GetState ( pUpdateRate, pActive, ppName, pTimeBias, pPercentDeadband,
pLCID, phClientGroup, phServerGroup )
```

Fetches the status of the group. The client application must inform the OPC server where the results are to be stored using a pointer.

Notes:

- The “pTimeBias” parameter has no significance for the OPC server for Computing.
- The “pPercentDeadband” parameter has no significance for the OPC server for Computing.
- The “LCID” parameter, in other words language-specific textual values in read/write, has no significance for SIMATIC variables.

```
SetName ( szName )
```

Allows the name of a group to be changed. The name must always be unique.

```
SetState ( pRequestedUpdateRate, pRevisedUpdateRate, pActive,
pTimeBias, pPercentDeadband, pLCID, phClientGroup )
```

SetState allows various properties of the group to be changed.

Notes:

- The “pTimeBias” parameter has no significance for the OPC server for Computing.
- The “pPercentDeadband” parameter has no significance for the OPC server for Computing.
- The “LCID” parameter, in other words language-specific textual values in read/write, has no significance for SIMATIC variables.
- The “UpdateRate” is specified by the configuration parameter “Minimum Update Rate” as a multiple of this value.

IOPCSyncIO Interface

This interface provides methods for synchronous reading and writing. Synchronous means that the client waits until the read or write operation is completed and only then continues execution.

The use of synchronous calls is recommended when the client requires the result for further processing. Other clients are not blocked since the OPC server for Computing starts a separate thread for each client.

In general, it is advisable to use the IData interface for processing variable changes (or IAdviseSink on the client side). This interface guarantees the highest possible data throughput and also reduces the actual number of calls to the absolute minimum (only when changes occur).

```
Read ( dwSource, dwNumItems, phServer, ppItemValues, ppErrors )
```

Reads the values, status information or time stamp of one or more items in a group. The values can be read from the cache of the server or directly from the hardware. Reading from the cache is, however, only possible when the group is activated.

Note: The call is monitored by the timeout monitoring on the server. The corresponding configuration parameter is “Read/Write Timeout”.

```
Write ( dwNumItems, phServer, pItemValues, ppErrors )
```

Writes values for one or more items of a group to the hardware.

Note: The call is monitored by the timeout monitoring on the server. The corresponding configuration parameter is “Read/Write Timeout”.

IOPCAsyncIO Interface

This interface of the Group class provides methods for asynchronous reading and writing of items. Asynchronous means that the client triggers a read or write operation and then continues operation. Asynchronous operations provide a transaction ID. When the server has completed the read or write operation, the client receives a message sent to its IAdviseSink interface.

Cancel (dwTransactionID)

Cancels an outstanding job.

Read (dwConnection, dwSource, dwNumItems, phServer, pTransactionID, ppErrors)

Sends an asynchronous read command. The result is sent to the IAdviseSink interface of the client.

Note: The call is monitored by the timeout monitoring on the server. The corresponding configuration parameter is "Read/Write Timeout". If the timeout monitoring is aborted, there is a callback with hrStatus=E_ABORT.

Refresh (dwConnection, dwSource, pTransactionID)

Requests a current value for every active OPC item.

Write (dwConnection, dwNumItems, phServer, pItemValues, pTransactionID, ppErrors)

Sends an asynchronous write command.

Note: The call is monitored by the timeout monitoring on the server. The corresponding configuration parameter is "Read/Write Timeout". If the timeout monitoring is aborted there is a callback with hrStatus=E_ABORT.

1.5 IDataObject Interface

The IDataObject interface is the standard interface of OLE for data transmission. It contains methods for establishing a message connection between the client and a server group.

Description of the Mechanism

If the server wants to send a message to a client, the client must provide a partner for the server. This partner is the IAdviseSink interface of the client. A server sends a message to a client by calling the OnDataChange method of the IAdviseSink interface of the client.

Representation of the Mechanism

Figure 1-3 illustrates how the "IAdviseSink" interface on the client and "IDataObject" on the server interact.

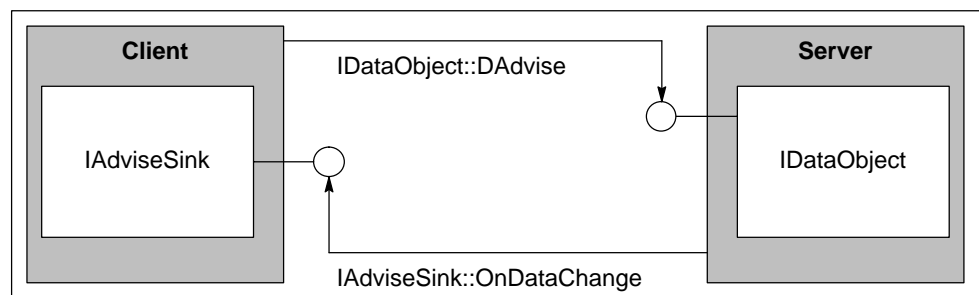


Figure 1-3 IAdviseSink (Client) and IDataObject (Server) Interfaces

DAdvise (pFmt, adv, pSnk, pConnection);

Establishes a connection between a server and the client. If a client wants to receive a message, it must establish a connection using this method. It transfers a pointer to its IAdviseSink interface to the server.

DUnadvise (Connection)

Terminates an existing connection between a client and server.

IEnumOPCItemAttributes Interface

This interface based on the IEnum standard interface returns the items of a group. The interface is supplied only by "IOPCItemMgr:CreateEnumerator". It is not obtainable with QueryInterface.

Clone (ppEnumItemAttributes);

Creates an identical copy of the enumerator.

Next (celt, ppItemArray, pceltFetched);

Fetches the next OPC item of the group.

Note: The OPC server for Computing does not support engineering units. EUNType and EUNInfo are therefore irrelevant.

Reset (void);

Resets the list to the first item of the group.

Skip (celt);

Skips a number of items in the list.

IAsyncIO2 Interface (Version 2.0)

This interface in version 2.0 provides methods for asynchronous reading and writing of items. Asynchronous means that the client triggers a read or write operation and then continues operation. Version 2 of the interface for asynchronous communication uses connection points. This simplifies the processing of the transferred data.

Read (dwCount, phServer, dwTransactionID, pdwCancelID, ppErrors)

Sends an asynchronous read command. The result is sent to the client via a connection point.

Note: The call is monitored by the timeout monitoring on the server. If the set time is exceeded, this is indicated by the status E_ABORT.

Write (dwCount, phServer, pItemValues, dwTransactionID, pdwCancelID, ppErrors)

Sends an asynchronous write command. The message indicating completion of the job comes via the specified connection point.

Note: The call is monitored by the timeout monitoring on the server. If the set time is exceeded, this is indicated by the status E_ABORT.

Cancel12 (dwCancelID)

Cancels an outstanding job.

Refresh (dwSource, dwTransactionID, pdwCancelID)

Requests a current value for every active OPC item.

SetEnable (bEnable)

Activates messages via connection points. Messages generated by the Refresh method are sent regardless of these settings.

GetEnable (pbEnable)

Returns the current value of the flag for messages via connection points.

IConnectionPointContainer Interface

This interface is a standard COM interface for reporting asynchronous events via connection points. For more detailed information about using connection points, refer to the documentation of OLE or COM.

OPC Automation Interface

2

Chapter Overview

This chapter explains how to use the OPC automation interface. It also lists the properties and methods of the OPC automation interface. This is not a detailed interface description but contains supplementary information and notes relating specifically to the OPC server of Computing.

Versions of the Automation Interface

The OPC automation interface was specified by the OPC Foundation in version 1.0. The specification, however, was unclear in some aspects. Some weaknesses of version 1.0 became particularly clear with the introduction of Visual Basic 5.0.

Since mid-1998, there is a new version of the specification of the automation interface available with version number 2.0. The extent to which this version is approved by the OPC Foundation can be found in the product information.

The following descriptions relate to version 2.0 of the specification of the OPC automation interface.

Section	Description	Page
2.1	Creating and Using an OLE Object in Visual Basic	2-2
2.2	Object Model for the Automation Interface	2-5
2.3	The "OPCServer" Object	2-6
2.4	The "OPCBrowser" Object	2-8
2.5	The "OPCGroups" Collection Object	2-10
2.6	The "OPCGroup" Object	2-12
2.7	The "OPCItems" Collection Object	2-15
2.8	The "OPCItem" Object	2-17

2.1 Creating and Using an OLE Object in Visual Basic

Visual Basic from Microsoft is a development environment that supports the automation interface for the simple linking of OLE objects. The following sections show how the OPC server is used via the automation interface 2.0 in Visual Basic. At least version 4 of Visual Basic is required.

Tasks for Creating an OPC Object

Create a new Visual Basic project. Select the **Project ► References** menu command to display the “References” dialog box. As shown in Figure 2-1, activate the reference for the automation interface 2.0 of the OPC server.

Creating an OPC Object in Visual Basic uses the following five basic tasks:

1. Declaring the Variables
2. Connecting to the OPC Server
3. Generating an OPC Group
4. Adding OPC Items
5. Synchronous Reading

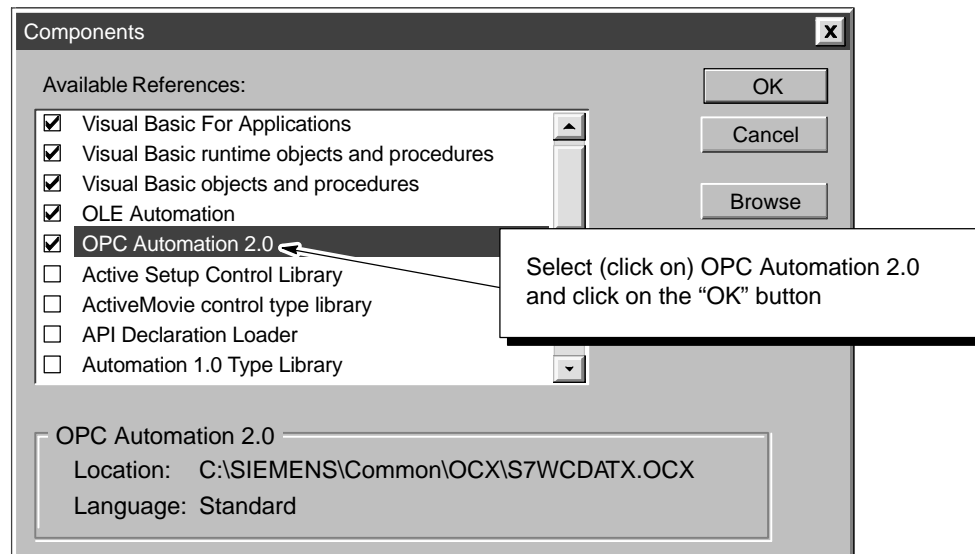


Figure 2-1 Activating the Reference for the Automation Interface

Step 1: Declaring a Variable of the Object Type

In Visual Basic or Visual Basic for Applications, a variable of the type Object refers to an OLE object. The DIM statement declares an object instance. Example:

```
Dim ObjServer As OPCServer
```


Step 2: Creating and Assigning the Object

Creating an OPC Server Object: Visual Basic programs are also OPC clients. To be able to access a process variable, the client must first create a server object and then connect to an OPC server.

Example: The following code section shows how a Visual Basic client can connect to an OPC server object.

```
Set ObjServer = New OPCServer
ObjServer.Connect ("OPCServer.WinAC")
```

Step 3: Generating an OPC Group

The next step is to create a group object to accommodate process variables.

First, the group object and then the collection object are declared. The declared collection object is then assigned OPCGroups, a property of the server object. A group is finally created by calling the Add method of the collection object.

Example: The following lines show how a group is created in Visual Basic.

```
'Declarations
Dim GroupObj As OPCGroup
Dim GroupCollection As OPCGroups
Set GroupCollection = ObjServer.OPCGroups
Set GroupObj = GroupCollection.Add("MyGroup")
```

Step 4: Adding OPC Items

OPC items will now be inserted in the created group object. The items represent the connections to the process variables, their parameter ItemID specified which variable will be addressed.

The AddItems method allows several items to be inserted into a group in one call. The transfer parameters and the return values are therefore one-dimensional arrays with identical sizes. The "IItems" variable contains the number of items to be inserted.

When the method is called, the server initializes the values of the arrays "IServerHandles", "IErrors" and "ItemsObj". The "IErrors" array contains status information for each item inserted that indicated whether or not the item was successfully inserted.

Example: The following example creates two items in the previously created OPC group "GrpObj". The first item represents MD0, and the second item represents MD4.

```
'Declaration
    Dim ItemCollection As OPCItems
    Dim ItemServerHandle() As Long
    Const MAX_INDEX = 2
    Dim lNumItems As Long
    Dim lClientHandles(MAX_INDEX) As Long
    Dim perror() As Long
    Dim szItemIDs(MAX_INDEX) As String
    Dim AccPath(MAX_INDEX) As String
    Dim ReqDataTypes(MAX_INDEX) As Integer
'Definition of ItemIDs
    szItemIDs(1) = "MD0:Real"
    szItemIDs(2) = "MD4:Real"
    AccPath(1) = ""
    AccPath(2) = ""
    ReqDataTypes(1) = vbVLong
    ReqDataTypes(2) = vbVString
    lClientHandles(1) = 1
    lClientHandles(2) = 2
'Add Items to Group
    Set ItemCollection = GroupObj.OPCItems
    ItemCollection.Add MAX_INDEX, szItemIDs, lClientHandles, _
ItemServerHandle, perror, ReqDataTypes, AccPath
```

Step 5: Synchronous Reading

In the last step, a synchronous read operation to read the process variables of a group is executed. Synchronous means that the server only returns control to the Visual Basic program after the required results have been returned via the communications system. This means that the flow of communication is delayed by the time required for this communication.

Example: Just like the AddItems command, OPCRead is also a group operation, in other words several process variables within a group can be accessed with one call. The "lNumItems" parameter specifies the number of variables to be read. The individual variables themselves are described in the array "lServerHandles" by the handle assigned by the server.

```
'Definition of the Variables for OPCRead
'Out Parameter
    Dim vValues() As Variant
    Dim pErrors() as Long
    GroupObj.SyncRead OPCDevice, 2, ItemServerHandle, vValues, pErrors
```

2.2 Object Model for the Automation Interface

The object model for the OPC automation interface according specification 2.0 differs from the model described in Section 5.3: Separate collection objects manage the objects OPC-Group and OPC-Item. The collection objects provide functions for counting the objects assigned to them. The browsing functions are also brought together in a separate object.

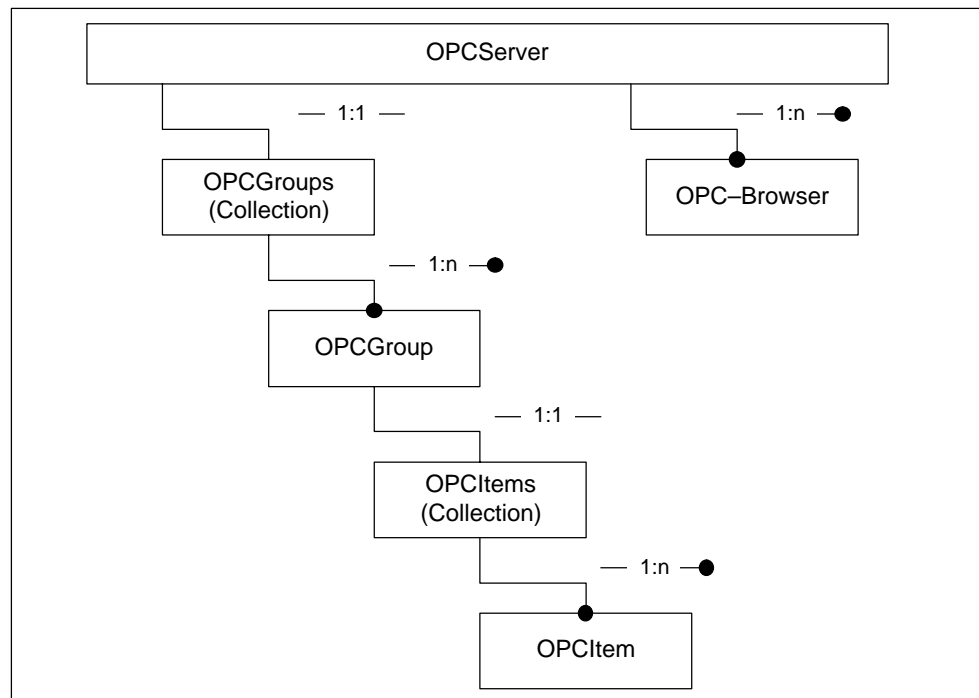


Figure 2-2 Object Model for the Automation Interface

2.3 The “OPCServer” Object

Objects of the OPC server class are created by the client. The properties of an OPC server contain general information about the server. When an OPC server object is created, an OPCGroup collection is also created as a property of the OPC server object.

Properties of “OPCServer”

Table 2-1 Properties of the “OPCServer” Object

Property	Type	Brief Description
StartTime	Date	Time at which the server was started (in UTC)
CurrentTime	Date	The current time (UTC), as supplied to the server by the system
LastUpdateTime	Date	The time (UTC), at which the server sent the last update of data to the client
MajorVersion	Integer	The major version number of the server
MinorVersion	Integer	The minor version number of the server
BuildNumber	Integer	The build number of the server
VendorInfo	String	The vendor information
ServerState	Long	Status of the server
Bandwidth	Long	Bandwidth of the server
OPCGroups	OPC Groups	A collection of OPCGroup objects
PublicGroup Names	Variant	The names of the public groups provided by this server
OPCServers	Variant	The names (ProgID) of the registered OPC servers. Use one of these names for the “Connect” method. The names are returned as an array of strings.

Notes:

- The OPC server for Computing provides the following as vendor information: “Computing OPC–Server”
- Public groups are not supported by the OPC server for Computing.
- The Bandwith property is not supported by the OPC server for Computing.

Methods of the “OPCServer”

Connect (ProgID As String, Optional NodeName As String) As Long

Structure of a link to an OPC server. The ProgID for Computing is:
OPCServer.WinAC

Disconnect ()

Structure of the link to an OPC server.

Note: The OPC server for Computing closes all communications connections after the Disconnect of the last OPC client.

ReleaseAll ()

Deletes all current groups and items as preparation for a Disconnect.

CreateBrowser () As OPCBrowser

Creates an object of the OPCBrowser class to investigate the address area of the server.

Note: Refer to the description of the object in Section 2.4.

GetErrorString (ErrorCode As Long , Optional LocaleID As Long) As String

Supplies the error message for a specific error code.

Note: The OPC server for Computing supports German and English error texts. Errors detected by the Windows operating system are explained in the language in which the operating system was installed.

2.4 The “OPCBrowser” Object

The OPCBrowser object is a collection object with which the address area of the OPC server can be investigated. An object of the OPCBrowser class must be created by the CreateBrowser method of the OPCServer object. It is possible to create several OPCBrowser objects for one server.

Properties of “OPCBrowser”

Table 2-2 Properties of the “OPCBrowser” Object

Property	Type	Brief Description
Organization	Long	The organizational structure of the addressarea: hierarchy or flat
Filter	String	The filter effective for the ShowBranches and ShowLeafs methods
DataType	Integer	The data type required for the ShowLeafs method. The default of this property is vbEmpty.
AccessRights	Long	The access rights required for the ShowLeafs method
CurrentPosition	String (read-only)	Current position in the tree of the address area. The value is “ ” when initialized in the root or when the organization structure is flat.
Count	Long	Properties necessary for the collection, provides the number of entries

Notes:

- The structure of the address area of the OPC server for Computing is hierarchical.
- The rules for creating a filter are as follows:
 - Asterisk (*) Any character string, including empty strings
 - Plus (+) Any string of characters, however at least one character
 - Question marks (?) Any single character
 - Open/close bracket ([]) Exactly one character from the specified set
- To use one of the filter characters, this must be preceded by a back slash (\).

Methods of “OPCBrowser”

Item (Key As Variant) As String

Provides the name of the entry specified by the index “Key”.

ShowBranches ()

Enters the names of the branches of the current browse position into the collection.

ShowLeafs (Optional Flat As Boolean)

Enters the names of the leaves of the current browse position into the collection. If the parameter “Flat” is true, the collection with all leaves of the current and deeper branches are filled starting from the current browse position. The default for “Flat” is false.

MoveUp ()

Moves the current position in the address area one level up.

MoveDown (Branch As String)

Moves the current position in the address area into the current branch (one level deeper).

MoveToRoot ()

Moves the current position in the address area to the root.

GetItemID (Leaf As String) As String

Creates a complete ItemID in the hierarchical address area. This function is necessary since browsing itself only provides the designations below the current node.

GetAccessPaths (ItemID As String) As Variant

This provides the possibility of querying the access path of an ItemID.

Note: Not required with the OPC server for Computing.

2.5 The “OPCGroups” Collection Object

The OPCGroups object is a collection object for creating and managing OPC groups. The default properties of OPC groups specify default values for creating all OPC groups.

Public groups are not supported by the OPC server for Computing.

Properties of “OPCGroups”

Table 2-3 Properties of the “OPCGroups” Object

Property	Type	Brief Description
Parent	OPC Server (read-only)	Provides a reference to the corresponding OPC server object
DefaultGroupActive	Boolean	Specifies the initial value for the “ActiveState” property of newly created OPC groups Default: True
DefaultGroupUpdate	Long	Specifies the initial value for the “update rate” property of newly created OPC groups Default: 1000 milliseconds
DefaultDeadband	Single	Specifies the initial value for the “Deadband” property of newly created OPC groups
DefaultLocale	Long	Specifies the initial value for the “locale ID” property of newly created OPC groups
DefaultTimeBias	Long	Specifies the initial value for the “time bias” property of newly created OPCGroups
Count	Long	Properties necessary for the collection, provides the number of entries

Notes:

- The DefaultTimeBias property is not evaluated by the OPC server for Computing.
- DefaultLocale is irrelevant for the OPC server for Computing.
- The DefaultGroupUpdate is specified by configuration parameter “Minimum Update Rate” as a multiple of the configuration value.
- The DefaultDeadband property has no significance for the OPC server for Computing.

Methods of “OPCGroups”

`Item (Key As Variant) As OPCGroup`

Provides a reference to the indexed object of the collection.

`Add (Name As string, ByRef ServerHandle As Long) As OPCGroup`

Creates a group in the server object.

Note: If the szName parameter is empty, a name is generated beginning with the underscore character (for example “_123456”). User-defined names should therefore not begin with the underscore character.

`GetOPCGroup (Key As Variant) As OPCGroup`

Provides the reference to an OPC group indicated by the name or the server handle.

`Remove (Key As Variant)`

Deletes a group on the server.

`RemoveAll (Key As Variant)`

Deletes all groups and items of the server.

Events of “OPCGroups”

`AllGroupsDataChange (GroupHandle as Long, MasterQuality as Long, MasterError as Long, NumItems as Long, ClientHandles() as Long, ItemValues() as Variant, Qualities() as Long, TimeStamps() as Date)`

This event simplifies the processing of events throughout all groups of the collection by reporting changes in the value and state of all items in all groups.

2.6 The “OPCGroup” Object

The “OPC Group” class manages the individual process variables, the OPC items. Using these group objects, a client can form semantically meaningful units of OPC items and execute operations with them.

Properties of “OPCGroup”

Table 2-4 Properties of the “OPCGroup” Object

Property	Type	Brief Description
Parent	OPC Server	Provides a reference to the corresponding OPC server object
Name	String	The name of the group
IsPublic	Boolean	Returns “True” when this group is a public group, otherwise False
IsActive	Boolean	Active state of the group An active group creates events for all active items of the group.
ClientHandle	Long	A handle assigned by the client and that can be used in the client program to localize data (for example, a line in a table)
ServerHandle	Long	A unique handle assigned for the group by the server The client must transfer this handle with one of the many methods that influence the group (for example, Remove).
LocaleID	Long	Specifies the language ID for strings supplied by the server (for example, error texts)
TimeBias	Long	Provides the time offset used to change the time stamp of the data to the local time
DeadBand	Single	Specifies a bandwidth within which value changes do not result in a message
UpdateRate	Long	The fastest rate at which a client is informed of changes in values or states of items
OPCItems	OPC Items	Collections object “OPCItems” for managing the items of a group

Notes:

- The TimeBias property is not evaluated by the OPC server for Computing.
- LocaleID is irrelevant for the OPC server for Computing.
- The UpdateRate is specified by the configuration parameter “Minimum Update Rate” as a multiple of the configuration value.
- The PercentDeadBand property has no significance for the OPC server for Computing.

Methods of “OPCGroup”

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

Synchronous reading of the values, status information or time stamp of one or more items in a group. The values can be read from the cache of the server or directly from the hardware. Reading from the cache is, however, only possible when the group is activated.

Note: The call is monitored by the timeout monitoring on the server. The corresponding configuration parameter is “Read/Write Timeout”.

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

Synchronous writing of values for one or more items of a group to the hardware.

Note: The call is monitored by the timeout monitoring on the server. The corresponding configuration parameter is “Read/Write Timeout”.

```
AsyncRead (Source As Integer, NumItems As Long, ServerHandles() As Variant, ByRef Errors() As Variant, ByRef TransactionID As Long) As Long
```

Sends an asynchronous read command. The result is returned with the “AsyncReadComplete” event.

Note: The call is monitored by the timeout monitoring on the server. The corresponding configuration parameter is “Read/Write Timeout”.

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

Sends an asynchronous write command. The result is returned with the “AsyncWriteComplete” event.

Note: The call is monitored by the timeout monitoring on the server. The corresponding configuration parameter is “Read/Write Timeout”.

```
AsyncRefresh (Source As Integer, ByRef TransactionID As Long) As Long
```

Requests a current value for every active OPC item. The results are returned by the “DataChange” event.

`AsyncCancel (TransactionID As Long)`

Cancels an outstanding asynchronous job.

Events of “OPCGroup”

The OPC automation interface supplies the changes to the values of active terms and the results of asynchronous operations with events.

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

The DataChange event occurs when it is detected that an active item has a change value or a change quality. Checking value changes is triggered by the UpdateRate timer. Only active items are created within a group of events.

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

The AsyncReadComplete event is triggered when a read job is completed.

`AsyncWriteComplete (TransactionID As Long, NumItems As Long,
ClientHandles() As Long, Errors() As Variant)`

The AsyncWriteComplete event is triggered when a write job is completed.

`AsyncCancelComplete (TransactionID As Long)`

The AsyncCancelComplete event is triggered when a cancel job is completed.

2.7 The “OPCItems” Collection Object

The OPCItems object is a collection object for creating and managing OPC items. The default properties of OPCItems specify default values for all OPC items to be created.

Properties of “OPCItems”

Table 2-5 Properties of the “OPCItems” Collection Object

Property	Type	Brief Description
Parent OPC	Group	Supplies a reference to the corresponding OPCGroup object
DefaultRequestedDataType	Integer	Specifies the initial value for the “RequestedDataType” property of newly added items The default value is vbEmpty for the canonical data type.
DefaultAccessPath	String	Specifies the initial value for the “AccessPath” property of newly added items The default is an empty string.
DefaultIsActive	Boolean	Specifies the initial value for the “Active State” property of newly added items The default value is true.
Count	Long	Properties necessary for the collection, provides the number of entries

Methods of “OPCItems”

Item (ItemSpecifier As Variant) As OPCItem

Provides a reference to the item of the collection described by the ItemSpecifier index. (The “GetOPCItem” method, on the other hand, provides a reference via the server handle.)

GetOPCItem (ServerHandle As Long) As OPCItem

Provides a reference to the server handle created by Add.

```
Add (NumItems As Long, ItemIDs() As String, ClientHandles() As Long,
ByRef ServerHandles() As Long, ByRef Errors() As Long, Optional
RequestedDataTypes() As Variant, Optional AccessPaths() As Variant)
```

Inserts one or more items in the OPCItems collection of a group.

Note: It is possible to add the same OPC item to the same group more than once. In this case each of these items nevertheless has its own server handle. The server handles of the items are only unique within a group and not for all items of all groups.

```
Remove (NumItems As Long, ServerHandles() As Long, ByRef Errors() As
Long)
```

Deletes one or more items from a group.

```
Validate (NumItems As Long, ItemIDs() As String, ByRef Errors() As
Long, Optional RequestedDataTypes () As Variant, Optional
AccessPaths() As Variant)
```

Checks the validity of an OPC item, for example whether it was added to a group without any error occurring, and supplies information such as the canonical data type.

Note: See Add.

```
SetActive (NumItems As Long, ServerHandles() As Long, ActiveState As
Boolean, ByRef Errors() As Long)
```

Sets the active state of one or more items in a group.

```
SetClientHandles (NumItems As Long, ServerHandles() As Long,
ClientHandles() As Long, ByRef Errors() As Longt)
```

Changes the client handle of one or more items in a group.

```
SetDataTypes (NumItems As Long, ServerHandles() As Long,
RequestedDataTypes() As Long, ByRef Errors() As Long )
```

Sets the required data type of one or more items in a group.

2.8 The “OPCItem” Object

An object of the class OPC item represents a link to a process variable, for example to the input module of a programmable controller. A process variable is data of the process I/Os that can be written and/or read, for example the temperature of a tank. Each process variable is associated with a value (variant data type), a quality, and a time stamp.

Properties of “OPCItem”

Table 2-6 Properties of the “OPCItem” Object

Property	Type	Brief Description
Parent OPC	Group	Supplies a reference to the parent OPCGroup object
ClientHandle	Long	A handle that can be freely defined by the user to allow simpler assignment of the process variable in internal data structures of the client
ServerHandle	Long	A handle assigned uniquely to the item by the server This handle is required in several operations to identify an item.
AccessPath	String	The access path of the item as specified in the Add function
AccessRights	Long	Provides the access rights of the variable
ItemID	String	The unique name of the item as specified in the Add function
IsActive	Boolean	Specifies whether message events will be created for this item
RequestedDataType	Integer	The data type in which the value of the item will be supplied
Value	Variant	The last valid value of the variable (default property of the OPCItem object)
Quality	Long	The quality of the value last read The quality indicates the validity of the value of the variable.
TimeStamp	Date	The time at which the last value was acquired
CanonicalDataType	Integer	The original data type of the item
EUType	Integer	Identifies the unit of the value If no units are available, the value is always “VT_EMPTY”.
EUInfo	Variant	Information about the unit of the value

Note: The OPC server for Computing does not support units (engineering units)

Methods of “OPCItem”

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

Reads the value, the quality, and/or the time stamp of this variable.

Write (Value As Variant)

Sets the value of this variable synchronously.

To

SIEMENS ENERGY & AUTOMATION INC
ATTN: TECHNICAL COMMUNICATIONS M/S 519
3000 BILL GARLAND ROAD
PO BOX 1255
JOHNSON CITY TN USA 37605-1255

From

Name: _____
Job Title: _____
Company Name: _____
Street: _____
City and State: _____
Country: _____
Telephone: _____

Please check any industry that applies to you:

- | | |
|---|---|
| <input type="checkbox"/> Automotive | <input type="checkbox"/> Pharmaceutical |
| <input type="checkbox"/> Chemical | <input type="checkbox"/> Plastic |
| <input type="checkbox"/> Electrical Machinery | <input type="checkbox"/> Pulp and Paper |
| <input type="checkbox"/> Food | <input type="checkbox"/> Textiles |
| <input type="checkbox"/> Instrument and Control | <input type="checkbox"/> Transportation |
| <input type="checkbox"/> Non-electrical Machinery | <input type="checkbox"/> Other _____ |
| <input type="checkbox"/> Petrochemical | |



Remarks Form

Your comments and recommendations will help us to improve the quality and usefulness of our publications. Please take the first available opportunity to fill out this questionnaire and return it to Siemens.

Please give each of the following questions your own personal mark within a range from 1 (very good) to 5 (very poor).

1. Do the contents meet your requirements?
2. Is the information you need easy to find?
3. Is the text easy to understand?
4. Does the level of technical detail meet your requirements?
5. Please rate the quality of the graphics and tables.

Additional comments:

This image shows a full page of primary-ruled paper. It features multiple horizontal rows, each consisting of two parallel dashed lines. These rows are evenly spaced across the entire page, providing a guide for handwriting practice. The background is white, and there are no margins or additional markings present.