Integration Objects'

OPC DA Client Development Toolkit

OPC DA Client Toolkit
Version 3.0 Rev.0

USER GUIDE

OPC DA Client Toolkit User Guide Version 3.0 Rev .0.
Published May 2016.

TABLE OF CONTENTS

LIST OF TABLES

PREFACE

ABOUT THIS USER GUIDE

This guide describes the functions exported by Integration Objects' OPC DA Client Toolkit and explains how to use this toolkit.

TARGET AUDIENCE

This document is intended for developers of OPC DA client applications. It assumes a working knowledge of programming using C++. It also assumes about a basic knowledge of OPC (OLE for Process Control) and OPC DA specification.

DOCUMENT CONVENTIONS

Convention	Description
Bold	Click/selection action required

CUSTOMER SUPPORT SERVICES

Phone	Email
Americas: +1 713 609 9208 **Europe-Africa-Middle East** +216 71 195 360	Support: customerservice@integrationobjects.com Sales: sales@integrationobjects.com Online: www.integrationobjects.com

INTRODUCTION

1. Overview

Integration Objects' OPC DA Client Toolkit is a Windows DLL (Dynamic Link Library) that implements all required and all optional OPC Data Access interfaces for both OPC Data Access specifications version
1.0a and 2.05. This Toolkit handles all OPC and COM details necessary to interface with OPC DA Servers. It is a tool for fast and easy implementation of client applications collecting real-time process date. Using this Toolkit, custom applications will be able to access data points from any OPC server, without having to be concerned with the details of OPC standards.

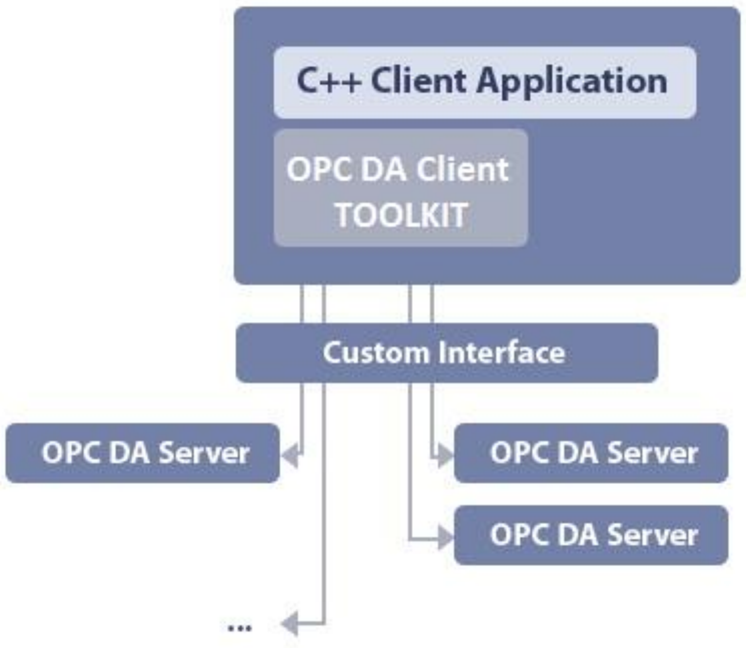


Figure 1: OPC DA Client Toolkit Architecture

2. Features

Features include:

- Auto-discovery of all OPC DA servers available on the network,
- Management of simultaneous connections to multiple local and remote OPC DA Servers,
- Browsing of OPC items available in the connected servers, if these latter implement the “browse" interface,
- Managing synchronous and asynchronous reads and writes,
- Support of “callback” feature to notify client applications when data values change.

Object	Interfaces	Implemented
OPCServer	IOPCCommon	Yes
	IOPCServer	Yes
	IOPCServerPublicGroups(optional)	Yes
	IOPCBrowseServerAddressSpace(optional)	Yes
	IOPCItemProperties	Yes
	IConnectionPointContainer	Yes
	IPersistFile(optional)	No
OPCGroup	IOPCGroupStateMgt	Yes
	IOPCPublicGroupStateMgt(optional)	Yes
	IOPCItemMgt	Yes
	IOPCSyncIO	Yes
	IOPCAsyncIO2	Yes
	IOPCAsyncIO	Yes
	IConnectionPointContainer	Yes
	IDataObject	Yes
OPCItemAttributes	IEnumOPCItemAttributes	Yes

Table 1: Interfaces

3. Systems Compatibility

This API runs under the following operating systems:

- Windows XP
- Windows 7
- Windows Server 2003
- Windows Server 2008

The OPC DA Client Toolkit is well integrated with Microsoft **Visual C**++, **Visual Studio 2010** and higher, **Borland C++ 5.0**, **Borland C++ 6.0** programming environment.

4. OPC Compatibility

OPC Data Access 1.0a
OPC Data Access 2.05

USING OPC DA CLIENT TOOLKIT

1. How to Link OPCDADll.lib with Client Application

With Visual C++ 6.0:

- Include OPCDADll.lib file with the project files for the custom application, this file contains the export definitions for the DLL's API.
- Include OPCDADll.h .

With Borland C++ 5.0 and Borland C++ 6.0:

- Include OPCDADll.lib file with the project files for the custom application, this file contains the export definitions for the DLL's API.
- Include OPCDA.h and OPCDADll.h.
- Copy the OPCDADll.dll file in the current folder.

With Visual Studio 2010 or higher:

For 32 bit applications:

- Include the **OPCDADLL.lib** file existing under "**Integration Objects' OPC DA Client Toolkit\bin\VS2010\x86**" folder with the project files for the custom application, this file contains the export definitions for the DLL's API.
- Include OPCDADll.h.

For 64 bit applications:

- Include the **OPCDADLL.lib** file existing under "**Integration Objects' OPC DA Client Toolkit\bin\VS2010\x64**" folder with the project files for the custom application, this file contains the export definitions for the DLL's API.
- Include OPCDADll.h.

2. Files Included in the Distribution

After running the setup, you will get the following folders and files on your system under the installation folder:

Folder	Description
Bin	**OPCDADLL.dll**: Release SDK DLL. **OPCDADLL.lib**
OPC Sample Demos	Contains 3 samples for OPC Clients.
OPC Sample Projects	Contains three sample projects corresponding to the OPC sample Demos.
Documents	**OPC DA Client Toolkit User's Guide.pdf**: this user guide

Table 2: Distributed Files

3. How to Build your Project with Visual Studio 2010 or Higher

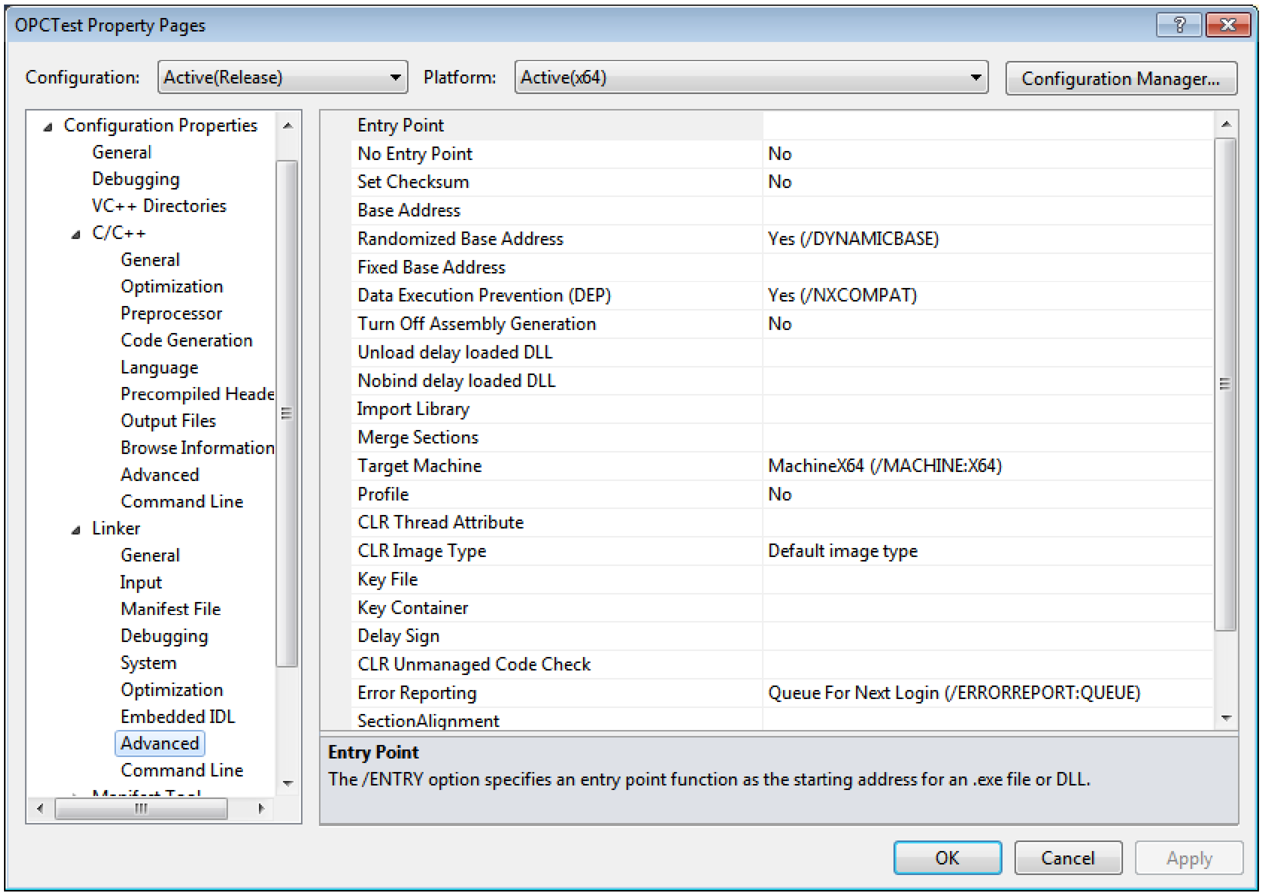
3.1. 64 BIT APPLICATIONS

1. Copy the **OPCDADLL.lib** file existing under "**Integration Objects' OPC DA Client Toolkit\bin\VS2010\x64**" folder to your project folder.

2. Copy the **OPCDADLL.dll** file existing under "**Integration Objects' OPC DA Client Toolkit\bin\VS2010\x64\Release**" folder to your release folder.

3. Copy the **OPCDADLL.dll** file existing under "**Integration Objects' OPC DA Client Toolkit\bin\VS2010\x64\Debug**" folder to your debug folder.

4. In the project properties, change the platform to (x64):



Entry Point	
No Entry Point	No
Set Checksum	No
Base Address	
Randomized Base Address	Yes (/DYNAMICBASE)
Fixed Base Address	
Data Execution Prevention (DEP)	Yes (/NXCOMPAT)
Turn Off Assembly Generation	No
Unload delay loaded DLL	
Nobind delay loaded DLL	
Import Library	
Merge Sections	
Target Machine	MachineX64 (/MACHINE:X64)
Profile	No
CLR Thread Attribute	
CLR Image Type	Default image type
Key File	
Key Container	
Delay Sign	
CLR Unmanaged Code Check	
Error Reporting	Queue For Next Login (/ERRORREPORT:QUEUE)
SectionAlignment	

Entry Point
The /ENTRY option specifies an entry point function as the starting address for an .exe file or DLL.

Figure 2: 64 bit application

5. Build your project.

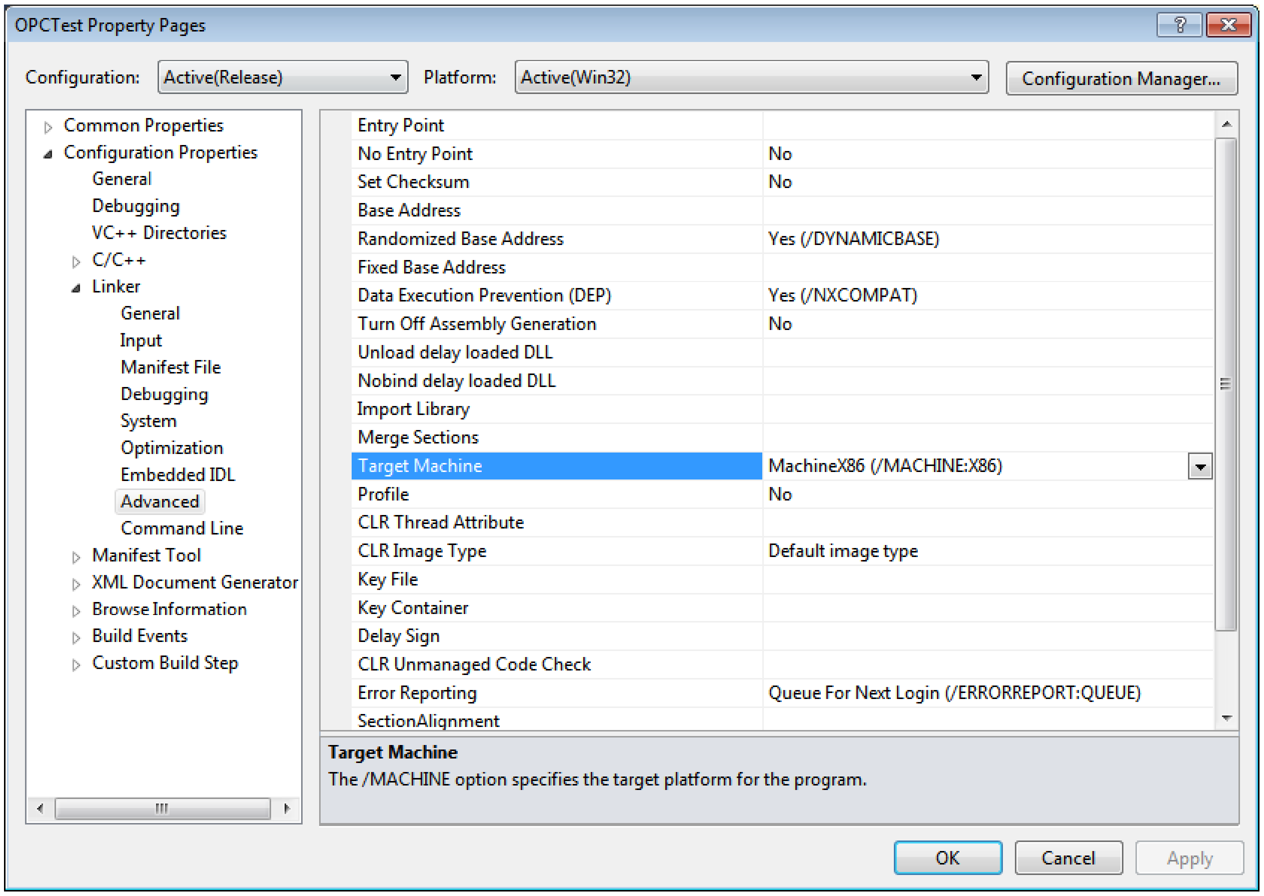
3.2. 32 BIT APPLICATIONS

1. Copy the **OPCDADLL.lib** file existing under "**Integration Objects' OPC DA Client Toolkit\bin\VS2010\x86**" folder to your project folder.

2. Copy the **OPCDADLL.dll** file existing under "**Integration Objects' OPC DA Client Toolkit\bin\VS2010\x86\Release**" folder to your release folder.

3. Copy the **OPCDADLL.dll** file existing under "**Integration Objects' OPC DA Client Toolkit\bin\VS2010\x86\Debug**" folder to your debug folder.

4. In the project properties, change the platform to win32:


OPCTest Property Pages

Configuration: Active(Release) Platform: Active(Win32) Configuration Manager...

Entry Point	
No Entry Point	No
Set Checksum	No
Base Address	
Randomized Base Address	Yes (/DYNAMICBASE)
Fixed Base Address	
Data Execution Prevention (DEP)	Yes (/NXCOMPAT)
Turn Off Assembly Generation	No
Unload delay loaded DLL	
Nobind delay loaded DLL	
Import Library	
Merge Sections	
Target Machine	MachineX86 (/MACHINE:X86)
Profile	No
CLR Thread Attribute	
CLR Image Type	Default image type
Key File	
Key Container	
Delay Sign	
CLR Unmanaged Code Check	
Error Reporting	Queue For Next Login (/ERRORREPORT:QUEUE)
SectionAlignment	

Target Machine
The /MACHINE option specifies the target platform for the program.

Figure 3: 32 bit application

5. Build your project.

4. Exported Functions

4.1. Initialization

Before calling any function exported by the dll, an initialization step needs to be performed by calling the InitDll function.

4.1.1. InitDll

```
HRESULT InitDll (void(*lpcallback)(int,OPCHANDLE, WORD, HRESULT, HRESULT,
DWORD, OPCHANDLE *,VARIANT *,WORD  *, FILETIME  *, HRESULT * ))
```

This function performs DCOM and security initialization and allows user to specify the name of the callback function that will be called when data changes notifications are received from the server.
The callback is defined as follows:

```
void lpcallback  ( [in]int          Shandle
                 , [in]OPCHANDLE    Ghandle
                 , [in]DWORD        transId
                 , [in]HRESULT      hrMasterquality
                 , [in]HRESULT      hrMastererror
                 , [in]DWORD        dwCount
                 , [in]OPCHANDLE   *phClientItems
                 , [in]VARIANT     *pvValues
                 ,  [in]WORD       *pwQualities
                 , [in]FILETIME    *pftTimeStamps
                 , [in]HRESULT     *pErrors)
```

Parameters

Parameters	Description
Shandle	The client handle of the server that sends this notification.
Ghandle	The client handle of the group.
transId	0 if the call is the result of an ordinary subscription. If the call is the result of a call to Refresh2 then this is the value passed to Refresh2.
hrMasterquality	S_OK if OPC_QUALITY_MASK for all 'qualities' are OPC_QUALITY_GOOD, S_FALSE otherwise.
hrMastererror	S_OK if all 'errors are S_OK, S_FALSE otherwise.
dwCount	The number of items in the client handle list.
phClientItems	The list of client handles for the items which have changed.
pvValues	A list of VARIANTS containing the values (in RequestedDataType) for the items which have changed.
pwQualities	A list of Quality values for the items.
pftTimeStamps	A list of TimeStamps for the items.
pErrors	A list of HRESULTS for the items. If the quality of a data item has changed to UNCERTAIN or BAD. This field allows the server to return additional server specific errors which

	provide more useful information to the user. See below.

Table 3: Parameters of the Callback function

HRESULT Return Values

Return Code	Description
0	The function was successful.

Table 4: Return Values of the Callback function

InitDll Example

This example illustrates how to call the **InitDll** method. First we declare a callback function, testcallback, and set it to the name of the callback we already implemented, lpcallback, and finally we call the **InitDll** method by passing testcallback as parameter.

```
void (*testcallback) ( int          Shandle
                    ,OPCHANDLE      Ghandle
                    ,DWORD          transId
                    ,HRESULT        hrMasterquality
                    ,HRESULT        hrMastererror
                    ,DWORD          dwCount
                    ,OPCHANDLE      *phClientItems
                    ,VARIANT        *pvValues
                    ,WORD           *pwQualities
                    ,FILETIME       *pftTimeStamps
                    ,HRESULT        *pErrors);
testcallback = lpcallback;
HRESULT hr= InitDll(testcallback);
```

4.2. Network Browsing API

4.2.1. BrowseNetwork

```
int BrowseNetwork (BrowseStruct** ppBrowseStruct)
```

This function browses the available machines and subnetworks on the network.

Parameters

Parameters	Description
ppBrowseStruct	Object in which browsing results will be returned. For more details about this object, see below BrowseStruct class description.

Table 5: Parameters of BrowseNetwork

HRESULT Return Values

Return Code	Description
0	The function was successful.
-10	The function fails because the call to WnetOpenEnum returns an error code.
-20	The function fails because the call to WnetCloseEnum returns an error code.

Table 6:Return Values of BrowseNetwork

BrowseStruct Class Description

```
 class BrowseStruct
 {
public:
            BrowseStruct();
           ~BrowseStruct();
            char*        GetName();
            void         SetName(char*);
            int           GetNbNode ();
            int           GetNbLeaf ();
            BrowseStruct GetNodeAt (int);
            BrowseStruct GetNode (char* name);
            char*          GetLeafAt (int);
            void           AddNode(BrowseStruct*);
            void           AddLeaf(char*);

 public:
             char*      Name;
             int          nbNode;
             int        nbLeaf;
             list<BrowseStruct*, allocator<BrowseStruct*> > *NodeList
          list<char*, allocator<char*> > LeafList;
 };
```

BrowseStruct class Attributes

Attributes	Description
Name	The name of the current node.
nbNode	The number of children nodes in the current node.
nbLeaf	The number of leaves that are directly under the current node.
NodeList	The list of children nodes in the current node.
LeafList	The list of leaves' names that are directly under the current node.

Table 7: Attributes of BrowseStruct class

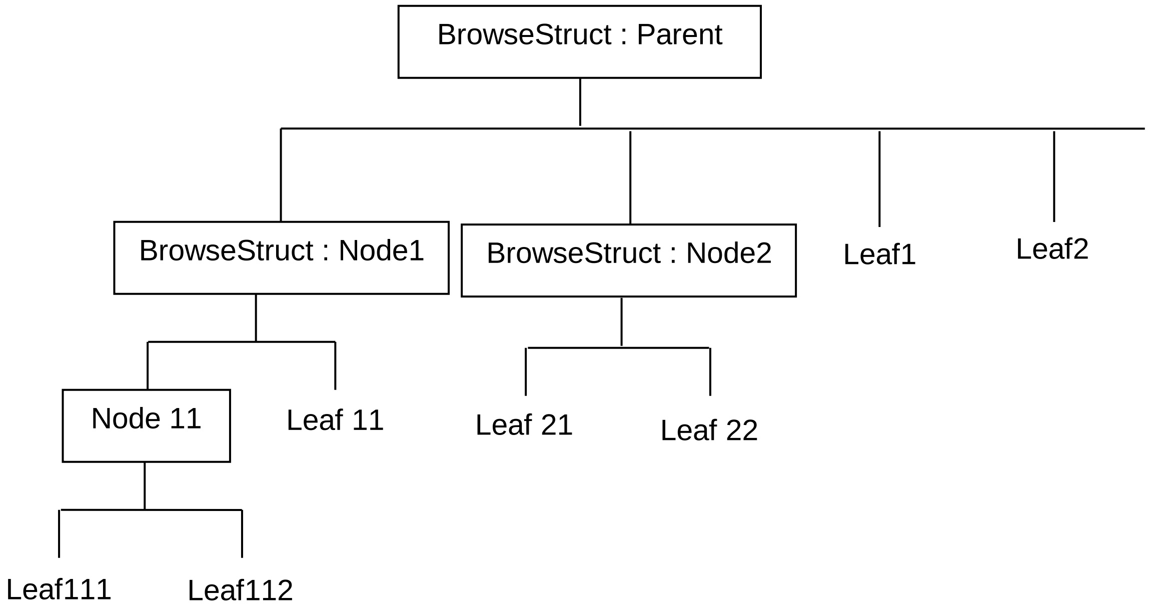
BrowseStruct class Methods

Methods	Description
BrowseStruct();	Constructor of the BrowseStruct class.
~BrowseStruct();	Destructor of the BrowseStruct class.
char* GetName();	Returns the "Name" attribute of the BrowseStruct object.
void SetName(char*);	Sets the attribute "Name" to the value passed in parameter.
int GetNbNode ();	Returns the "nbNode" attribute of the BrowseStruct Object.
int GetNbLeaf ();	Returns the "nbLeaf" attribute of the BrowseStruct Object.
BrowseStruct GetNodeAt (int index);	Returns the node located at position index in the "NodeList" attribute.
BrowseStruct GetNode (char* name);	Returns the node named "name" in the "NodeList" attribute.
char* GetLeafAt (int index);	Returns the name of the leaf located at position index in the "LeafList" attribute.
void AddNode(BrowseStruct*);	Adds a node to the "NodeList" attribute.

void AddLeaf(char*);	Adds a leaf to the “LeafList” attribute.

Table 8: Methods of BrowseStruct class

BrowseStruct Example



In this example
Name attribute is “Parent”
NbLeaf attribute is 2
NbNode attribute is 2
NodeList attribute contains Node1 and Node2 BrowseStruct objects
LeafList attribute contains Leaf1 and Leaf2

Dll’s exported functions to handle BrowseStruct class:

```
BrowseStruct* BrowseStruct_Constr()
```
This function creates a BrowseStruct object.

```
int  BrowseStruct_Destr(BrowseStruct* pBrowseStruct)
```
This function deletes the BrowseStruct object passed in parameter.

BrowseNetwork Example
This example creates a BrowseStruct object, pBrowseStruct, by calling the dll’s method “**BrowseStruct_constr()**”, then launches network browsing by calling the dll’s method “**BrowseNetwork(&pBrowseStruct)**”, and passing already created object **pBrowseStruct**

as a parameter. Finally to get browsing results, the user should implement his own method, as an example we implemented a method “ReadStruct” that displays Browsing results. Note that in the pBrowseStruct object, leaves will represent hosts in the current network, and nodes will represent subnets that compose the current network.

```
BrowseStruct *pBrowseStruct = BrowseStruct_Constr();
int result = BrowseNetwork(&pBrowseStruct);
ReadStruct(pBrowseStruct);
```

ReadStruct method:

```
void ReadStruct(BrowseStruct *pBrowseStruct)
{
 int LeafNb=0;
 int NodeNb =0;
 list<char*, allocator <char*> >::iterator  LeafIterator;
 list<BrowseStruct*, allocator<BrowseStruct*> >::iterator NodeIterator;

        printf("\n Browsing : %s", pBrowseStruct->Name);
        LeafNb = pBrowseStruct->nbLeaf;
        NodeNb= pBrowseStruct->nbNode;
        LeafIterator = pBrowseStruct->LeafList.begin();
        for (int i = 0; i<LeafNb; i++)
        {
           printf("\n Name of the leaf located at position %d : %s", i,
          (*LeafIterator));
           LeafIterator++;
        }
        NodeIterator = pBrowseStruct->NodeList->begin();
        for (i= 0; i<LeafNb; i++)
        {
           ReadStruct(*NodeIterator);
           NodeIterator++;
        }
}
```

4.2.2. GetLocalDAServerList

```
void GetLocalDAServerList([in] int Type  ,
                          [in] char* Catid  ,
                          [out] list<char*, allocator <char*>> *AServerList)
```

This function returns a list of OPC DA Servers installed on the local machine. The following table describes the parameters of this function:

Parameters:

In/Out	Parameter	Description
In	Type	0: use the catid to retrive the server list

		1 :Spec 1.0 using OPCENUM 2 :Spec 2.0 using OPCENUM 4 :Find OPC Spec 1.0 path HKEY_CLASSES_ROOT/appname/OPC
In	CatId	This is the OPC Data Access Interface.
Out	DAServerList	The OPC Server ProgID list

Table 9: Parameters of GetLocalDAServerList

Example:

```
char* CATID_OPCDAServer20="{63D5F432-CFE4-11d1-B2C8-0060083BA1FB}";
list<char*, allocator < char*> > *DAServerList=new list <char*, allocator <
char*> >;
GetLocalDAServerList(4, CATID_OPCDAServer20 , DAServerList);
```

4.2.3. GetRemoteDAServerList

This function returns a list of OPC DA Servers installed on the remote machine.

Syntax 1:

```
void GetRemoteDAServerList([in] char* NodeName
                          , [in] char* Catid
                          , [out] list<char*, allocator<char*> > *DAServerList)
```

The following table describes the parameters of this function:

Parameters:

In/Out	Parameter	Description
In	NodeName	The Host name or address
In	Catid	This is the OPC Data Access Interface
Out	DAServerList	The OPC Server ProgID list

Table 10: Parameters of GetRemoteDAServerList

Example:

```
char* CATID_OPCDAServer20="{63D5F432-CFE4-11d1-B2C8-0060083BA1FB}";
char MachineName[200];
list<char*, allocator < char*> > *DAServerList=new list <char*, allocator
   < char*> >;
GetRemoteDAServerList(MachineName  , CATID_OPCDAServer20 , DAServerList);
```

Syntax 2:

```
void GetRemoteDAServerList([in] int iType
                          , [in] char* NodeName
                          , [in] char* Catid
                          , [out] list<char*, allocator<char*> > *DAServerList)
```

The following table describes the parameters of this function:

Parameters:

In/Out	Parameter	Description
In	iType	This is the seek type. 0 : using OPCENUM 1 : Browse from registry
In	NodeName	The Host name or address
In	Catid	This is the OPC Data Access Interface
Out	DAServerList	The OPC Server ProgID list

Table 11: Parameters of GetRemoteDAServerList

Example:

```
char* CATID_OPCDAServer20="{63D5F432-CFE4-11d1-B2C8-0060083BA1FB}";
char MachineName[200];
list<char*, allocator < char*> > *DAServerList=new list <char*, allocator
   < char*> >;
GetRemoteDAServerList(0,MachineName  , CATID_OPCDAServer20 ,
   DAServerList);
```

4.3. Server Management

4.3.1. ConnectToServer

```
HRESULT ConnectToServer ( [in] char *ProgID,
                          [in] char *NodeName,
                          [out] int *Shandle)
```

This function establishes a connection with the server identified by ProgID and located on the machine " NodeName ". If the connection is established successfully, and if the server is DA 2.0, a new instance of OPCShutdown is created, to handle Shutdown Requests from this new OPC Server. The following table describes the parameters of this function:

Parameters

Parameters	Description
ProgID	Identifier of the server to which we want to establish a connection.
NodeName	Name of the machine on which this server is located.
Shandle	Client handle of the server to which the connection is established, client application will use this handle for many subsequent functions that the client requests the Dll to perform on the server.

Table 12: Parameters of ConnectToServer

HRESULT Return Values

Return Code	Description
0	Connected to server successfully
-7	CoCreateInstanceEx : failed, E_INVALIDARG error code : invalid NodeName argument.
-8	CoCreateInstanceEx : failed, can't retrieve the IUnknown interface.
-9	CoCreateInstanceEx : failed for unknown reason
-10	CLSIDFromProgID returns CO_E_CLASSSTRING error code: The registered CLSID for the ProgID is invalid.
-11	CLSIDFromProgID: returns REGDB_E_WRITEREGDB code
-12	CLSIDFromProgID: failed, unknown reason
-13	Invalid NodeName, empty string
-14	Invalid ProgID, empty string

Table 13: Return Values of ConnectToServer

ConnectToServer Example

This example establishes a local connection to "IntegrationObject.PI.OPC" OPC Server and returns its client handle, Shandle.

```
string ProgID="IntegrationObject.PI.OPC";
string NodeName= "localhost";
int Shandle=0;
HRESULT hr =ConnectToServer(ProgID.begin(),NodeName.begin(),&Shandle);
```

4.3.2. DisconnectServer

```
int DisconnectServer ([in] int SHandle)
```

This function disconnects from the server identified by its SHandle.

Parameters

Parameters	Description
Shandle	Client identifier of the server to delete. The identifier was returned when calling ConnectToServer method.

Table 14: Parameters of DisconnectServer

HRESULT Return Values

Return Code	Description
-1	Invalid Shandle, server not found.
>= 0	Number of server's interface not yet released.

Table 15: Return Values of DisconnectServer

4.3.3. GetOPCServerStatus

```
HRESULT GetOPCServerStatus ([in] int Shandle, [out] OPCSERVERSTATUS
&Serverstatus)
```

This function returns the current status of the identified server.

Parameters

Parameters	Description
Shandle	Client handle of the concerned server.
Serverstatus	Returned status of the server.

Table 16: Parameters of GetOPCServerStatus

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
-1	Failed: Invalid Shandle, server not found.
-3	Failed: Server's IOPCServer Interface not found.
-5	The operation failed.
-6	Not enough memory.
-7	An argument to the function was invalid.
-9	IOPCServer: GetStatus: failed for unknown error.

Table 17: Return Values of GetOPCServerStatus

GetOPCServerStatus Example

This example returns the status, returned in the parameter Serverstatus, of the server identified by Shandle.

```
OPCSERVERSTATUS Serverstatus;
HRESULT hr  = GetOPCServerStatus (Shandle, Serverstatus);
```

Where Shandle is the client handle of the server, the value returned when calling ConnectToServer method.

Note: Refer to "Data Access Custom Interface Standard Version 2.05" for details about OPCSERVERSTATUS structure.

4.3.4. BROWSEACCESSPATH

```
HRESULT BrowseAccessPath ([in] int Shandle, [in] char* ItemId, [out] list
<char*, allocator<char*> > *accesspathlst)
```

This function identifies the possible access paths for the specified "ItemId".

Parameters

Parameters	Description
Shandle	Client handle of the OPC server to which the item belongs.
ItemId	Identifier of the item to which we want to browse the

	available AccessPaths.
accesspathlst	List of access paths found for this item.

Table 18: Parameters of BrowseAccessPath

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
1	IOPCBrowseServerAddressSpace::BrowseAccessPaths: There is nothing to enumerate.
-1	Failed: Invalid Shandle, server object not found.
-3	Failed: Server's IOPCBrowseServerAddressSpace Interface not found.
-5	The function failed.
-6	Not enough memory.
-7	An argument to the function was invalid.
-9	IOPCBrowseServerAddressSpace::BrowseAccessPaths failed for unknown reason.
-10	The server does not require or support access paths.

Table 19: Return Values of BrowseAccessPath

BrowseAccessPpath Example

This example creates a list of char*, accesspathlst, by calling the dll's method "**createStringListInstance**". Then calls "**BrowseAccessPath**" method which returns the access paths found for the item "sinusoid" in IntegrationObjects.PI.OPC" OPC Server identified by Shandle.

```
list<char*, allocator<char*> > *accesspathlst= createStringListInstance();
res = BrowseAccessPath(Shandle, "sinusoid", accesspathlst);
```

4.3.5. GETERRORSTRING

```
HRESULT GetErrorString ([in]  int Shandle ,
                        [in] HRESULT dwError,
                        [out] LPWSTR *ErrorString)
```

This function returns the error string for a server specific error code.

Parameters

Parameters	Description
Shandle	Client handle of the server.
dwError	A 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.
ErrorString	The string corresponding to "dwError".

Table 20: Parameters of GetErrorString

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
-1	Failed: Invalid Shandle, server not found.
-3	Failed: Server's IOPCBrowseServerAddressSpace Interface not found
-5	The function failed.
-6	Not enough memory
-7	An argument to the function was invalid. (For example, the given error code was not valid.)
-9	IOPCServer:GetErrorString failed for unknown reason.

Table 21: Return Values of GetErrorString

4.3.6. BrowseServer

```
HRESULT BrowseServer (int Shandle, BrowseStruct **ppBrowseStruct)
```

This function browses the available data items in the server identified by Shandle.

Parameters

Parameters	Description
Shandle	Client handle of the server to browse.
ppBrowseStruct	Object in which Browsing results will be returned. (For more details about this object, see BrowseStruct class description in “NetworkBrowsing” section).

Table 22: Parameters of BrowseServer

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
-1	Failed: Invalid Shandle, server not found.

Table 23: Return Values of BrowseServer

BrowseServer Example

This example creates a BrowseStruct object, pBrowseStruct, by calling the dll’s method “**BrowseStruct_constr()**”, then launches the browsing of the server identified by Shandle by calling the dll’s method “**BrowseNetwork(&pBrowseStruct)**” and passing the already created object **pBrowseStruct** as parameter. Finally to get browsing results, the user should implement his own method, as an example we implemented a method “**ReadStruct**” that display Browsing results.

Note that in the pBrowseStruct object, leaves will represent items in the current area, and nodes will represent areas that compose the current area.

```
BrowseStruct * pBrowseStruct = BrowseStruct_Constr();
 HRESULT hr  = BrowseServer(Shandle, &pBrowseStruct);
         ReadStruct(pBrowseStruct);
```

4.3.7. BROWSEFLAT

```
HRESULT BrowseFlat([in] int Shandle, [in] char* strfilter , [in] long
dwAccessRightsFilter , [in] VARTYPE vtDataTypeFilter , [out] list<char*,
allocator<char*> > *ItemsList);
```

This function returns a list of item IDs. The following table describes the parameters of this function.

Parameters

In/Out	Parameter	Description
In	Shandle	The identifier of the connection
In	strfilter	A server specific filter string
In	vtDataTypeFilter	Filter the returned list based in the available data types; VT_EMPTY indicates no filtering.
In	dwAccessRightsFilter	Filter based on the Access Rights bit mask
Out	ItemsList	The returned list of items

Table 24: BrowseFlat Parameters

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
1	IOPCServer:BrowseFlat :The operation completed with one or more errors.
-1	The license expired.
-5	IOPCServer:BrowseFlat failed.
-6	IOPCServer:BrowseFlat: Not enough memory.
-7	IOPCServer:BrowseFlat : An argument to the function was invalid.
-9	IOPCServer:BrowseFlat: Failed for unknown reason.
-10	IOPCServer:BrowseFlat: The interface asked for is not supported by the server.
-11	IOPCServer:BrowseFlat: Invalid filter

Table 25: Return Values of BrowseFlat

Example:

```
int Shandle;
list<char*,allocator<char*>>*itemList =new list<char*,allocator<char*>>;
HRESULT  res= BrowseFlat(Shandle,"",0,VT_EMPTY,itemList);
```

4.4. GROUP MANAGEMENT

4.4.1. ADDGROUP

```
HRESULT AddGroup( [in] int Shandle,
                  [in] char* Gname,
                  [in] float *pDeadband,
                  [in] DWORD UpdateRate,
                  [in] int Readmode,
                  [in] int  Writemode,
                  [in] long TimeBias,
                  [out] DWORD &RevisedUpdateRate,
                  [out] int &Ghandle)
```

This function adds an OPC Group.

Parameters

Parameters	Description
Shandle	Client handle of the server to which we want to add a new group.
Gname	Name of the group.
pDeadband	A pointer to the desired percent change in an item value that will cause a subscription callback for that value to the client. A NULL pointer is equivalent to 0.0
UpdateRate	Specifies the fastest rate at which data changes may be sent from the server for items in this group. Passing 0 indicates the server should use the fastest practical rate. The rate is specified in milliseconds.
Readmode	The requested mode to read data, it should take one of the values listed in the table below.
Writemode	The requested mode to write data, it should take one of the values listed in the table below.
TimeBias	The time bias zone of the group.
RevisedUpdateRate	The returned update Rate that the server will actually use for the updateRate, this value may differ from the requested

	UpdateRate.
Ghandle	The returned client handle to the newly created group. This handle is attributed by the API and is different from the handle returned by the server. Client application will use this handle for many subsequent functions that the client requests the Dll to perform on the group.

Table 26: Parameters of AddGroup

Read Mode Values

Read Mode	Description
READSYNC	Allows the client to perform synchronous read operations to the server.
READASYNC1	Allows the client to perform asynchronous read operations to the server. The API will create an Advise connection between the client and the group. Since this connection is made, the client will be notified of changes in the items of the group. These changes are passed to client application via the callback specified while calling the “InitDll” function. This mode is useful when the server version is 1.0.
READASYNC2	Allows the client to perform asynchronous read operations to the server. The API will create a connection between the client and the group by using connection points. Once the connection is established, the client will be notified of changes in the items of the group. These changes will be passed to client application via the callback specified while calling the “InitDll” function. This mode is useful when server version is 2.0.
READASYNC	Allows the client to perform asynchronous read operations to the server. In this case, the API will decide, depending on the server version, whether to use connection point (READASYNC2) or advise sink connection (READASYNC1).

Table 27: Read Mode Values

Write Mode Values

Write Mode	Description
WRITESYNC	Allows the client to perform synchronous write operations to the server.

WRITEASYNC1	Allows the client to perform asynchronous write operations to a server version 1.0. (see READASYNC1 in the previous table for details).
WRITEASYNC2	Allows the client to perform asynchronous write operations to a server version 2.0. (see READASYNC2 in the previous table for details).
WRITEASYNC	Allows the client to perform asynchronous write operations to the server. (see READASYNC in the previous table for details).

Table 28: Write Mode Values

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
-1	Failed: Invalid Shandle, server not found.
-3	Failed: Server's IOPCServer Interface not found.
-5	IOPCServer:AddGroup failed.
-6	IOPCServer:AddGroup : Not enough memory.
-7	IOPCServer:AddGroup : An argument to the function was invalid.
-8	IOPCServer:AddGroup : Duplicate name not allowed.
-9	IOPCServer:AdGroup : Failed for unknown reason.
-10	IOPCServer:AddGroup : The interface(riid) asked for is not supported by the server.
-13	IConnectionPointContainer interface of the newly added group not available.
-23	IDataObject interface of the newly added group not available.
-25	FindConnectionPoint failed: Failed to obtain server's ConnectionPoint

-35	IDataObject :DAdvise : Data/Time Advise Failed.
-36	IDataObject :DAdvise : WriteComplete Advise Failed.
-44	IConnectionPoint: Advise: E_POINTER error.
-45	IConnectionPoint: Advise: CONNECT_E_ADVISELIMIT error
-46	IConnectionPoint:Advise :E_OUTOFMEMORY error
-47	IConnectionPoint: Advise: CONNECT_E_CANNOTCONNECT error
-48	IConnectionPoint: Advise: E_UNEXPECTED error
-49	IConnectionPoint: Advise: Failed for unknown reason.

Table 29: Return Values of AddGroup

AddGroup example

This example adds a new group, named “test” to the server identified by Shandle, and returns its client handle ghandle.

```
 int ghandle;
float deadband= 0.0;
DWORD RevUpdateRate;

HRESULT hr =AddGroup( Shandle
                       ,"test"
                      , &deadband
                      , 1000
                      , READASYNC2
                      , WRITEASYNC2
                      , 0
                      , RevUpdateRate
                      , ghandle);
```

4.4.2. RemoveGroup

```
HRESULT RemoveGroup ([in] int Shandle, [in] int Ghandle)
```

This function removes an OPC Group from a server.

Parameters

Parameters	Description
Shandle	Client handle of the server to which belongs the group to remove.
Ghandle	Client handle of the group to remove.

Table 30: Parameters of RemoveGroup

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
-1	Failed: Invalid Shandle, server object not found.
-2	Failed: invalid Ghandle, Group not found.
-3	Failed: Server's IOPCServer Interface not found.
-5	IOPCServer: RemoveGroup: The operation failed.
-6	IOPCServer: RemoveGroup: Not enough memory.
-7	IOPCServer: RemoveGroup: An argument to the function was invalid?
-8	IOPCServer: RemoveGroup: The group was not removed because references exist yet. The group will be marked as deleted, and will be removed automatically by the server when all references to it are released.
-9	IOPCServer: RemoveGroup: failed for unknown reason.

Table 31: Return Values of RemoveGroup

4.4.3. SetGroupState

```
HRESULT SetGroupState ([in] int Shandle,
                       [in] int Ghandle,
                       [in] DWORD *UpdateRate,
                       [out] DWORD &RevisedUpdateRate,
                       [in] BOOL *Active,
                       [in] LONG *TimeBias,
                       [in] FLOAT *PercentDeadband,
                       [in] DWORD *LCID,
                       [in] OPCHANDLE *hClientGroup)
```

This function changes various properties of the group. If the client doesn't want to change one of these parameters, he simply passes a "NULL" value.

Parameters

Parameters	Description
Shandle	Client handle of the server to which the group belongs.
Ghandle	Client handle of the group.
UpdateRate	The new update rate requested for the group by the client.
RevisedUpdateRate	The closest update rate the server is able to provide for this group.
Active	TRUE to activate the group and FALSE to deactivate it.
TimeBias	Enables the client to change the time zone Bias of the group.
PercentDeadband	The new update rate requested by the client.
LCID	The new Localisation ID that will be used by the group.
hClientGroup	Changes the client handle of the group.

Table 32: Parameters of SetGroupState

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
-1	Failed: Invalid Shandle, server object not found.
-2	Failed: invalid Ghandle, Group not found.
-3	Failed: group's IOPCGroupStateMgt Interface not found.
-5	IOPCGroupStateMgt : GetState: The operation failed.

-6	IOPCGroupStateMgt : GetState: Not enough memory
-7	IOPCGroupStateMgt : GetState: An argument to the function was invalid.

Table 33: Return Values of SetGroupState

SetGroupState example

This example changes the status of the group identified by ghandle, and belonging to the server identified by Shandle. In this example we just want to deactivate the group, that's why all other parameters are set to NULL, so we don't change their current value.

```
DWORD     RevisedUpdateRate = 0;
HRESULT hr= SetGroupState(Shandle, ghandle, NULL, RevisedUpdateRate,
false, NULL, NULL, NULL, NULL);
```

4.4.4. GetGroupState

```
        HRESULT GetGroupState ([in] int Shandle,
                               [in] int Ghandle,
                               [out] DWORD &UpdateRate,
                               [out] BOOL &Active,
                               [out] char* &gname,
                               [out] LONG &TimeBias,
                               [out] FLOAT &PercentDeadband,
                               [out] DWORD &LCID,
                               [out] OPCHANDLE &hClientGroup,
                               [out] OPCHANDLE &hServerGroup)
```

This function returns the current group state.

Parameters

Parameters	Description
Shandle	Client handle of the server to which the concerned group belongs.
Ghandle	Client handle of the concerned group.
UpdateRate	The current group update rate.
Active	The current group state (active or not).
gname	The current group name.

TimeBias	The current group TimeBias.
PercentDeadband	The current group percentDeadband.
LCID	The current group LCID.
hClientGroup	Client handle of the group.
hServerGroup	Server handle of the group. This handle is attributed by the server when this group was created inside the server.

Table 34: Parameters of GetGroupState

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
-1	Failed: Invalid Shandle, server object not found.
-2	Failed: invalid Ghandle, group not found.
-3	Failed: group's IOPCGroupStateMgt Interface not found.
-5	IOPCGroupStateMgt: GetState: The operation failed.
-6	IOPCGroupStateMgt: GetState: Not enough memory.
-7	IOPCGroupStateMgt: GetState: An argument to the function was invalid.

Table 35: Return Values of GetGroupState

GetGroupState Example

This example gets the current status of the group iden,tified by ghandle.

```
OPCHANDLE hServerGroup=0;
DWORD   RevisedUpdateRate=0;
DWORD   UpdateRate=0;
BOOL     Active= true;
char*      gname;
LONG     TimeBias=0;
FLOAT    PercentDeadband=0.0;
DWORD   LCID=0;
OPCHANDLE cliengrouphandle=0;
```

```
HRESULT hr = GetGroupState(Shandle, ghandle, UpdateRate,
Active,gname, TimeBias, PercentDeadband , LCID, cliengrouphandle
, hServerGroup);
```

4.4.5. SetGroupName

```
HRESULT SetGroupName ([in] int Shandle,[in] int Ghandle,[in] char* name)
```

This function changes the group name.

Parameters

Parameters	Description
Shandle	Client handle of the server to which the group belongs.
Ghandle	Client handle of the group.
name	The new name given to the group.

Table 36: Parameters of SetGroupName

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
-1	Failed: Invalid Shandle, server not found.
-2	Failed: invalid Ghandle, group not found.
-3	Failed: Group's IOPCGroupStateMgt Interface not found.
-4	Invalid argument: empty name.
-5	IOPCGroupStateMgt: SetName: Failed.
-6	IOPCGroupStateMgt: SetName: Not enough memory.
-7	IOPCGroupStateMgt: SetName: An argument to the function was invalid.
-8	IOPCGroupStateMgt: SetName: Duplicate name not allowed.

-9	IOPCGroupStateMgt: SetName: Failed for unknown reason.

Table 37: Return Values of SetGroupName

4.5. Items Management

4.5.1. AddItem

```
HRESULT AddItem ([in]  int Shandle,
                      [in] int Ghandle,
                     [in] char*ItemID,
                     [in] char*  AccessPath,
                     [in] bool bActive,
                     [in] DWORD dwBlobSize,
                     [in] BYTE *pBlob,
                     [in] VARTYPE vtRequestedDataType,
                     [out] int *pIhandle,
                     [out] HRESULT *pError)
```

This function adds an item to a specified OPC Group and returns a client handle “pIhandle” that identifies uniquely the item in the group.

Parameters

Parameters	Description
Shandle	Client handle of the server to which the item will be added.
Ghandle	Client handle of the group to which the item will be added.
ItemID	A null-terminated string that uniquely identifies the OPC data item.
AccessPath	The access path of the item if supported, else a Null string should be passed.
bActive	State of the item to be added.
dwBlobSize	The size of the pBlob for this item.
pBlob	pBlob is a pointer to the Blob.

vtRequestedDataType	The data type requested by the client. Passing VT_EMPTY means the client will accept the server's canonical datatype.
pIhandle	The returned client handle to the newly created item. Client application will use this handle for many subsequent functions that the client requests the API to perform on the item.
pError	Indicates if the item was successfully added, otherwise provides the reason of the failure.

Table 38: Parameters of AddItem

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
-1	Failed: Invalid Shandle, server object not found.
-2	Failed: Invalid Ghandle, group object not found.
-3	Failed: Group's IOPCItemMgt Interface not found.
-4	IOPCItemMgt: AddItems: The function completed with an error. See the pError to determine what happened.
-5	IOPCItemMgt: AddItems: Failed.
-6	IOPCItemMgt: AddItems: Not enough memory.
-7	IOPCItemMgt: AddItems: An argument to the function was invalid.
-9	IOPCItemMgt: AddItems: Failed for unknown reason.

Table 39: Return Values of AddItem

AddItem example

This example adds "sinusoid" item, and returns its client handle in the pIhandle parameter. If the item isn't added successfully the reason of the failure will be returned in the error parameter.

```
int Ihandle =0;
```

```
HRESULT error = S_OK;
HRESULT hr = AddItem (Shandle,ghandle,"sinusoid", "", true, 0,NULL
                          , VT_R8, &Ihandle, &error);
```

4.5.2. AddItems

```
HRESULT AddItems ([in] int SHandle,
                  [in] int GHandle,
                  [in/out] list<StructItem*> *ItemList,
                  [out] HRESULT ** ppErrors)
```

This function adds many items to a group. This function uses the list ItemList to store the items attributes.

Parameters

Parameters	Description
Shandle	Client handle of the server to which the items will be added.
Ghandle	Client handle of the group to which the items will be added.
ItemList	List of StructItem to be added. This is an in/out parameter because client handle of the added items we be returned in the “Ihandle” attribute of each StructItem object. (For more details about “StructItem” class, see below StructItem class description).
ppErrors	Array of HRESULT that indicates which of the items was successfully added. When an item failed it provides a reason.

Table 40: Parameters of AddItems

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
1	IOPCItemMgt: AddItems: The function completed with errors. See the ppErrors to determine what happened.

-1	Failed: Invalid Shandle, server object not found.
-2	Failed: invalid Ghandle, group object not found.
-3	Failed: group's IOPCItemMgt Interface not found.
-5	IOPCItemMgt: AddItems: Failed.
-6	IOPCItemMgt: AddItems: Not enough memory.
-7	IOPCItemMgt: AddItems: An argument to the function was invalid.
-9	IOPCItemMgt: AddItems: Failed for unknown reason.

Table 41: Return Values of AddItems

StructItem class description

The class OPC structure used to handle OPC item between client and the API.

```
class StructItem
{
public:
          StructItem(char* ItemID
                    , char* AccessPath
                    , bool bActive
                    , DWORD  dwBlobSize
                    , BYTE * pBlob
                    , VARTYPE vtRequestedDataType);
          ~StructItem();
 public :
          char*        AccessPath;
          char*        ItemID;
          bool         bActive;
          DWORD dwBlobSize;
           BYTE        * pBlob;
           VARTYPE     vtRequestedDataType;
           int         Ihandle;
           int         IShandle;
 };
```

Attributes:

Attributes	Description
AccessPath	The access path of the item. By convention a pointer to a NUL string specifies that the server should select the access path.
ItemID	A null-terminated string that uniquely identifies the OPC data item.
bActive	Indicate the active state of item. TRUE if item is activate. FALSE if item is deactivate.
dwBlobSize	The size of the pBlob for this item.
pBlob	pBlob is a pointer to the Blob.
vtRequestedDataType	The data type requested by the client. IF the server cannot provide the item in this format. Passing VT_EMPTY means the client will accept the server's canonical datatype.
Ihandle	This attribute will receive the client handle of the newly added item. This handle is attributed by the IOPCAPI dll and client application will use this handle for many subsequent functions that the client requests the API to perform on the item.
IShandle	This attribute will receive the handle attributed by the OPC Server to the newly added Item.

Table 42: Attributes of StructItem class

Methods

Methods	Description
StructItem(char* ItemID, char* AccPath,bool bActive, DWORD dwBlobSize, BYTE * pBlob,VARTYPE vtReqDataType);	Constructor of the StructItem class.
~BrowseStruct();	Destructor of the StructItem class.

Table 43: Methods of StructItem class

Dll's exported functions to handle StructItem class

4.5.3. StructItem_Constr

```
StructItem* StructItem_Constr ([in] char* ItemID,
                                [in] char* AccessPath,
                                [in] bool bActive,
                                [in] DWORD dwBlobSize,
                                [in] BYTE * pBlob,
                                [in] VARTYPE vtRequestedDataType)
```

This function creates a StructItem object.

4.5.4. StructItem_Destr

```
int  StructItem_Destr ([in] StructItem* Objet)
```

This function deletes the StructItem object passed in parameter.

AddItems Example

This example adds two items: “sinusoid” and “sinusoidu” to the group identified by ghandle.
Before calling “AddItems”, the user should prepare the “ItemStructlst” parameter. He should, first, create a list of StructItem objects by calling “createItemStructListInstance” method. Then, for each item to add, create a StructItem instance, item1 and item2, by calling “StructItem_Constr” method. Then insert the new StructItem object into the already created list, ItemStructlst. Finally call the “AddItems” method and pass the list of StructItem as parameter. Client handle of the added items will be returned in the attribute “Ihandle” of each StructItem object in the ItemStructlst

```
list <StructItem*> *ItemStructlst = createItemStructListInstance();
StructItem *item1= StructItem_Constr("sinusoid","", true, 0,NULL,
VT_R8);
ItemStructlst->insert(ItemStructlst->end(),item1);
StructItem *item2= StructItem_Constr("sinusoidu", "", true, 0, NULL,
VT_R8);
ItemStructlst->insert(ItemStructlst->end(), item2);
HRESULT* perrors;
HRESULT hr = AddItems(Shandle, ghandle, ItemStructlst, &perrors);
```

4.5.5. RemoveItem

```
HRESULT RemoveItem ([in] int Shandle,
                    [in] int Ghandle,
                    [in] int Ihandle,
                    [out] HRESULT **pperror)
```

This function removes an OPC item identified by Ihandle.

Parameters

Parameters	Description
Shandle	Client handle of the server from which the item will be removed.
Ghandle	Client handle of the group from which the item will be removed.
Ihandle	Client handle of the item to remove.
pperror	Indicates if the item was successfully removed, else provides a reason.

Table 44: Parameters of RemoveItem

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
-1	Failed: Invalid Shandle, server object not found.
-2	Failed: Invalid Ghandle, group object not found.
-3	Failed: Group’s IOPCItemMgt Interface not found.
-4	Failed: Invalid Ihandle, Item not found.
-5	IOPCItemMgt: RemoveItems: The function was unsuccessful.
-7	IOPCItemMgt: RemoveItems: An argument to the function was invalid.
-8	IOPCItemMgt: RemoveItems: Cannot remove items from a public group.
-9	IOPCItemMgt: RemoveItems: Failed for unknown reason.
-10	IOPCItemMgt: RemoveItems: The function completed with one error. See the pperror to determine what happened.

Table 45: Return Values of RemoveItem

4.5.6. RemoveItems

```
HRESULT RemoveItems ([in] int SHandle,
                     [in] int GHandle,
                     [in] list<int, allocator<int> > *handleLst,
                     [out] HRESULT **pperrors)
```

This function removes one or more OPC items. The handles of the items to remove are stored in the “handleLst” parameter.

Parameters

Parameters	Description
Shandle	Client handle of the server from which items will be removed.
Ghandle	Client handle of the group from which items will be removed.
handleLst	List of Client handles of the items to remove.
pperrors	The array of HRESULT that indicates which of the items was successfully removed. When an item failed, it provides a reason.

Table 46: Parameters of RemoveItems

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
1	IOPCItemMgt: RemoveItems: The function completed with one error. See the pperrors to determine what happened.
-1	Failed: Invalid Shandle, server object not found.
-2	Failed: Invalid Ghandle, group object not found.
-3	Failed: Group’s IOPCItemMgt Interface not found.
-5	IOPCItemMgt: RemoveItems: The function was unsuccessful.

-7	IOPCItemMgt: RemoveItems: An argument to the function was invalid.
-8	IOPCItemMgt: RemoveItems: Cannot remove items from a public group.
-9	IOPCItemMgt: RemoveItems: Failed for unknown reason.

Table 47: Return Values of RemoveItems

4.5.7. REMOVEALLITEMS

```
HRESULT RemoveAllItems ([in] int Shandle,
                         [in] int Ghandle,
                         [out] HRESULT **pperrors)
```

This function removes all the OPC items associated to a specific group.
Parameters

Parameters	Description
Shandle	Client handle of the server from which the items will be removed.
Ghandle	Client handle of the group from which the items will be removed.
pperrors	The array of HRESULT that indicates which of the items was successfully removed. When an item failed, it provides a reason.

Table 48: Parameters of RemoveAllItems

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
1	IOPCItemMgt: RemoveItems: The function completed with one error. See the pperrors to determine what happened.
-1	Failed: Invalid Shandle, server object not found.
-2	Failed: Invalid Ghandle, group object not found.

-3	Failed: Group's IOPCItemMgt Interface not found.
-5	IOPCItemMgt: RemoveItems: The function was unsuccessful.
-7	IOPCItemMgt: RemoveItems: An argument to the function was invalid.
-8	IOPCItemMgt: RemoveItems: Cannot remove items from a public group.
-9	IOPCItemMgt: RemoveItems: Failed for unknown reason.

Table 49: Return Values of RemoveAllItems

4.5.8. GetItemProperties

```
HRESULT GetItemProperties ([in] int Shandle,
                           [in] char*ItemID,
                           [out] DWORD *Count,
                           [out] VARIANT**ppvData,
                           [out] HRESULT **ppErrors,
                           [out] LPWSTR **ppDescriptions,
                           [out] VARTYPE **ppvtDataTypes)
```

This function returns a list of the current properties values for a given "ItemID".

Parameters

Parameters	Description
Shandle	Client handle of the server to which the item belong.
ItemID	The ItemID for which the client wants to read the list of properties.
Count	The number of properties returned. Indicates the number of values in the array ppvData.
ppvData	An array of VARIANTS, returned by the API, which contains the current values of the requested properties.
ppErrors	Error array indicating whether each property was returned.
ppDescriptions	An array that indicates a brief vendor supplied text description of every property.

ppvtDataTypes	An Array of VARTYPE that indicates the datatype for every property.
ppPropertyIDs	DWORD IDs for the returned properties.

Table 50: Parameters of GetItemProperties

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
1	IOPCItemProperties: GetItemProperties: The operation completed with one or more errors. Refer to ppErrors for failure analysis.
-1	Failed: Invalid Shandle, server object not found.
-3	Failed: Server's IOPCItemProperties Interface not found.
-4	Failed: ItemID empty.
-6	IOPCItemProperties: QueryAvailableProperties: Not enough memory.
-7	IOPCItemProperties: QueryAvailableProperties: An argument to the function was invalid.
-8	IOPCItemProperties: QueryAvailableProperties: The ItemID is syntactically invalid.
-9	IOPCItemProperties: QueryAvailableProperties: Failed for unknown reason.
-10	IOPCItemProperties: QueryAvailableProperties: The ItemID is not in the server address space.
-15	IOPCItemProperties: GetItemProperties: Failed.
-16	IOPCItemProperties: GetItemProperties: Not enough memory.
-17	IOPCItemProperties: GetItemProperties: An argument to the function was invalid.

-19	IOPCItemProperties: GetItemProperties: Failed for unknown reason.

Table 51: Return Values of GetItemProperties

4.5.9. CHANGEITEMSTATE

```
HRESULT ChangeItemState ( [in] int Shandle,
                          [in] int Ghandle,
                          [in]list<int, allocator<int> >*handleLst,
                          [in] bool bActive,
                          [out] HRESULT **pperrors)
```

This function sets one or more items in a group to active or inactive. The handles of the items are stored in the "handleLst" parameter.

Parameters

Parameters	Description
Shandle	Client handle of the server to which these items belong.
Ghandle	Client handle of the group to which these items belong.
handleLst	List of Client handles of items for which we want to change the state.
bActive	TRUE if items should be activated. FALSE if items should be deactivated.
pperrors	Array of HRESULT that indicates which items were successfully affected.

Table 52: Parameters of ChangeItemState

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
1	IOPCItemMgt: SetActiveState: The operation completed with one or more errors. Refer to pperrors for failure analysis.

-1	Failed: Invalid Shandle, server object not found.
-2	Failed: Invalid Ghandle, group object not found.
-3	Failed: Group's IOPCItemMgt interface not found.
-5	IOPCItemMgt: SetActiveState: Failed.
-7	IOPCItemMgt: SetActiveState: An argument to the function was invalid.
-9	IOPCItemMgt: SetActiveState: Failed for unknown reason.

Table 53: Return Values of ChangeItemState

4.5.10. ValidateItems

```
HRESULT ValidateItems ( [in] int Shandle,
                        [in] int Ghandle,
                        [in] list<StructItem*>*ItemList,
                        [in] BOOL bBlobUpdate,
                        [out] OPCITEMRESULT **ppValidationResults,
                        [out] HRESULT **pperrors)
```

This function determines if an item is valid (could it be added without error). Also returns information about the item such as canonical data type.

Parameters

Parameters	Description
Shandle	Client handle of the server to which these items belong.
Ghandle	Client handle of the group to which these items belong.
ItemList	List of StructItem (for details about StructItem class refer to section “Utils”).
bBlobUpdate	If non-zero (and the server supports Blobs) the server should return updated Blobs in ppValidationResults. If zero (False) the server will not return Blobs in OPCITEMRESULTs.
ppValidationResults	Array of OPCITEMRESULTs. This tells the client additional information about the item including the canonical datatype. Refer to “Data Access Custom Interface Standard Version 2.05” for details about

	OPCITEMRESULT structure.
pperrors	Array of HRESULT that indicates which of the items was successfully validated. For any item that failed it provides a reason.

Table 54: Parameters of ValidateItems

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
1	IOPCItemMgt: ValidateItems: The operation completed with one or more errors. Refer to pperrors for failure analysis.
-1	Failed: Invalid Shandle, server object not found.
-2	Failed: Invalid Ghandle, group object not found.
-3	Failed: Group's IOPCItemMgt interface not found.
-5	IOPCItemMgt: ValidateItems: Failed.
-6	IOPCItemMgt: ValidateItems: Not enough memory.
-7	IOPCItemMgt: ValidateItems: An argument to the function was invalid.
-9	IOPCItemMgt: ValidateItems: Failed for unknown reason.

Table 55: Return Values of ValidateItems

4.6. UTILS

4.6.1. CREATEINTLISTINSTANCE

```
list<int, allocator<int> > *createIntListInstance()
```

This function creates a list of int.

4.6.2. DELETEINTLISTINSTANCE

```
void deleteIntListInstance([in] list<int, allocator<int> > *intlist)
```

This function deletes the list of int passed in parameter.

4.6.3. CreateItemStructListInstance

```
list<StructItem*> * createItemStructListInstance()
```

This function creates a list of ItemStruct objects.

4.6.4. DeleteItemStructListInstance

```
void deleteItemStructListInstance([in] list<StructItem*> *Itemlist)
```

This function deletes the list of ItemStruct objects passed in parameter.

4.7. Performing Read and Write operations

4.7.1. StartRead

```
HRESULT StartRead ([in] int Shandle, [in] int Ghandle, [in] int Readmode)
```

This function launches read operation.

Parameters

Parameters	Description
Shandle	Client handle of the server to which the group belongs.
Ghandle	Client handle of the group on which we want to perform read operations.
Readmode	Should be one of the following: READSYNC, READASYNC1, READASYNC2, READASYNC.

Table 56: Parameters of StartRead

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
-1	Failed: Invalid Shandle, server not found.
-2	Failed: Invalid Ghandle, group not found.
-13	IConnectionPointContainer interface of the concerned group is not available.
-23	IDataObject interface of the concerned group is not available.

-25	FindConnectionPoint, failed: failed to obtain server's ConnectionPoint
-35	IDataObject :DAdvise : Data/Time Advise Failed
-36	IDataObject :DAdvise : WriteComplete Advise Failed
-46	IConnectionPoint:Advise :E_OUTOFMEMORY error
-48	IConnectionPoint: Advise: E_UNEXPECTED error
-44	IConnectionPoint: Advise: E_POINTER error.
-45	IConnectionPoint: Advise: CONNECT_E_ADVISELIMIT error
-47	IConnectionPoint: Advise: CONNECT_E_CANNOTCONNECT error
-49	IConnectionPoint: Advise: failed for unknown error.

Table 57: Return Values of StartRead

4.7.2. StartWrite

```
HRESULT StartWrite ([in]  int Shandle, [in] int Ghandle, [in] int mode)
```

This function launches write operation.

Parameters

Parameters	Description
Shandle	Client handle of the server to which the group belongs.
Ghandle	Client handle of the group on which we want to perform write operations
mode	Should be one of the following: WRITEDSYNC, WRITEASYNC1, WRITEASYNC2, WRITEASYNC.

Table 58: Parameters of StartWrite

HRESULT Return Values

Return Code	Description

0	The operation succeeded.
-1	Failed: Invalid Shandle, server not found.
-2	Failed: Invalid Ghandle, group not found.
-13	IConnectionPointContainer interface of the concerned group is not available.
-23	IDataObject interface of the concerned group is not available.
-25	FindConnectionPoint, failed: failed to obtain server's ConnectionPoint
-35	IDataObject :DAdvise : Data/Time Advise Failed
-36	IDataObject :DAdvise : WriteComplete Advise Failed
-46	IConnectionPoint:Advise :E_OUTOFMEMORY error
-48	IConnectionPoint: Advise: E_UNEXPECTED error
-44	IConnectionPoint: Advise: E_POINTER error.
-45	IConnectionPoint: Advise: CONNECT_E_ADVISELIMIT error
-47	IConnectionPoint: Advise: CONNECT_E_CANNOTCONNECT error
-49	IConnectionPoint: Advise: failed for unknown error.

Table 59: Return Values of StartWrite

4.7.3. StopRead

```
HRESULT StopRead ([in] int Shandle, [in] int Ghandle)
```

This function stops read operation. Shandle is the client handle of the server to which the group belongs. Ghandle is the client handle of the group.

Parameters

Parameters	Description
Shandle	Client handle of the server to which the group belongs.

Ghandle	Client handle of the group on which we want to stop read operations.

Table 60: Parameters of StopRead

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
-1	Failed: Invalid Shandle, server not found.
-2	Failed: Invalid Ghandle, group not found.

Table 61: Return Values of StopRead

4.7.4.StopWrite

```
HRESULT StopWrite ([in] int Shandle, [in] int Ghandle)
```

This function stops write operation.

Parameters

Parameters	Description
Shandle	Client handle of the server to which the concerned group belongs.
Ghandle	Client handle of the group on which we want to stop write operations.

Table 62: Parameters of StopWrite

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
-1	Failed: Invalid Shandle, server not found.
-2	Failed: Invalid Ghandle, group not found.

Table 63: Return Values of StopWrite

4.7.5.SyncRead

```
HRESULT SyncRead ([in] int Shandle,
                       [in] int Ghandle,
                       [in] list<int, allocator<int> > * handleLst,
                       [in] char source,
                       [out] OPCITEMSTATE **ppItemValues,
                       [out] HRESULT **pperrors)
```

This function performs synchronous read operation to one or more items in a group.

Parameters

Parameters	Description
Shandle	Client handle of the server to which the concerned group belongs.
Ghandle	Client handle of the group.
handleLst	List of client handle of items that we want to read their current values.
Source	Could be “CACHE” or “DEVICE”.
ppItemValues	Array of structures in which the item values will be returned.
pperrors	Array of errors generated by these reads.

Table 64: Parameters of SyncRead

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
1	IOPCSyncIO: Read: The operation completed with one or more errors. Refer to pperrors for failure analysis.
-1	Failed: Invalid Shandle, server not found.
-2	Failed: Invalid Ghandle, group not found.
-3	Failed: group’s IOPCSyncIO interface not available.
-5	IOPCSyncIO: Read: Failed.
-6	IOPCSyncIO: Read: Not enough memory.
-7	IOPCSyncIO: Read: An argument to the function was invalid.
-9	IOPCSyncIO: Read: Failed for unknown error.

Table 65: Return Values of SyncRead

SyncRead Example

This example creates a list of int, handlelist, by calling the dll’s method “createIntListInstance”. Then, inserts client handle of the items to be read, Ihandle1 and Ihandle2. These handles are returned by the dll when adding these items. Finally, calls the “SyncRead” method, Items’ values will be returned in the pItemValues parameter.

```
list<int, allocator<int> > *handlelist = createIntListInstance();
handlelist->insert(handlelist->end(), Ihandle1);
handlelist->insert(handlelist->end(), Ihandle2);
OPCITEMSTATE *pItemValues;
HRESULT *perrors;
hr = SyncRead( Shandle, ghandle, handleLst , DEVICE, &pItemValues ,
&perrors );
```

Note: Refer to “Data Access Custom Interface Standard Version 2.05” for details about OPCITEMSTATE structure.

4.7.6. SyncWrite

```
HRESULT SyncWrite ([in] int Shandle,
                   [in] int Ghandle,
                   [in] list<int, allocator<int> >* handleLst,
                   [in] VARIANT *pItemValues,
                   [out] HRESULT **pperrors)
```

This function performs synchronous write operation to one or more items in a group.

Parameters

Parameters	Description
Shandle	Client handle of the server to which the concerned group belongs.
Ghandle	Client handle of the group.
handleLst	List of client handle of items on which we want to perform the write operation.
pItemValues	List of values to be written.
pperrors	Array of errors generated by these reads.

Table 66: Parameters of SyncWrite

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
1	IOPCSyncIO: Write: The operation completed with one or more errors. Refer to pperrors for failure analysis.
-1	Failed: Invalid Shandle, server not found.
-2	Failed: Invalid Ghandle, group not found.
-3	Failed: Group's IOPCSyncIO interface not available.
-5	IOPCSyncIO: Write: Failed.
-6	IOPCSyncIO: Write: Not enough memory.
-7	IOPCSyncIO: Write: An argument to the function was invalid.
-9	IOPCSyncIO: Write: Failed for unknown error.

Table 67: Return Values of SyncWrite

SyncWrite Example

This example creates a list of int, handlelist, by calling the dll's method "createIntListInstance". Then, inserts client handle of the items to be read, Ihandle1 and Ihandle2. These handles are returned by the dll when adding these items. Prepare the value to be written, pItemValues. Finally, calls the "SyncWrite" method.

```
list<int, allocator<int> > *handlelist = createIntListInstance();
handlelist->insert(handlelist->end(), Ihandle1);
handlelist->insert(handlelist->end(), Ihandle2);
VARIANT pItemValues[2];
VariantInit(&pItemValues[0]);
VariantInit(&pItemValues[1]);
pItemValues [0].vt = VT_R8;
pItemValues [0].dblVal = 1.11;
pItemValues [1].vt = VT_R8;
pItemValues [1].dblVal = 2.22;
HRESULT *perrors;
HRESULT hr = SyncWrite( Shandle, ghandle, handleLst ,  pItemValues
, &perrors );
```

4.7.7. ASync1Read

```
HRESULT ASync1Read ( [in] int Shandle,
                     [in] int Ghandle,
                     [in] list<int, allocator<int> >* handleLst,
                     [in] char source,
                     [out] DWORD *TransactionID,
                     [out] HRESULT **pperrors)
```

This function performs Asynchronous read on one or more items in a group. The current value for these items, timestamp and OPC Quality flags will be supplied to the client application via the callback for OPC Server Version 1.

Parameters

Parameters	Description
Shandle	Client handle of the server to which the concerned group belongs.
Ghandle	Client handle of the group.
handleLst	List of client handle of items that we want to know their current values.
source	Could be “CACHE” or “DEVICE”.
TransactionID	Server generated transaction Id.
pperrors	Array of errors generated by these reads.

Table 68: Parameters of ASync1Read

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
1	IOPCAsyncIO: Read: The operation completed with one or more errors. Refer to pperrors for failure analysis.
-1	Failed: Invalid Shandle, server not found.
-2	Failed: Invalid Ghandle, group not found.
-3	Failed: Group’s IOPCAsyncIO interface not available.
-4	Read launched with another mode (READSYNC or READASYNC2 mode), to launch ASync1Read, you should call StartRead (Shandle, Ghandle,

	READASYNC1) before or add the group with a READASYNC1 mode to register the needed callback (IDataObject:Dadvise).
-5	IOPCAsyncIO: Read: Failed.
-6	IOPCAsyncIO: Read: Not enough memory.
-7	IOPCAsyncIO: Read: An argument to the function was invalid.
-8	IOPCAsyncIO: Read: The client has not registered a callback via IDataObject:DAdvise.
-9	IOPCAsyncIO: Read: failed for unknown error.

Table 69: Return Values of ASync1Read

ASync1Read Example

This example performs an asynchronous read of the items identified by their client handle carried by the handlelist parameter, and returns an identifier, TransactionID, that identifies this request when items values are received by the callback function, lpcallback (refer to "initialisation" section for details about this callback).

```
list<int, allocator<int> > *handlelist = createIntListInstance();
handlelist->insert(handlelist->end(), Ihandle1);
handlelist->insert(handlelist->end(), Ihandle2);
DWORD TransactionID;
HRESULT hr = ASync2Read(Shandle, ghandle, handlelist, CACHE,  &
TransactionID, &perrors);
```

4.7.8. ASYNC2READ

```
HRESULT ASync2Read ([in] int Shandle,
                    [in] int Ghandle,
                    [in] list<int, allocator<int> > * handleLst ,
                    [out]  DWORD *CancelID,
                    [out] HRESULT **pperrors)
```

This function performs Asynchronous read on one or more items in a group. The current value for these items, timestamp and OPC Quality flags will be supplied to the client application via the callback. For OPC Server Version 2.

Parameters

Parameters	Description
Shandle	Client handle of the server to which the concerned group belongs.

Ghandle	Client handle of the group.
handleLst	List of client handle of items that we want to know their current values.
CancelIID	Server generated ID to be used in case this operation needs to be cancelled.
pperrors	Array of errors generated by these reads.

Table 70: Parameters of ASync2Read

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
1	IOPCAsyncIO2: Read: The operation completed with one or more errors. Refer to pperrors for failure analysis.
-1	Failed: Invalid Shandle, server not found.
-2	Failed: Invalid Ghandle, group not found.
-3	Failed: Group’s IOPCAsyncIO2 interface not available.
-4	Read launched with another mode (READSYNC or READASYNC1 mode), to launch Async2Read, you should call StartRead (Shandle, Ghandle, READASYNC2) before or add the group with a READASYNC2 mode to register the needed callback (IConnectionPoint: Advise).
-5	IOPCAsyncIO2: Read: Failed.
-6	IOPCAsyncIO2: Read: Not enough memory.
-7	IOPCAsyncIO2: Read: An argument to the function was invalid.
-8	IOPCAsyncIO2: Read: The client has not registered a callback via IConnectionPoint:Advise.
-9	IOPCAsyncIO2: Read: failed for unknown error.

Table 71: Return Values of ASync2Read

4.7.9. ASYNC1WRITE

```
HRESULT ASync1Write ([in] int Shandle,
                     [in] int Ghandle,
                     [in] list<int, allocator<int> > * handleLst,
                     [in] VARIANT *pItemValues,
                     [out]  DWORD *TransactionID,
                     [out] HRESULT **pperrors)
```

This function performs Asynchronous write on one or more items in a group. For OPC Server Version 1.

Parameters

Parameters	Description
Shandle	Client handle of the server to which the concerned group belongs.
Ghandle	Client handle of the group.
handleLst	List of client handle of items on which we want to perform the write operation.
pItemValues	List of values to be written.
TransactionID	Server generated transaction Id.
pperrors	Array of errors generated by these writes.

Table 72: Parameters of ASync1Write

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
1	IOPCAsyncIO1: Write: The operation completed with one or more errors. Refer to pperrors for failure analysis.
-1	Failed: Invalid Shandle, server not found.
-2	Failed: Invalid Ghandle, group not found.
-3	Failed: group's IOPCAsyncIO interface not available.
-4	Write launched with another mode (WRITESYNC or WRITEASYNC2 mode), to launch Async1Write, you should call StartWrite (Shandle, Ghandle, WRITEASYNC1) before or add the group with a

	WRITEASYNC1 mode to register the needed callback (IDataObject: DAdvise).
-5	IOPCAsyncIO1: Write: Failed.
-6	IOPCAsyncIO1: Write: Not enough memory.
-7	IOPCAsyncIO2: Write: An argument to the function was invalid.
-8	IOPCAsyncIO1: Write: The client has not registered a callback via IDataObject:DAdvise.
-9	IOPCAsyncIO1: Write: failed for unknown error.

Table 73: Return Values of ASync1Write

4.7.10. ASYNC2WRITE

```
HRESULT ASync2Write ([in]  int Shandle ,
                     [in] int Ghandle,
                    [in] list<int, allocator<int> > * handleLst,
                    [in] VARIANT *pItemValues,
                     [out] DWORD *pdwCancelID,
                     [out]  HRESULT **pperrors)
```

This function performs Asynchronous write on one or more items in a group. For OPC Server Version 2.

Parameters

Parameters	Description
Shandle	Client handle of the server to which the concerned group belongs.
Ghandle	Client handle of the group.
handleLst	List of client handle of items on which we want to perform the write operation.
pItemValues	List of values to be written.
CancelID	Server generated ID to be used in case this operation needs to be cancelled.
pperrors	Array of errors generated by these writes.

Table 74: Parameters of ASync2Write

HRESULT Return Values

Return Code	Description
0	The operation succeeded.
1	IOPCAsyncIO2: Write: The operation completed with one or more errors. Refer to pperrors for failure analysis.
-1	Failed: Invalid Shandle, server not found.
-2	Failed: Invalid Ghandle, group not found.
-3	Failed: group's IOPCAsyncIO2 interface not available.
-4	Write launched with another mode (WRITESYNC or WRITEASYNC1 mode), to launch Async2Write, you should call StartWrite (Shandle, Ghandle, WRITEASYNC2) before or add the group with a WRITEASYNC2 mode to register the needed callback (IConnectionPoint:Advise).
-5	IOPCAsyncIO2: Write: Failed.
-6	IOPCAsyncIO2: Write: Not enough memory.
-7	IOPCAsyncIO2: Write: An argument to the function was invalid.
-8	IOPCAsyncIO2: Write: The client has not registered a callback via IConnectionPoint:Advise.
-9	IOPCAsyncIO2: Write: failed for unknown error.

Table 75: Return Values of ASync2Write

TOOLKIT TRACING CAPABILITIES

The DLL has tracing capabilities. Developer can record the DLL errors and debugging information (COM and OPC messages) in a log file named “OPCDAdll_LogEvent.LOG”. If difficulties occur with the DLL the log file can be extremely valuable for troubleshooting. Under normal operation, the DLL logs very little information.
This log file is generated at the start-up where client executable is located. The toolkit incorporates a configuration file “Config.ini” which includes several logging parameters. All these parameters have default settings and can be changed at start-up by editing the configuration file.

To change this file:

1. Open **Config.ini** in a text editor.
2. Edit any of the parameters listed in the following tables:

Log Setting	Description	Default Value
LogFileMaxSize	The maximum log file size, in bytes. Once this size is reached during run-time, the log file is overwritten	1048576*2 ~ 2 Mo (MegaByte)
LogLevel	The log level. The higher the log level, the more information is recorded. We recommend using level 0 for better performance.	0
ArchiveLastLog	TRUE: The old file is copied to an intermediate file with incremental extension, before being overwritten. FALSE: Any pre-existing log file is erased and overwritten at start-up.	FALSE

Table 76: Configuration File Parameters

3. Save the file for the log settings and performance parameter to take effect.

Sample Configuration File

```
[LogSetting]
LogFileMaxSize = 2097152
LogLevel = 0
ArchiveLastLog = False
```

For additional information on this guide, questions or problems to report, please contact:

Offices

- Americas:	+1 713 609 9208
- Europe-Africa-Middle East:	+216 71 195 360

Email

- Support Services: customerservice@integrationobjects.com
- Sales: sales@integrationobjects.com

To find out how you can benefit from other Integration Objects products and custom-designed solutions, please visit our website:www.integrationobjects.com