Integration Objects'

.NET Toolkit for OPC DA/HDA/A&E Client Applications Development

OPC .NET Client Toolkit
Version 3.0 Rev.0

USER GUIDE

OPC Compatibility
OPC Data Access 1.10
OPC Data Access 2.05
OPC Data Access 3.00
OPC Historical Data Access 1.10
OPC Historical Data Access 1.20
OPC Alarms and Events 1.10

customerservice@integrationobjects.com

OPC .NET Client Toolkit User Guide Version 3.0 Rev.0
Published December 2023

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

PREFACE

ABOUT THIS USER GUIDE

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

TARGET AUDIENCE

This reference manual is intended for .NET developers of OPC Classic DA/HDA/A&E client applications. It assumes that you have a working knowledge of OPC Classic and programming with the .NET languages.

RELATED DOCUMENTATION

OPC Foundation (www.opcfoundation.org)

- OPC Data Access Specification
- OPC Historical Data Access Specification
- OPC Alarms and Events Specification

DOCUMENT CONVENTIONS

Convention	Description
`Monospaced type`	Indicates a file reference
	Information to be noted

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 .NET Client Toolkit is a DLL that implements all required and optional OPC Data Access interfaces for the 2.05 standard, OPC Historical Data Access interfaces for the 1.20 standard, and OPC Alarms and Events interfaces for the 1.10 standard. It also implements the required interfaces for the OPC Data Access 3.00. Refer to the list of supported interfaces in the next section.
This toolkit handles all COM and OPC details necessary to communicate with OPC servers. It is a tool for fast and easy creation of OPC client applications using the .NET framework

Using this toolkit, .NET custom applications will be able to access real-time, historical and alarms and events data from any OPC server without having to be concerned with the details of the OPC standard.

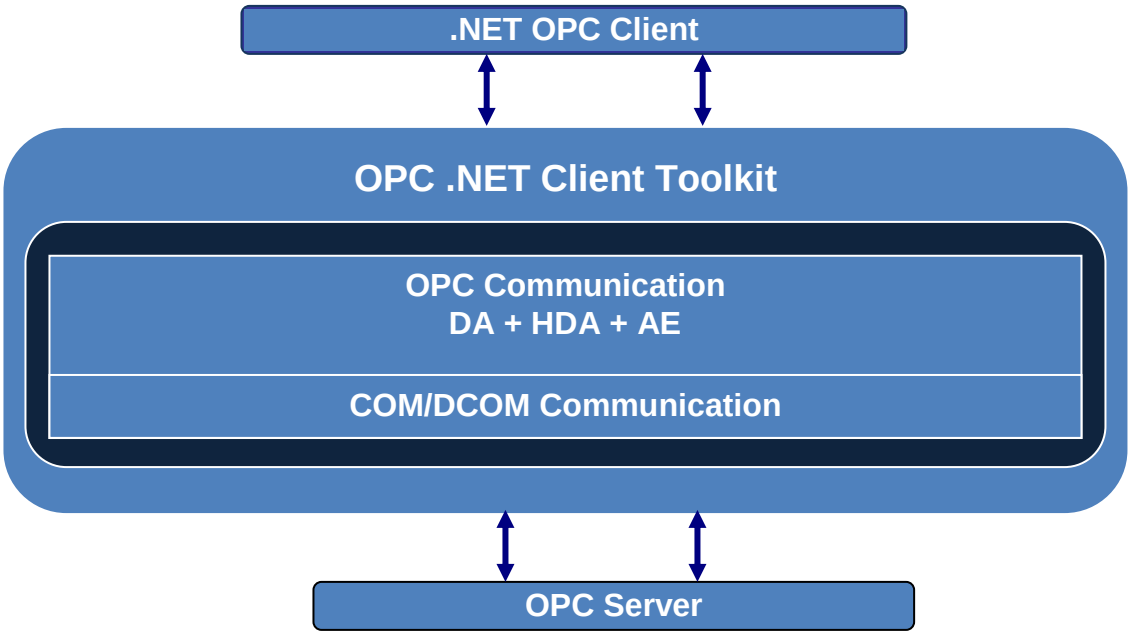

Figure 1: Overview of the OPC .NET Client Toolkit

2. Features

The main features of OPC .NET Client Toolkit are:

- Support of the latest released specifications for OPC Data Access, OPC Historical Data Access, and OPC Alarms and Events. This toolkit is fully compliant with OPC DA version 1.10/2.05, OPC HDA version 1.10/1.20 and OPC AE version 1.10.
- Support of the required interfaces of OPC Data Access version 3.00.
- Auto-discovery of OPC servers available on the network.
- Managing multiple local and remote connections to multiple OPC Servers.
- Offering the following OPC Data Access functionalities:
  - Browsing OPC Server address space
  - Managing OPC DA groups (Add, Remove)
  - Performing synchronous/asynchronous reads and writes of attributes values, their timestamps, and their qualities from/to OPC DA servers
- Offering the following OPC Historical Data Access functionalities:
  - Browsing OPC Server address space
  - Managing HDA Items (Validate item, Get handle and Release handle)
  - Collecting historical data from OPC HDA servers
  - Retrieving aggregates over a time range, such as average, minimum, maximum, etc.
  - Performing synchronous/asynchronous reading of attributes values and their timestamps from the HDA servers for a specified time domain and a given item
  - Adding, updating or removing entries from OPC HDA servers
- Offering the following OPC Alarms and Events functionalities:
  - Area and source browsing
  - Managing event subscriptions (Add, Remove)
  - Receiving alarms and events in real time
  - Acknowledging alarms
  - Adding customized filter constraints to be applied on incoming alarms
- Support of 32 bit and 64 bit applications

3. Supported Interfaces

The supported interfaces are summarized in the table below:

OPC DA Interfaces	V. 1.10/2.05	V3.0	Supported
OPCServer	Required		
IOPCCommon	Required	Required	Yes
IOPCDA_Server	Required	Required	Yes
IOPCDA_ServerPublicGroups	Optional	Required	Yes

IOPCDA_BrowseServerAddressSpace	Optional	Required	Yes
IOPCDA_ItemProperties	Required	N/A	Yes
IOPCBrowse	N/A	Required	Yes
IOPCItemIO	N/A	Required	Yes
OPCGroup			
IOPCDA_ItemMgt	Required	Required	Yes
IOPCDA_GroupStateMgt	Optional	Required	Yes
IOPCDA_PublicGroupStateMgt	Required	N/A	Yes
IOPCDA_SyncIO	Required	Required	Yes
IOPCDA_SyncIO2	N/A	Required	Yes
IOPCDA_AsyncIO2	Required	Required	Yes
IOPCDA_AsyncIO3	N/A	Required	Yes
IConnectionPointContainer	Required if any async interface is supported	Required	Yes
IOPCGroupStateMgt2	N/A	Required	Yes
IOPCItemDeadbandMgt	N/A	Required	Yes
IOPCItemSamplingMgt	N/A	Optional	No
OPC HDA Interfaces	**V. 1.10/1.20**		**Supported**
OPCServer			
IOPCCommon	Required		Yes
IOPCHDA_Server	Required		Yes
IOPCHDA_SyncRead	Required		Yes
IOPCHDA_SyncUpdate	Optional		Yes
IOPCHDA_SyncAnnotations	Optional		Yes
IOPCHDA_AsyncRead	Optional		Yes

IOPCHDA_AsyncUpdate	Optional	Yes
IOPCHDA_AsyncAnnotations	Optional	Yes
IOPCHDA_Playback	Optional	Yes
IConnectionPointContainer	Required if any Async interface is supported	Yes
OPCBrowser		
IOPCHDA_Browser	Optional	Yes
OPC A&E Interfaces	**V. 1.10**	**Supported**
OPCEventServer		
IOPCCommon	Required	Yes
IOPCAE_EventServer	Required	Yes
IOPCAE_EventServer2	Optional	Yes
IConnectionPointContainer	Required if any Async interface is supported	Yes
OPCEventAreaBrowser		
IOPCEventAreaBrowser	Optional	Yes
OPCEventSubscription		
IOPCEventSubscriptionMgt	Required	Yes
IOPCEventSubscriptionMgt2	Optional	Yes

Table 1: Supported Interfaces

4. Operating Systems Compatibility

This toolkit runs under the following operating systems:

- Windows XP
- Windows 7
- Windows 8
- Windows 10
- Windows Server 2003
- Windows Server 2008

- Windows Server 2012
- Windows Server 2016
- Windows Server 2019
- Windows 11
- Windows Server 2022

5. OPC Compatibility

- OPC Data Access 1.10, 2.05 and 3.00.
- OPC Historical Data Access 1.20
- OPC Alarms and Events 1.10

GETTING STARTED

1. Pre-Installation Considerations

In order to properly run any OPC Client developed by using the OPC .NET Client Toolkit, you need to install the following software components on the target system:

- The OPC core components 3.00 which consist of all shared OPC modules including the DCOM proxy/stub libraries, the OPC Server Enumerator, .NET wrappers, etc.
  You can alternatively apply the OPC Core Components 3.00 Redistributable delivered with the current package or download it from the OPC Foundation site (www.opcfoundation.org).
- .NET Framework 4.6.1 or higher.
- .NET Core 3.1 or higher.

2. Installation

To install the OPC .NET Client Toolkit, right click on the downloaded installation executable and select “Run as administrator” from the displayed menu. The prompted wizard will take you through the different installation steps.

If you are evaluating the OPC .NET Client Toolkit, make sure to select demo version when reaching the setup type dialog. Otherwise, select full version. The evaluation license allows you to use the toolkit for 30 days and limits the runtime to 2 hours.

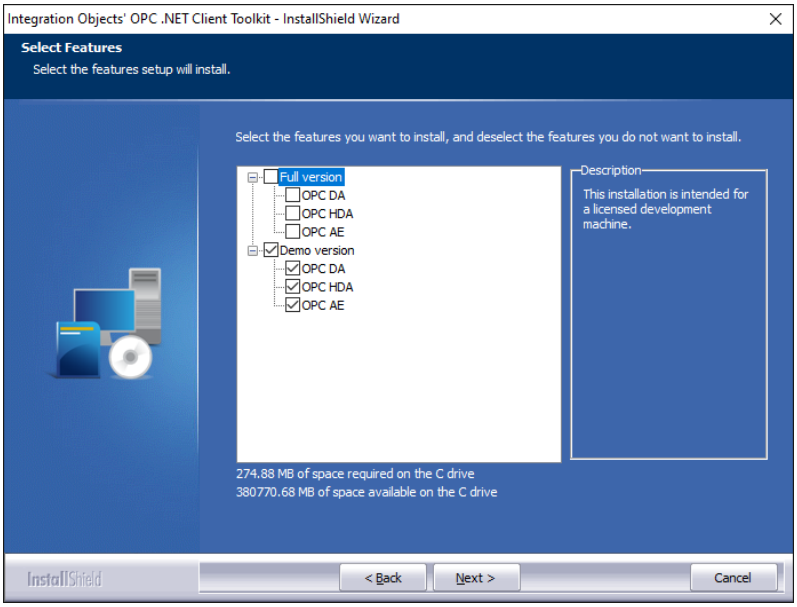

Figure 2: Select Features Dialog

Check the Install OPC Core Components if you want to install all shared OPC modules including the DCOM proxy/stub libraries, the OPC Server Enumerator, .NET wrappers, etc.

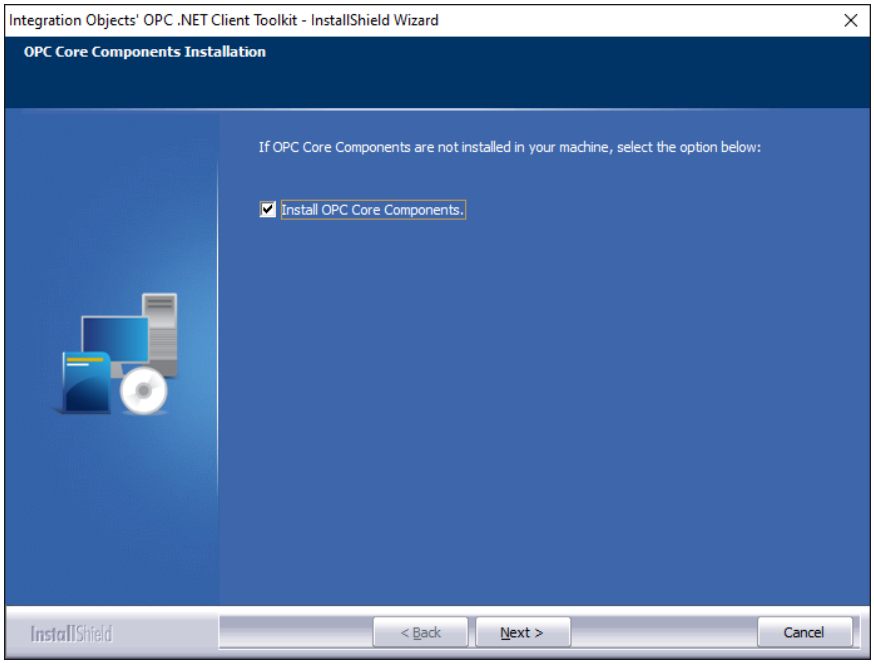

Figure 3: OPC Core Components Installation Dialog

Once the installation is complete, you will have the following shortcuts in your start menu:

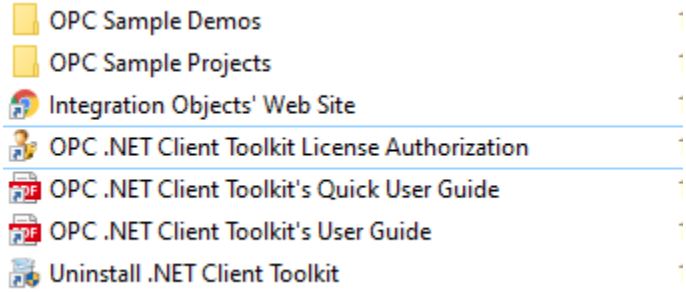

Figure 4: OPC .Net Client Toolkit Start Menu

3. Compiling and Linking Applications

This section focuses on the steps required to compile and correctly link applications to develop a custom OPC client application using Integration Objects' OPC Client .NET Toolkit with Microsoft Visual Studio 2017.

For users who have to build the application in a **32-bit** machine, the target platform has to be **x86** as shown in the screenshot below.

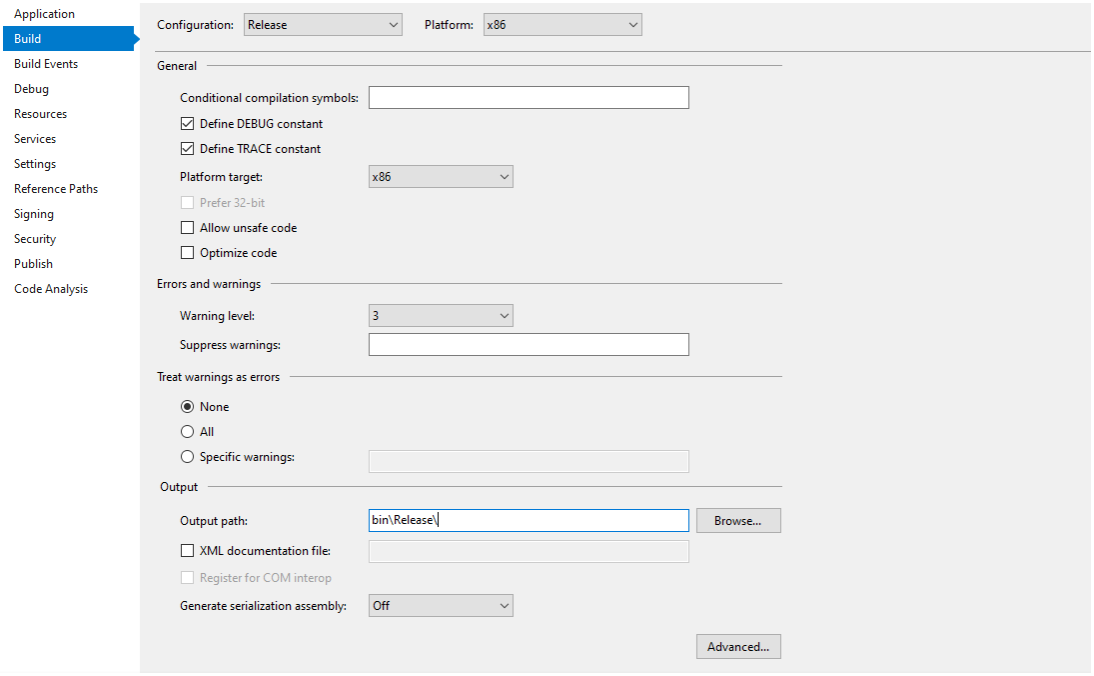

Figure 5: Platform Target for 32-bit Machine

For users who have to build the application in a **64-bit** machine, the target platform has to be **Any CPU** as shown in the screen shot below.

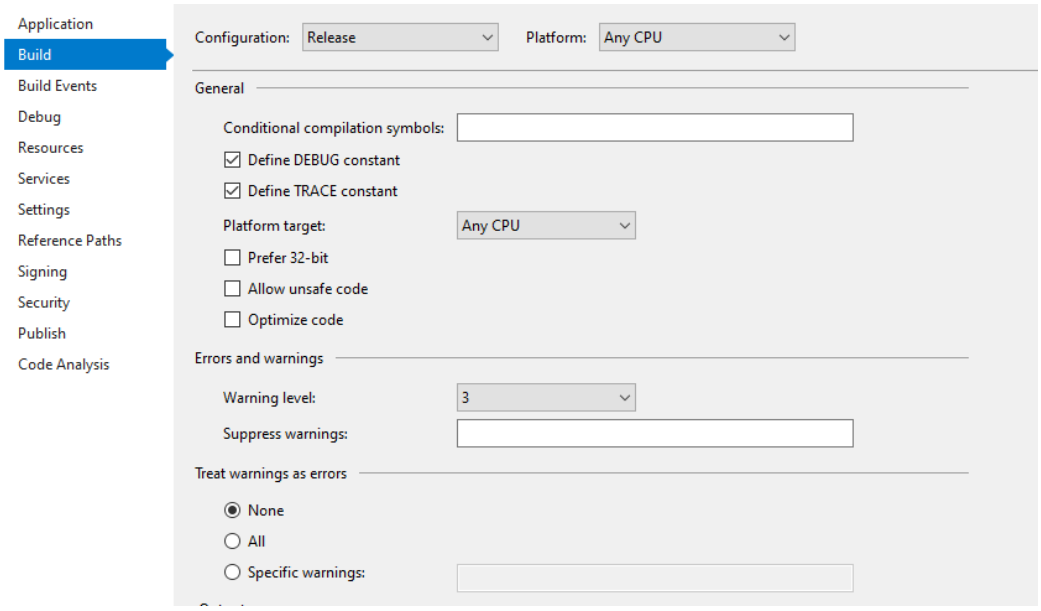

Figure 5: Platform Target for 64-bit Machine

3.1. Step 1

Start Visual Studio 2017 and choose **New Project**. The following window will be displayed.

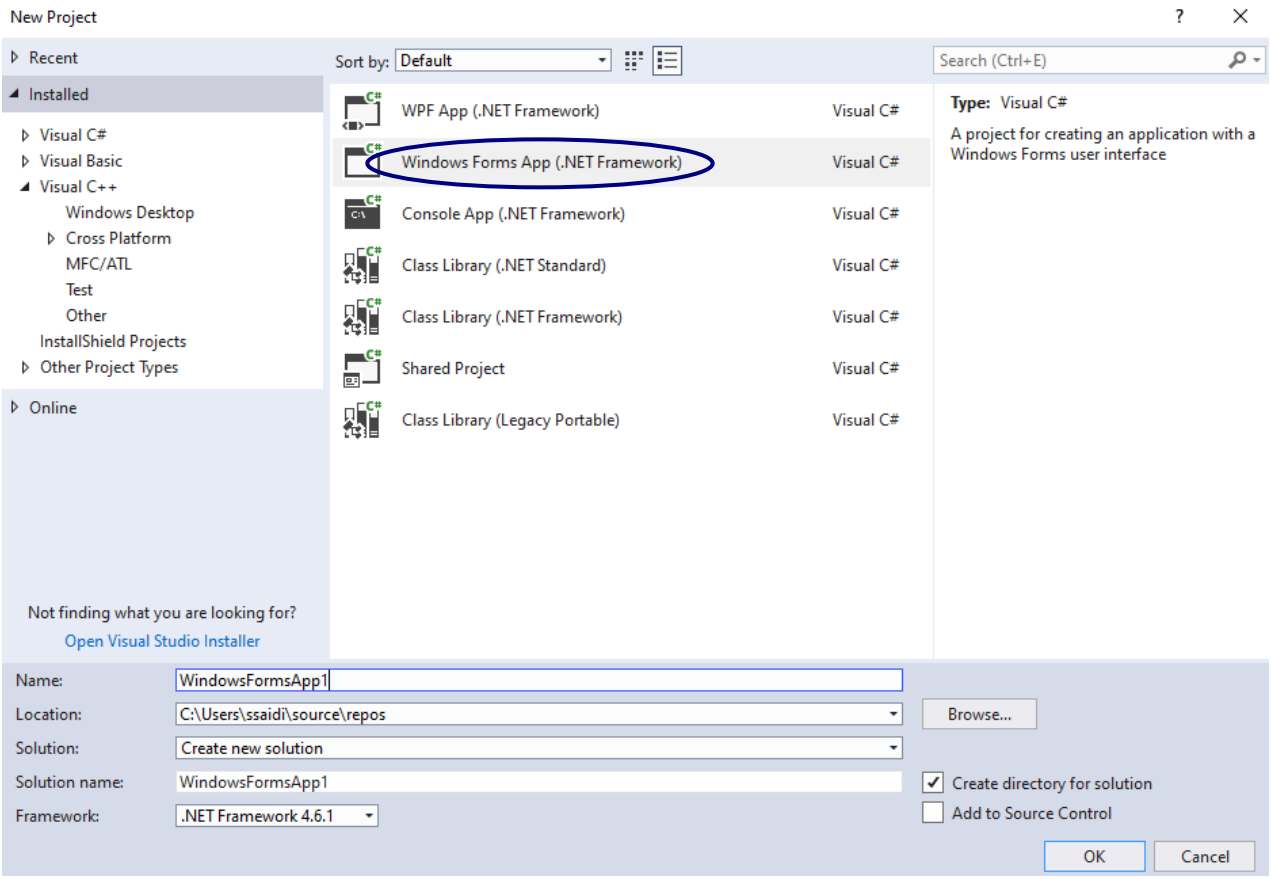

Figure 6: New Windows Form Project

Choose Visual C# **Windows Forms Application** Project and then click **OK**.

3.2. Step 2

A project named WindowsFormsApp1 with a form called Form1 will be automatically created.

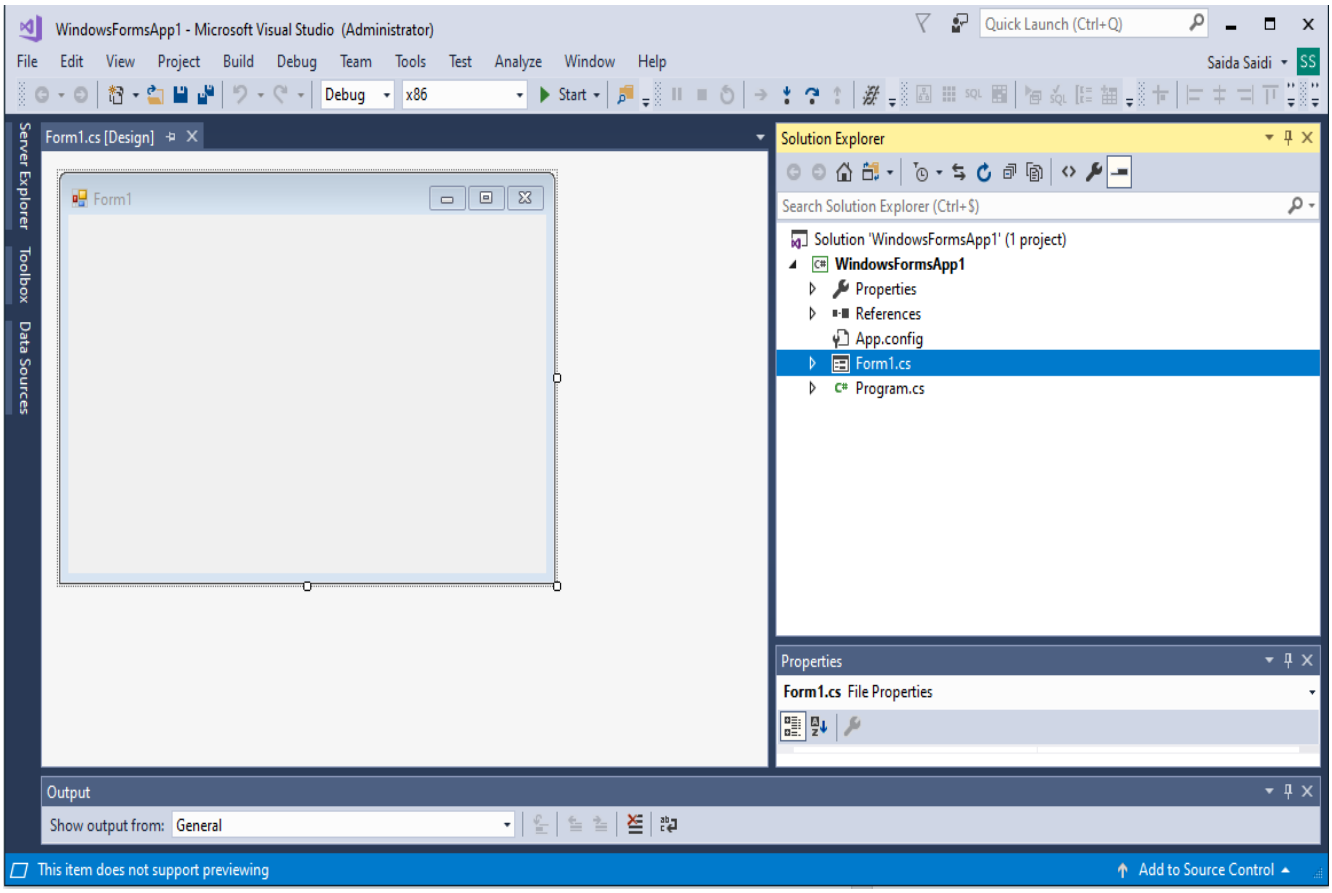

Figure 7: Windows Forms Project Template

3.3. Step 3

Add reference to the OPC .NET Client Toolkit as shown below.

1. Right click on References then click Add Reference… from the displayed menu.

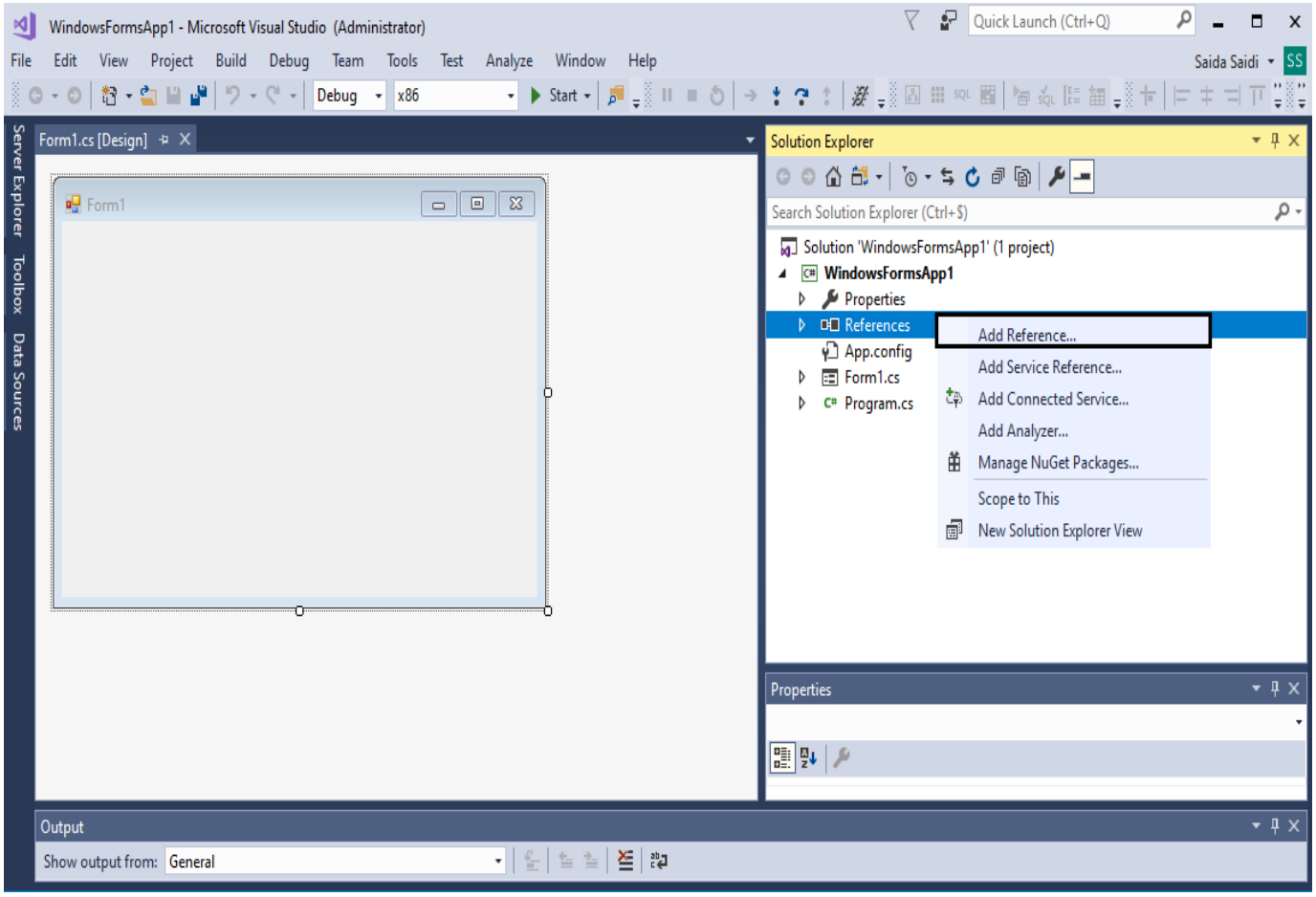

Figure 8: Solution Explorer

2. Select Browse tab from the displayed Add Reference window.

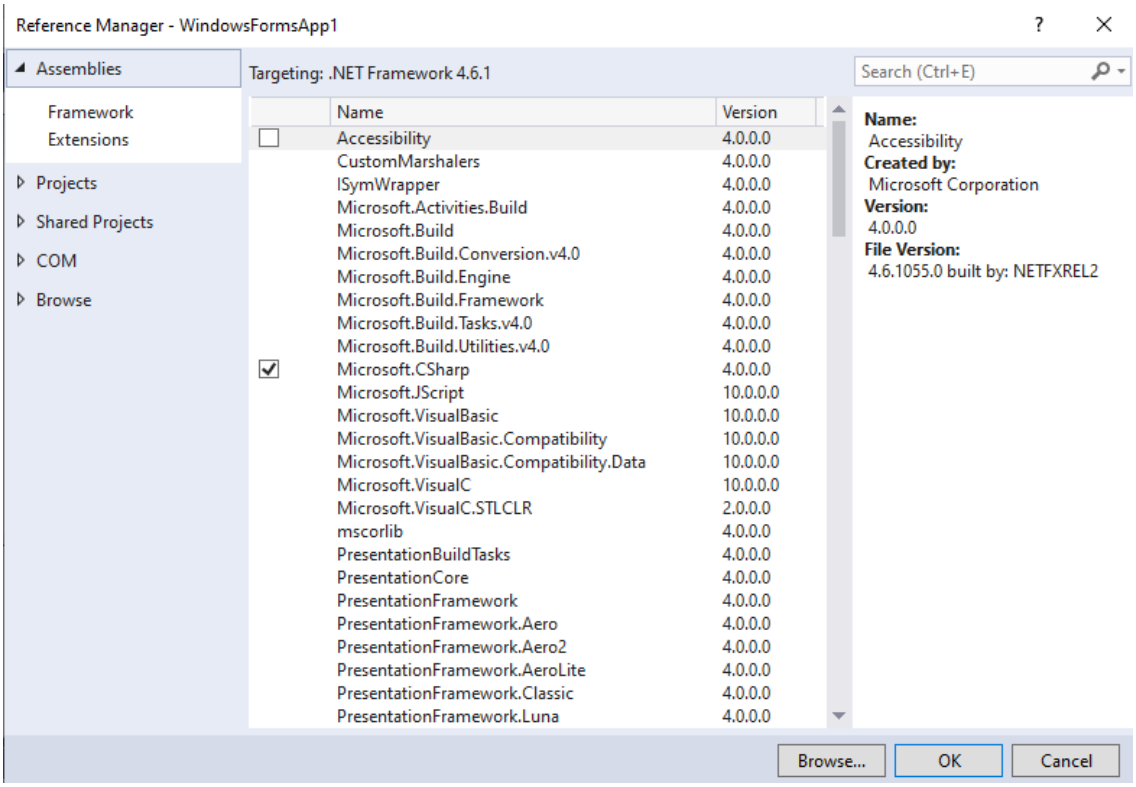

Figure 9: Adding a Reference

3. Select **IntegrationObjects.OPCNetClientSDK.dll** located under: **.:\Program Files\Integration Objects\Integration Objects' OPC .NET Client Toolkit\bin**

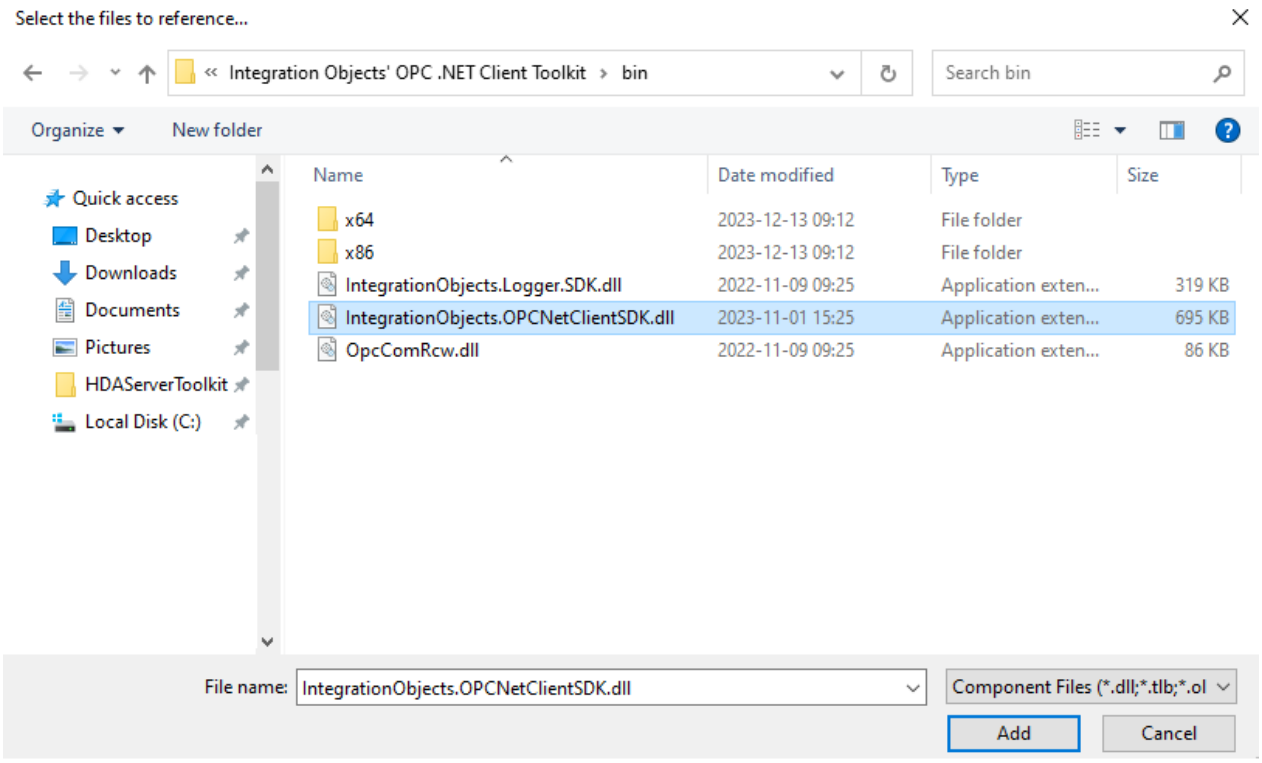

Figure 10: Choosing a Reference

If OPC .NET Client Toolkit license changed to full license, make sure to reference the new full version .dll files on the application sample project or to copy them under the output folder.

4. Runtime Deployment

In order to deploy the client application in the runtime machine, follow the steps below:

1. Create a new folder
2. Copy the following files to this new folder:
   - ConfigOPCClientSDK.ini
   - IntegrationObjects.Logger.SDK.dll
   - IntegrationObjects.OPCNetClientSDK.dll
   - License.dll
   - OpcComRcw.dll
   - The application executable and any other custom depending assembly
3. Move the folder to the runtime machine

Make sure that OPC .NET Client Toolkit is not installed in the runtime machine and that the path of the application folder does not include the words "Debug" or "Release".

5. Removing OPC .NET Client Toolkit

To uninstall the OPC .NET Client Toolkit, follow the steps below:

1. Click the **"Uninstall .NET Client Toolkit"** shortcut available in the start menu, as shown in the figure below:

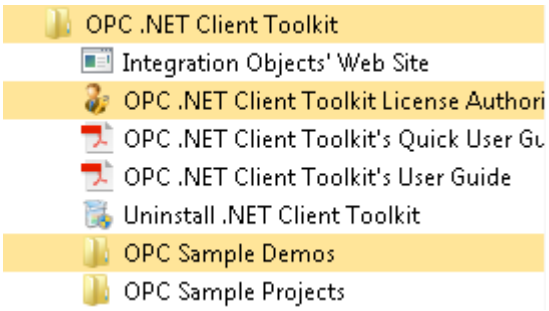

Figure 11: Uninstall Shortcut in the Start Menu

The following dialog box will appear:

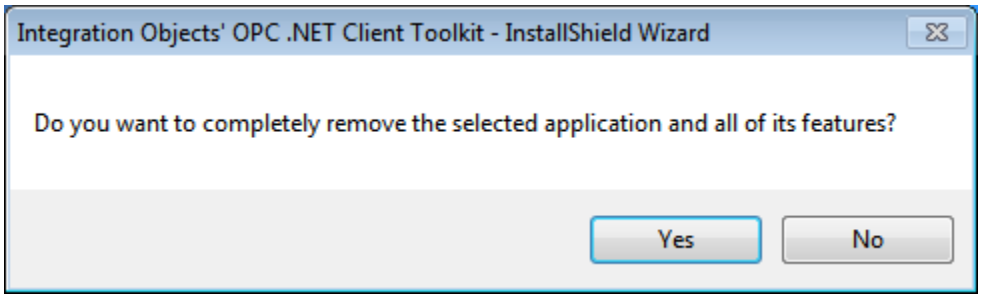

Figure 12: Uninstall the OPC .NET Client Toolkit

2. Click the **Yes** button to start uninstalling.

3. The wizard will then take you through the removal steps. At the end, click **Finish** when the un-installation is complete.

If you are using Windows 10, Windows Server 2012, Windows Server 2016 or Windows Server 2019 operating system, the uninstaller needs to be run from the start menu as shown below.

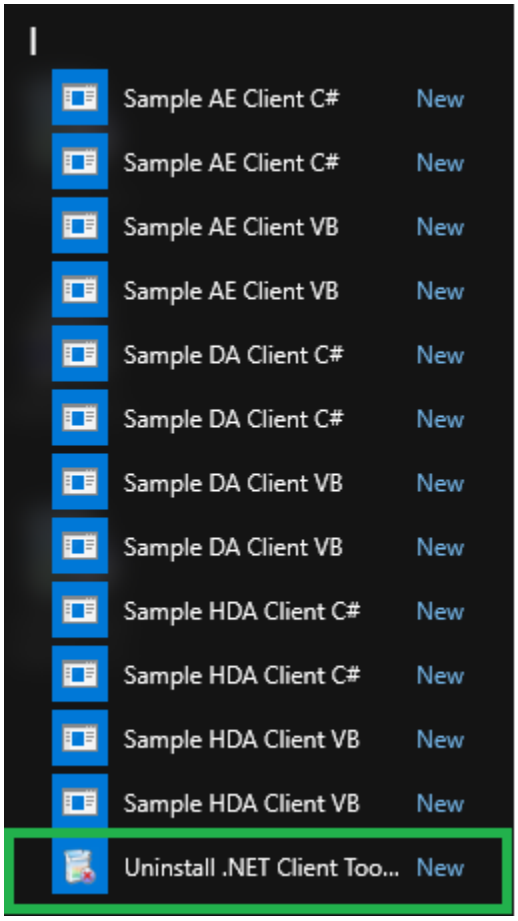

Figure 13: Windows 10 Startup Menu Uninstall Shortcut

The OPC .NET Client Toolkit can also be manually removed as follows:

1. Go to the **Control Panel**.
2. Click **Programs and Features**
3. In the **Programs and Features** dialog screen, select **Integration Objects' OPC .NET Client Toolkit**.
4. Click **Uninstall** then **OK**.

HOW TO INTERACT WITH THE OPC DA .NET CLIENT TOOLKIT

1. Initialization of the API

After the DLL initialization, the client application should hold responsibility for properly initializing the API. Thus, the OPC .NET Client Toolkit exports the API function named OPCDAManager that performed the basic initialization functions.

2. DA Servers Auto-Discovery

OPC .NET Client Toolkit provides a way to automatically discover and find all available OPC DA servers on the network. The following table describes the parameters of the **ListAllDA205Servers** function.

```
void ListAllDA205Servers(string          host,
                         out ArrayList   lServers,
                         out string      strError)
```

Parameters

In/Out	Parameter	Description
In	**host**	Name of the machine that hosts OPC DA servers
Out	**lServers**	Contains the list of located OPC DA servers
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 2: Parameters of the ListAllDA205Servers Function

```
void ListAllDAServers(string CatID,
                      string host,
                      out ArrayList lServers,
                      out string strError)
```


customerservice@integrationobjects.com

Parameters

In/Out	Parameter	Description
In	**CatID**	The category of OPC Data Access Interface.
In	**host**	Name of the machine that hosts OPC DA servers
Out	**IServers**	Contains the list of located OPC DA servers
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 3: Parameters of the ListAllDAServers Function

Values for CatID Parameter:

OPC DA version	Value
OPC Data Access Servers Version 1.0	{63D5F430-CFE4-11d1-B2C8-0060083BA1FB}
OPC Data Access Servers Version 2.0	{63D5F432-CFE4-11d1-B2C8-0060083BA1FB}
OPC Data Access Servers Version 3.0	{CC603642-66D7-48f1-B69A-B625E73652D7}

Table 4: Values of the CatID Parameter

3. Server Management

3.1. Connect To Server

a. Connect

This function establishes a connection to the server identified by **progidOPCserver** and located on the machine **strHostName**. The following table describes the parameters of the **Connect** function.

```
int Connect(string          progidOPCserver,
            string            strHostName,
            out int           index ,
            out string        strError)
```

Parameters

In/Out	Parameter	Description
In	**progidOPCserver**	The server progID
In	**strHostName**	The server node name
Out	**index**	If the call succeeds, this parameter will contain a unique identifier for the connection. It should be passed in any further call to the server.
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 5: Parameters of the Connect Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_OK	The operation succeeded.

Table 6: Returned Codes of the Connect Function

b. Connect with Authentication

This function establishes a connection to the server with user authentication. The following table describes the parameters of the **ConnectWithAuthentication** function.

```
int ConnectWithAuthentication(string    progIDOPCserver,
                        string    strHostName,
                        string    strDomain,
                        string    strLogin,
                        string    strPassword,
                        out int    index,
                        out string    strError)
```

Parameters

In/Out	Parameter	Description
In	**progIDOPCserver**	The server progID
In	**strHostName**	The server node name
In	**strDomain**	The domain name. In case the server machine is in the workgroup, this parameter can be replaced with the machine name instead of the domain name.
In	**strLogin**	The login of the server's user.
In	**strPassword**	The password of the server's user.
Out	**index**	If the call succeeds, this parameter will contain a unique identifier for the connection. It should be passed in any further call to the server.
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 7: Parameters of the ConnectWithAuthentication Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_OK	The operation succeeded.

Table 8: Returned Codes of the ConnectWithAuthentication Function

3.2. Disconnect from Server

To disconnect from the server, the client may use the method called **Disconnect**, providing the **ServerHandle** returned by the Connect function. The following table describes the parameters of this function.

```
int Disconnect(int ServerHandle, out string strError)
```

Parameters

In/Out	Parameter	Description
In	**ServerHandle**	The identifier of the connection
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 9: Parameters of the Disconnect Function

Return Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_OK	The operation succeeded.

Table 10: Returned Codes of the Disconnect Function

3.3. GetServerStatus

This function returns information about the current server status. The following table describes the parameters deployed and needed by the **GetServerStatus** function.

```
int GetServerStatus(int           serverindex,
                    out long      ftStartTime,
                    out long      ftCurrentTime,
                    out long      ftLastUpdateTime,
                    out int       dwGroupCount,
                    out int       dwBandWidth,
                    out short     wMajorVersion,
                    out short     wMinorVersion,
```

```
out short        wBuildNumber,
out short        wReserved,
out string       szVendorInfo,
out int          ServerState,
out string       strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
Out	**ftStartTime**	Time the server was started
Out	**ftCurrentTime**	The current time as known by the server
Out	**ftLastUpdateTime**	The time the server sent the last data value update to the client
Out	**dwGroupCount**	The total number of groups
Out	**dwBandWidth**	The bandwidth of the server
Out	**wMajorVersion**	The major version of the server software
Out	**wMinorVersion**	The minor version of the server software
Out	**wBuildNumber**	The build number of the server software
Out	**szVendorInfo**	Vender additional information
Out	**ServerState**	The current status of the server
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 11: Parameters of the GetServerStatus Function

The ServerState Parameter Values

Status	Value
OPC_STATUS_RUNNING	1

OPC_STATUS_FAILED	2
OPC_STATUS_NOCONFIG	3
OPC_STATUS_SUSPENDED	4
OPC_STATUS_TEST	5

Table 12: Values of the ServerState Parameter

Return Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
S_FALSE	The operation failed. Refer to the strError parameter for more details about the error.
S_OK	The operation succeeded.

Table 13: Returned codes of GetServerStatus Function

3.4. Subscription to Callbacks Functions

For asynchronous requests, the user must provide callback functions used by the API to asynchronously inform the user that there is incoming data, or send back the request results.

According to the OPC DA specification, the client must provide five types of callback functions:

1. OnDataChange
2. OnReadComplete
3. OnWriteComplete
4. OnCancelComplete
5. OnShutdown

4. Browsing

4.1. GetItemID

This function gives the fully qualified ItemID. The following table describes the parameters of this function.

```
int GetItemID(int serverindex,
              string szItemDataID,
```

```
                    out string szItemID,
                    out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**szItemDataID**	The name of the BRANCH or the LEAF at the current level
Out	**szItemID**	The resulting ItemID
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 14: Parameters of the GetItemID Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
S_OK	The operation succeeded.

Table 15: Returned Codes of the GetItemID Function

4.2. QueryOrganization

This function gives the name space type of the considered server. The following table describes the parameters of this function.

```
int   QueryOrganization(int        serverindex,
                        out int    pNameSpaceType,
                        out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
Out	**pNameSpaceType**	Place to put OPCNAMESPACE result which will be OPC_NS_HIERARCHIAL or OPC_NS_FLAT
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 16: Parameters of the QueryOrganization Function

The pNameSpaceType parameter values

Status	Value
OPC_NS_HIERARCHIAL	1
OPC_NS_FLAT	2

Table 17: Values of the pNameSpaceType Parameter

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
S_OK	The operation succeeded.

Table 18: Returned Codes of the QueryOrganization Function

4.3. ChangeBrowsePosition

This function provides the way to move in a hierarchical space. The following table describes the parameters of this function.

```
int ChangeBrowsePosition(int      serverindex,
                         int      dwBrowseDirection,
                         string   szString,
                         out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**dwBrowseDirection**	OPC_BROWSE_UP or OPC_BROWSE_DOWN or OPC_BROWSE_TO
In	**szString**	The name of the branch to move into
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 19: Parameters of the ChangeBrowsePosition Function

The dwBrowseDirection parameter values

Status	Value
OPC_BROWSE_UP	1
OPC_BROWSE_DOWN	2
OPC_BROWSE_TO	3

Table 20 : Values of the dwBrowseDirection Parameter

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_OK	The operation succeeded.

Table 21: Returned Codes of the ChangeBrowsePosition Function

4.4. BrowseOPCItemIDs

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

```
int BrowseOPCItemIDs(int             serverindex,
                     int             dwBrowseFilterType,
                     string          szFilterCriteria,
                     short           vtDataTypeFilter,
                     int             dwAccessRightsFilter,
                     out ArrayList   ppIEnumString,
                      out string      strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**dwBrowseFilterType**	OPC_BRANCH or OPC_LEAF or OPC_FLAT
In	**szFilterCriteria**	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	**ppIEnumString**	The returned interface pointer
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will

		contain the returned error message. Otherwise, it will contain an empty string.

Table 22: Parameters of the BrowseOPCItemIDs Function

The dwBrowseFilterType parameter values

Status	Value
OPC_BRANCH	1
OPC_LEAF	2
OPC_FLAT	3

Table 23 : Values of the dwBrowseFilterType Parameter

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_FALSE	There is nothing to enumerate
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
OPC_E_INVALIDFILTER	The filter string was not valid
S_OK	The operation succeeded.

Table 24: Returned Codes of the BrowseOPCItemIDs Function

4.5. IOPCBrowse

4.5.1. Browse

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

```
BrowseElement[] Browse (int                  serverindex,
                        ItemIdentifier       itemID,
                        BrowseFilters        filters,
                        OPCBrowsePosition    position)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**itemID**	The identifier of branch which is the target of the search
In	**filters**	The filters to use to limit the set of child elements returned
In	**position**	An object used to continue a browse that could not be completed

Table 25: Parameters of the Browse Function

The filters parameter values

Parameter	Description
MaxElementsReturned	Server must not return any more elements than this value
BrowseFilter	An enumeration {All, Branch, Item} specifying which subset of browse elements to return
ElementNameFilter	A wildcard string that conforms to the Visual Basic LIKE operator, which will be used to filter Element names.
VendorFilter	A server specific filter string. This is entirely free format and may be entered by the user via an EDIT field.

ReturnAllProperties	Server must return all properties which are available for each of the returned elements. If true, PropertyIDs is ignored.
ReturnPropertyValues	Server must return the property values in addition to the property names.
PropertyIDs	An array of Property IDs to be returned with each element. if ReturnAllProperties is true, PropertyIDs is ignored and all properties are returned.

Table 26: Parameters of the Filter

Returned BrowseElement parameters

Parameter	Description
Name	A descriptive name for element that is unique within a branch
ItemName	The primary identifier for the element within the server namespace
ItemPath	A secondary identifier for the element within the server namespace
IsItem	Whether the element refers to an item with data that can be accessed
HasChildren	Whether the element has children
Properties	Server must return the property values in addition to the property names
PropertyIDs	The set of properties for the element

Table 27: Parameters of the BrowseElement

The browseFilter values

Status	Value
All	0
Branch	1

Item	2

Table 28: Values of the BrowseFilter

4.5.2. GetProperties

This function returns a list of properties for each item. The following table describes the parameters of this function.

```
ItemPropertyCollection[] GetProperties (int                serverindex,
                                        BrowseFilters      filters,
                                        String[]           ItemIDs)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**itemID**	The identifier of branch which is the target of the search
In	**filters**	The filters to use to limit the set of child elements returned
In	**ItemIDs**	A list of item identifiers.

Table 29: Parameters of the GetProperties Function

Returned ItemPropertyCollection parameters

Parameter	Description
ItemName	The primary identifier for the item within the server namespace.
ItemPath	A secondary identifier for the item within the server namespace.
ResultID	The error id for the result of an operation on an item.
DiagnosticInfo	Vendor specific diagnostic information.

ItemProperty	Contains a description of an element in the server address space.

Table 30: Parameters of the ItemPropertyCollection

ItemProperty parameters

Parameter	Description
ID	The property identifier.
Description	A short description of the property.
DataType	The data type of the property.
Value	The value of the property.
ItemName	The primary identifier for the property if it is directly accessible as an item.
ItemPath	The secondary identifier for the property if it is directly accessible as an item.
ResultID	The error id for the result of an operation on a property.
DiagnosticInfo	Vendor specific diagnostic information.

Table 31: Parameters of the ItemProperty

4.6. IOPCItemIO

4.6.1. Read

Reads one or more values, qualities and timestamps for the items specified. This is functionally similar to the IOPCSyncIO::Read method.

```
ItemValueResult[] Read(int                serverindex,
                       Item[]                 items)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection

In	**items**	The set of items to read.

Table 32: Parameters of the Read Function

The Item parameters

Parameter	Description
ReqType	The data type to use when returning the item value.
MaxAge	The oldest (in milliseconds) acceptable cached value when reading an item.
MaxAgeSpecified	Whether the Max Age is specified.
Active	Whether the server should send data change updates.
ActiveSpecified	Whether the Active state is specified.
Deadband	The minimum percentage change required to trigger a data update for an item.
DeadbandSpecified	Whether the Deadband is specified.
SamplingRate	How frequently the server should sample the item value.
SamplingRateSpecified	Whether the Sampling Rate is specified.
EnableBuffering	Whether the server should buffer multiple data changes between data updates.
EnableBufferingSpecified	Whether the Enable Buffering is specified.

Table 33: Parameters of the Item

The ItemValueResult parameters

Parameter	Description
ItemName	The primary identifier for an item within the server namespace.
ItemPath	A secondary identifier for an item within the server namespace.

Value	The item value.
Quality	The quality of the item value.
QualitySpecified	Whether the quality is specified.
Timestamp	The UTC timestamp for the item value.
TimestampSpecified	Whether the timestamp is specified.
DiagnosticInfo	Vendor specific diagnostic information
ResultID	The error id for the result of an operation on a property.

Table 34: Parameters of the ItemValueResult

Returned ResultID

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
E_TIMEDOUT	The server did not respond within the specified timeout period.
E_NETWORK_ERROR	Could not connect to server.
E_ACCESS_DENIED	The server refused the connection.
E_NOTSUPPORTED	If a client attempts to write any value, quality, timestamp combination and the server does not support the requested combination (which could be a single quantity such as just timestamp), then the server will not

	perform any write and will return this error code.
S_OK	The operation succeeded.

Table 35: Returned ResultID of the Read Function

4.6.2. WriteVQT

Writes one or more values, qualities and timestamps for the items specified. This is functionally similar to the IOPCSyncIO2::WriteVQT except that there is no associated group.

```
int WriteVQT(int serverindex,
             string[] pszItemIDs,
             OPCITEMVQT[] arrVal,
             out int[] IntError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**pszItemIDs**	The list of ItemIDs to be written.
In	**arrVal**	The list of OPCItemVQT structure. Each structure contains a value, quality and timestamp to be written to the corresponding ItemID.
Out	**IntError**	The list of errors resulting from the write (1 per item).

Table 36: Parameters of the WriteVQT Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid

E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
OPC_E_NOTSUPPORTED	The requested (value, quality, timestamp) combination not supported by the server.
S_OK	The operation succeeded.

Table 37: Returned Codes of the WriteVQT Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_RANGE	The value was out of range
OPC_S_CLAMP	The value was accepted but was clamped.
OPC_E_BADRIGHTS	The item is not writable
OPC_E_BADTYPE	The passed data type cannot be accepted for this item
OPC_E_INVALIDITEMID	The passed item handle was invalid.
E_FAIL	The Write failed for this item
OPC_E_UNKNOWNITEMID	The item is no longer available in the server address space.
S_xxx **E_xxx**	Vendor specific information

Table 38: Error Array Codes of the WriteVQT Function

5. DA Group

5.1. AddGroup

This function adds a group to the considered server. The following table describes the parameters of this function.

```
public int AddGroup(int          serverindex,
                    string       szName,
                    int          bActive,
                    int          dwRequestedUpdateRate,
                    int          hClientGroup,
                    int[]        pTimeBias,
                    float[]      pPercentDeadband,
                    int          dwLCID,
                    out int      phServerGroup,
                    out int      pRevisedUpdateRate,
                    bool         isSync,
                    out int      indexgroup,
                    out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**szName**	The name of the group
In	**bActive**	• TRUE if the Group is to be created as active • FALSE if the Group is to be created as inactive
In	**dwRequestedUpdateRate**	The fastest rate at which data changes may be sent to OnDataChange for items in this group
In	**hClientGroup**	The handle of the group specified by the client
In	**pTimeBias**	The initial Time Bias (in minutes) for the Group
In	**pPercentDeadband**	The percent change in an item value that will cause a subscription callback for that value to a client

In	**dwLCID**	The language to be used by the server when returning values
Out	**phServerGroup**	The unique server-generated handle to the newly created group
Out	**pRevisedUpdateRate**	The server returns the value it will actually use for the Update Rate which may differ from the requested Update Rate
In	**IsSynch**	False for Ondatachange and Asynchronous modes. True if the read mode of the groups is set to synchronous.
Out	**indexgroup**	The returned identifier of the group
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 39. Parameters of the AddGroup Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_OK	The operation succeeded.

Table 40: Returned Codes of the AddGroup Function

5.2. RemoveGroup

This function removes groups from the considered server. The following table describes the parameters of this function.

```
int RemoveGroup(int         serverindex,
                int         groupindex,
```

```
int          hServerGroup,
int          bForce,
out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**hServerGroup**	The unique server-generated handle
In	**bForce**	Forces deletion of the group
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 41: Parameters of the RemoveGroup Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
OPC_S_INUSE	Was not removed because references exist. Group will be marked as deleted, and will be removed automatically by the server when all references to this object are released.
S_OK	The operation succeeded.

Table 42: Returned Codes of the RemoveGroup function

5.3. CreateGroupEnumerator

This function creates enumerators for the groups. The following table describes the parameters of this function.

```
int CreateGroupEnumerator(int          serverindex,
                          int          dwScope,
                          string s_riid,
                          out object   ppUnk,
                          out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**dwScope**	Indicates the class of groups to be enumerated
In	**s_riid**	The interface requested. This must be IID_IEnumUnknown or IID_IEnumString.
Out	**ppUnk**	The returned interface
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 43: Parameters of the CreateGroupEnumerator Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_FALSE	There are no groups which satisfy the request

E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
E_NOINTERFACE	The interface(riid) asked for is not supported by the server
S_OK	The operation succeeded.

Table 44: Returned Codes of the CreateGroupEnumerator Function

5.4. GetState

This function gives the state of the considered group. The following table describes the parameters of this function.

```
int GetState(    int         indexserver,
                 int         indexgroup,
                 out int     pUpdateRate,
                 out int     pActive,
                 out string  ppName,
                 out int     pTimeBias,
                 out float   pPercentDeadband,
                 out int     pLCID,
                 out int     phClientGroup,
                 out int     phServerGroup,
                  out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
Out	**pUpdateRate**	The current update rate (in milliseconds)
Out	**pActive**	The current active state of the group
Out	**ppName**	The current name of the group
Out	**pTimeBias**	The Time Zone Bias of the group
Out	**pPercentDeadband**	The percent change in an item value that will cause an exception report of that value to a client

Out	**pLCID**	The current LCID for the group
Out	**phClientGroup**	The client supplied group handle
Out	**phServerGroup**	The server supplied group handle
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 45: Parameters of the GetState Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
S_OK	The operation succeeded.

Table 46: Returned Codes of the GetState Function

5.5. CloneGroup

This function creates a copy of a group with a different name. The following table describes the parameters of this function.

```
int CloneGroup(int          indexserver,
               int          indexgroup,
               string       szName,
               out int      indexClonedGroup,
               out int      phServerGroup,
               out string strError)
```

Parameters

In/Out	Parameter	Description

In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**szName**	Name of the group
Out	**indexClonedGroup**	The identifier of the cloned group
Out	**phServerGroup**	The server supplied group handle
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 47: Parameters of the CloneGroup Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
E_NOINTERFACE	The interface(riid) asked for is not supported by the server
OPC_E_DUPLICATENAME	Duplicate name not allowed.
S_OK	The operation succeeded.

Table 48: Returned Codes of the CloneGroup Function

5.6. SetName

This function gives a new name for the specified group. The following table describes the parameters of this function.

```
int SetName(int indexserver,
            int      indexgroup,
```

```
    string    szName,
out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The group index
In	**szName**	New name for group
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 49: Parameters of the SetName Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
OPC_E_DUPLICATENAME	Duplicate name not allowed.
S_OK	The operation succeeded.

Table 50: Returned Cods of the SetName Function

5.7. SetState

This function sets properties of the considered group. The following table describes the parameters of this function.

```
int SetState(int            indexserver,
             int            indexgroup,
             int[]          pRequestedUpdateRate,
             out int        pRevisedUpdateRate,
             bool[]         pActive,
             int[]          pTimeBias,
             float[]        pPercentDeadband,
             int[]          pLCID,
             int[]          phClientGroup,
            out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**pRequestedUpdateRate**	The requested update rate (in milliseconds)
out	**pRevisedUpdateRate**	Closest update rate the server is able to provide for this group
In	**pActive**	The active state of the group
In	**pTimeBias**	The Time Zone Bias of the group
In	**pPercentDeadband**	The percent change in an item value that will cause an exception report of that value to a client
In	**pLCID**	The LCID for the group
In	**phClientGroup**	The client group handle
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 51: Parameters of the SetState Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
OPC_S_UNSUPPORTEDRATE	The server does not support the requested data rate but will use the closest available rate.
S_OK	The operation succeeded.

Table 52: Returned Codes of the SetState Function

6. DA Items

6.1. AddItems

This function adds OPC items to selected group. The following table describes the parameters of this function.

```
int AddItems(int            indexserver,
             int            indexgroup,
             int            dwCount,
             string[]       itemID,
             bool[]         active,
             int[]          handle,
             string[]       AccesPath,
             VarEnum[]      Requestedtype,
             out int[]      ppErrors,
             out int[]      HandleServer,
             out System.Runtime.InteropServices.VarEnum[]
             CanonicalDataType,
             out int[]      AccessRights,
             out string strError)
```

Parameters

In/Out	Parameter	Description
In	**indexserver**	The identifier of the connection
In	**indexgroup**	The identifier of the group
In	**dwCount**	The number of items to be added
In	**AccessPath**	The access path the server should associate with this item
In	**itemID**	The string that uniquely identifies the OPC data item
In	**active**	The active state of the item
In	**handle**	The client handle
In	**Requestedtype**	The data type requested by the client
Out	**ppErrors**	Returned errors
Out	**HandleServer**	The server handle used to refer to this item
Out	**CanonicalDataType**	The native data type
Out	**AccessRights**	Indicates if this item is read only, write only or read/write
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 53: Parameters of the AddItems Function

Error array Returned Errors

Result	Description
E_FAIL	The function failed
S_OK	The function was successful
OPC_E_INVALIDITEMID	The ItemID is not syntactically valid

OPC_E_BADTYPE	The requested data type cannot be returned for this item
OPC_E_UNKNOWNPATH	The item's access path is not known to the server.

Table 54: Error array Returned Codes of the AddItems Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
OPC_E_PUBLIC	Cannot add items to a public group
S_OK	The operation succeeded.

Table 55: Returned Codes of the AddItems Function

6.2. RemoveItems

This function removes items from the considered group. The following table describes the parameters of this function.

```
int RemoveItems(int          indexserver,
                int          indexgroup,
                int[]        arrHSrv,
                out int[]    arrErr,
                out string   strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection

In	**indexgroup**	The identifier of the group
In	**arrHSrv**	The server item handles
Out	**arrErr**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 56: Parameters of the RemoveItems Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
OPC_E_PUBLIC	Cannot remove items from a public group
S_OK	The operation succeeded.

Table 57: Returned Codes of the RemoveItems Function

Error array Returned Codes

Result	Description
S_OK	The corresponding item was removed.
OPC_E_INVALIDHANDLE	The corresponding Item handle was invalid.

Table 58: Error array Returned Codes of the RemoveItems Function

6.3. SetActiveState

This function changes the state of items to active or inactive. The following table describes the parameters of this function.

```
int SetActiveState(int            indexserver,
                   int            indexgroup,
                   int[]          arrHSrv,
                   bool           activate,
                   out int[]      arrErr,
                   out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**indexgroup**	The identifier of the group
In	**arrHSrv**	The server item handles
In	**activate**	The active state of the item
Out	**arrErr**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 59: Parameters of the SetActiveState Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.

E_INVALIDARG	An argument to the function was invalid.
S_OK	The operation succeeded.

Table 60: Returned Codes of the SetActiveState Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The corresponding Item handle was invalid.

Table 61: Error Array Codes of the SetActiveState Function

6.4. SetClientHandles

This function gives client handles for the considered items. The following table describes the parameters of this function.

```
int SetClientHandles(int          indexserver,
                     int          groupindex,
                     int[]        arrHSrv,
                     int[]        arrHClt,
                     out int[]    arrErr,
                     out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**indexgroup**	The identifier of the group
In	**arrHSrv**	The server item handles
In	**arrHClt**	The client handle
Out	**arrErr**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the

		returned error message. Otherwise, it will contain an empty string.

Table 62: Parameters of the SetClientHandles Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
E_INVALIDARG	An argument to the function was invalid.
S_OK	The operation succeeded.

Table 63: Returned Codes of the SetClientHandles Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The corresponding Item handle was invalid.

Table 64: Error Array Codes of the SetClientHandles Function

6.5. SetDatatypes

This function changes the data type of the considered items. The following table describes the parameters of this function.

```
int SetDatatypes( int             indexserver,
                  int             groupindex,
                   int[]          arrHSrv,
                   VarEnum[]      arrVT,
                   out int[]      arrErr,
```

```
                                    out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**indexgroup**	The identifier of the group
In	**arrHSrv**	The server item handles
In	**arrVT**	The new requested data types to be stored
Out	**arrErr**	Indicates which of the items were successfully affected
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 65: Parameters of the SetDataTypes Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
E_INVALIDARG	An argument to the function was invalid.
S_OK	The operation succeeded.

Table 66: Returned Codes of the SetDatatypes Function

Error array Returned Codes

Result	Description

S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The corresponding Item handle was invalid.
OPC_E_BADTYPE	The requested data type cannot be supported for this item.

Table 67: Error Array Codes of the SetDatatypes Function

6.6. ValidateItems

This function validates the items of the specific group. The following table describes the parameters of this function.

```
int ValidateItems(int                   indexserver,
                  int                   groupindex,
                  int                   dwCount,
                  string[]              AccessPath,
                  string[]              ItemID,
                  bool[]                Active,
                  int[]                 HandleClient,
                  VarEnum[]             RequestedDataType,
                  bool                  blobUpd,
                  out int[]             Error,
                  out int[]             HandleServer,
                  out VarEnum[]         CanonicalDataType,
                  out int[]             AccessRights,
                  out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**indexgroup**	The identifier of the group
In	**dwCount**	The number of items to be validated
In	**AccessPath**	The access path the server should associate with this item
In	**ItemID**	The string that identifies the OPC data item
In	**Active**	The active state of the item

In	**HandleClient**	The client handle
In	**RequestedDataType**	The data type requested by the client
In	**blobUpd**	The returned updated blobs
Out	**Error**	Returned errors
Out	**HandleServer**	The server item handles
Out	**CanonicalDataType**	The native data type
Out	**AccessRights**	Indicates if this item is read only, write only or read/write
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 68: Parameters of the ValidateItems Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
S_OK	The operation succeeded.

Table 69: Returned Codes of the ValidateItems Function

Error array Returned Codes

Result	Description

customerservice@integrationobjects.com

S_OK	The function was successful.
OPC_E_INVALIDITEMID	The ItemID is not valid
OPC_E_BADTYPE	The requested data type cannot be supported for this item.
OPC_E_UNKNOWNITEMID	The ItemID is not in the server address space
OPC_E_UNKNOWNPATH	The item's access path is not known to the server.
E_FAIL	The function was unsuccessful for this item.

Table 70: Error Array Codes of the ValidateItems Function

6.7. CreateEnumerator

This function creates an enumerator for the items in the considered group. The following table describes the parameters of this function.

```
Void CreateEnumerator(int          indexserver,
                      int          indexgroup,
                      out object   ppUnk)
```

Parameters

In/Out	Parameter	Description
In	**indexserver**	The identifier of the connection
In	**indexgroup**	The identifier of the group
Out	**ppUnk**	The returned interface

Table 71: Parameters of the CreateEnumerator Function

6.8. QueryAvailableProperties

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

```
int QueryAvailableProperties(int          indexserver,
                             string       itemID,
                             out int[]    PropertyID,
                             out string[] Description,
```

```
out VarEnum[]   DataType,
out string strError)
```

Parameters

In/Out	Parameter	Description
In	**indexserver**	The identifier of the connection
In	**itemID**	The ItemID for which the caller wants to know the available properties
Out	**PropertyID**	DWORD IDs for the returned properties. These IDs can be passed to GetItemProperties or LookupItemIDs
Out	**Description**	A brief vendor supplied text description of each property
Out	**DataType**	The data type that will be returned for this property by GetItemProperties
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 72: Parameters of the QueryAvailableProperties Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_FALSE	The operation completed with one or more errors. Refer to strError for more information.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory

OPC_E_UNKNOWNITEMID	The ItemID is not in the server address space
S_OK	The operation succeeded.

Table 73: Returned Codes of the QueryAvailableProperties Function

6.9. GetItemProperties

This function returns a list of the current data values for the requested item IDs. The following table describes the parameters of this function.

```
int GetItemProperties(int              indexserver,
                      string           itemID,
                      int[]            propertyIDs,
                      out int[]        PropertyID,
                      out int[]        Error,
                      out object[]     Data)
```

Parameters

In/Out	Parameter	Description
In	**indexserver**	The identifier of the connection
In	**itemID**	The string that identifies the OPC data item
Out	**propertyIDs**	The list of required properties' identifiers
Out	**Data**	The list of required properties' values
Out	**Error**	The returned errors

Table 74: Parameters of the GetItemProperties Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.

S_FALSE	The operation completed with one or more errors. Refer to Error for more information.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
OPC_E_UNKNOWNITEMID	The ItemID is not in the server address space
S_OK	The operation succeeded.

Table 75: Returned Codes of the GetItemProperties Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALID_PID	The passed Property ID is not defined for this item.

Table 76: Error Array Codes of the GetItemProperties Function

6.10. LookupItemIDs

This function returns a list of ITEMIDs (if available) for each of the requested items IDs. These indicate the ITEMID which could be added to an OPCGroup and used for more efficient access to the data corresponding to the Item Properties. The following table describes the parameters of this function.

```
int LookupItemIDs(int             indexserver,
                  string          itemID,
                  int[]           propertyIDs,
                  out int[]       PropertyID,
                  out int[]       Error,
                  out string[]    newItemID)
```

Parameters

In/Out	Parameter	Description
In	**indexserver**	The identifier of the connection

In	**itemID**	The ItemID for which the caller wants to look up the list of properties
Out	**propertyIDs**	DWORDIDs for the requested properties. These IDs were returned by QueryAvailableProperties
Out	**newItemID**	The returned list of ItemIDs
Out	**Error**	Error array indicating whether each New ItemID was returned

Table 77: Parameters of the LookupItemIDs Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to Error for more information.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
OPC_E_UNKNOWNITEMID	The ItemID is not in the server address space
S_OK	The operation succeeded.

Table 78: Returned Codes of the LookupItemIDs Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALID_PID	The passed Property ID is not defined for this item.

E_FAIL	The passed Property ID could not be translated into an ItemID.

Table 79: Error Array Codes of the LookupItemIDs Function

6.11. SetKeepAlive

Clients can set the keep-alive time for a subscription to cause the server to provide client callbacks on the subscription when there are no new events to report.

```
void SetKeepAlive(int serverindex,
                  int groupindex,
                  int KeepAliveTime,
                  out int RevisedKeepAliveTime)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**KeepAliveTime**	The maximum amount of time (in milliseconds) between subscription callbacks.
Out	**RevisedKeepAlive Time**	The KeepAliveTime the server is actually providing, which may differ from KeepAliveValue.

Table 80: Parameters of the SetKeepAlive method

6.12. GetKeepAlive

This function returns the currently active keep-alive time for the subscription.

```
void GetKeepAlive(int serverindex,
     int groupindex,
      out int KeepAliveTime)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group

In	**KeepAliveTime**	The maximum amount of time (in milliseconds) between subscription callbacks.

Table 81: Parameters of the GetKeepAlive method

6.13. SetItemDeadband

This function overrides the deadband specified for the group for each item.

```
int SetItemDeadband(int serverindex,
                    int groupindex,
                    int count,
                   int[] arrHSrv,
                   float[] PercentDeadband,
                   out int[] IntErrors)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**Count**	The number of items to be affected
In	**arrHSrv**	Array of Server items handles.
In	**PercentDeadband**	Array of deadband values. Each value must be from 0.0 to 100.0, which is the percentage of the change allowed per update period.
Out	**IntErrors**	Indicates the results of setting the deadband for each item.

Table 82: Parameters of the SetItemDeadband Function

Return Codes

Return Code	Description

CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
E_INVALIDARG	An argument to the function was invalid.
OPC_E_DEADBAND NOTSUPPORTED	The server does not support deadband.
S_OK	The operation succeeded.

Table 83: Returned Codes of the SetItemDeadband Function

IntError array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The corresponding Item handle was invalid.
OPC_E_DEADBANDNOT S UPPORTED	The item handle does not support deadband.
E_INVALIDARG	Requested deadband was not in the range of 0.0 to 100.0.

Table 84: IntError Array Codes of the SetItemDeadband Function

6.14. GetItemDeadband

This function returns the deadband values for each of the requested items.

```
int GetItemDeadband(int serverindex,
                    int groupindex,
                    int count,
                   int[] arrHSrv,
                   out float[] PercentDeadband,
                   out int[] IntErrors)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**Count**	The number of items to be affected
In	**arrHSrv**	Array of Server items handles.
Out	**PercentDeadband**	Array of deadband values. Each value must be from 0.0 to 100.0, which is the percentage of the change allowed per update period.
Out	**IntErrors**	Indicates the results of setting the deadband for each item.

Table 85: Parameters of the GetItemDeadband Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to IntErrors for more information.
E_INVALIDARG	An argument to the function was invalid.
OPC_E_DEADBAND NOTSUPPORTED	The server does not support deadband.
S_OK	The operation succeeded.

Table 86: Returned Codes of the SetItemDeadband Function

IntErrors array Returned Codes

Result	Description

customerservice@integrationobjects.com

S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The corresponding Item handle was invalid.
OPC_E_DEADBANDNOTS UPPORTED	The item handle does not support deadband.
OPC_E_DEADBANDNOTSET	The item deadband has not been set for this item.

Table 87: IntErrors Array Codes of the GetItemDeadband Function

6.15. ClearItemDeadband

This function clears the individual item PercentDeadband, effectively reverting them back to the deadband value set in the group.

```
int ClearItemDeadband(int serverindex,
                      int groupindex,
                      int Count,
                     int[] arrHSrv,
                     out int[] IntErrors)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**Count**	The number of items to be affected
In	**arrHSrv**	Array of Server items handles.
Out	**IntErrors**	Indicates the results of setting the deadband for each item.

Table 88: Parameters of the ClearItemDeadband Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid

E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to IntErrors for more information.
E_INVALIDARG	An argument to the function was invalid.
OPC_E_DEADBAND NOTSUPPORTED	The server does not support deadband.
S_OK	The operation succeeded.

Table 89: Returned Codes of the ClearItemDeadband Function

IntErrors array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The corresponding Item handle was invalid.
OPC_E_DEADBANDNOT S UPPORTED	The item handle does not support deadband.
OPC_E_DEADBANDNOT SET	The item deadband has not been set for this item.

Table 90: IntErrors Array Codes of the ClearItemDeadband Function

7. Synchronous Methods

7.1. IOPCSyncIO

7.1.1. Read

This function synchronously reads the value, quality, and timestamp information for items of the considered group. The following table describes the parameters of this function.

```
int Read(int                serverindex,
         int                groupindex,
         int            src,
         int[]              arrHSrv,
         out int[]      arrErr,
```

```
out int[]       HandleClient,
out object[]    DataValue,
out long[]      TimeStamp,
out short[]     Quality,
out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**src**	The data source (CACHE or DEVICE)
In	**arrHSrv**	The server item handles
Out	**arrErr**	Returned errors
Out	**HandleClient**	The client handle
Out	**DataValue**	The item value to read
Out	**TimeStamp**	The item timestamp for the value to read
Out	**Quality**	The quality flag for the value to read
Out	**StrError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 91: Parameters of the Read Function

The src parameter values

Status	Value
OPC_DS_CACHE	1
OPC_DS_DEVICE	2

Table 92 : Values of the src Parameter

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
S_OK	The operation succeeded.

Table 93: Returned Codes of the Read Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_BADRIGHTS	The item is not readable
OPC_E_INVALIDHANDLE	The passed item handle was invalid.
E_FAIL	The Read failed for this item
OPC_E_UNKNOWNITEMID	The item is no longer available in the server address space.
S_xxx **E_xxx**	Vendor specific information

Table 94: Error Array Codes of the Read Function

7.1.2. Write

This function synchronously writes values to items of the considered group. The following table describes the parameters of this function.

```
int Write(int          serverindex,
```

```
int         groupindex,
int[]       arrHSrv,
object[]    arrVal,
out int[]   arrErr,
out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**arrHSrv**	The server item handles
In	**arrVal**	The item value to write
Out	**arrErr**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 95: Parameters of the Write Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
S_OK	The operation succeeded.

Table 96: Returned Codes of the Write Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_RANGE	The value was out of range
OPC_S_CLAMP	The value was accepted but was clamped.
OPC_E_BADRIGHTS	The item is not readable
OPC_E_INVALIDHANDLE	The passed item handle was invalid.
E_FAIL	The Read failed for this item
OPC_E_UNKNOWNITEMID	The item is no longer available in the server address space.
S_xxx **E_xxx**	Vendor specific information

Table 97: Error Array Codes of the Write Function

7.2. IOPCSyncIO2

7.2.1. Read

This function synchronously reads the value, quality, and timestamp information for items of the considered group. The following table describes the parameters of this function.

```
int   Read2(int             serverindex,
            int             groupindex,
            int             src,
            int[]           arrHSrv,
            out int[]       arrErr,
            out int[]       HandleClient,
            out object[]    DataValue,
            out long[]      TimeStamp,
            out short[]     Quality,
            out string      strError)
```

Parameters

In/Out	Parameter	Description

In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**src**	The data source (CACHE or DEVICE)
In	**arrHSrv**	The server item handles
Out	**arrErr**	Returned errors
Out	**HandleClient**	The client handle
Out	**DataValue**	The item value to read
Out	**TimeStamp**	The item timestamp for the value to read
Out	**Quality**	The quality flag for the value to read
Out	**StrError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 98: Parameters of the Read2 Function

The src parameter values

Status	Value
OPC_DS_CACHE	1
OPC_DS_DEVICE	2

Table 99 : Values of the src Parameter

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.

S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
S_OK	The operation succeeded.

Table 100: Returned Codes of the Read2 Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_BADRIGHTS	The item is not readable
OPC_E_INVALIDHANDLE	The passed item handle was invalid.
E_FAIL	The Read failed for this item
OPC_E_UNKNOWNITEMID	The item is no longer available in the server address space.
S_xxx **E_xxx**	Vendor specific information

Table 101: Error Array Codes of the Read2 Function

7.2.2. Write

This function synchronously writes values to items of the considered group. The following table describes the parameters of this function.

```
int Write2(int        serverindex,
           int        groupindex,
           int[]      arrHSrv,
           object[]   arrVal,
           out int[]  arrErr,
          out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**arrHSrv**	The server item handles
In	**arrVal**	The item value to write
Out	**arrErr**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 102: Parameters of the Write2 Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
S_OK	The operation succeeded.

Table 103: Returned Codes of the Write2 Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.

OPC_E_RANGE	The value was out of range
OPC_S_CLAMP	The value was accepted but was clamped.
OPC_E_BADRIGHTS	The item is not readable
OPC_E_INVALIDHANDLE	The passed item handle was invalid.
E_FAIL	The Read failed for this item
OPC_E_UNKNOWNITEMID	The item is no longer available in the server address space.
S_xxx **E_xxx**	Vendor specific information

Table 104: Error Array Codes of the Write2 Function

7.2.3. ReadMaxAge

This function synchronously reads one or more values, qualities and timestamps for the items specified. This is functionally similar to the OPCSyncIO::Read method except no source is specified (DEVICE or CACHE).

```
int ReadMaxAge(int serverindex,
               int groupindex,
               int[] phServer,
               int[] pdwMaxAge,
               out object[] ppvValues,
               out short[] ppwQualities,
               out DateTime[] ppftTimeStamps,
               out int[] ppErrors,
               out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**phServer**	The server item handles

In	**pdwMaxAge**	An indicator, requested in milliseconds, that determines from where the data is obtained. If the information in the cache is within the MaxAge, then the data will be obtained from the cache. Otherwise the server must access the device for the requested information.
Out	**ppvValues**	The items' values
Out	**ppwQualities**	The qualities associated to the returned values
Out	**ppftTimeStamps**	The timestamps of the returned values
Out	**arrErr**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 105: Parameters of the ReadMaxAge Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
S_OK	The operation succeeded.

Table 106: Returned Codes of the ReadMaxAge Function

Error array Returned Codes

Result	Description

S_OK	The function was successful.
OPC_E_BADRIGHTS	The item is not readable
OPC_E_INVALIDHANDLE	The passed item handle was invalid.
E_FAIL	The Read failed for this item
OPC_E_UNKNOWNITEMID	The item is no longer available in the server address space.
S_xxx **E_xxx**	Vendor specific information

Table 107: Error Array Codes of the ReadMaxAge Function

7.2.4. WriteVQT

This function Writes one or more values, qualities and timestamps for the items specified. This is functionally similar to the IOPCSyncIO::Write except that Quality and Timestamp may be written.

```
int WriteVQT(int serverindex,
             int groupindex,
             int[] arrHSrv,
             OPCITEMVQT[] arrVal,
             out int[] arrErr,
             out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**arrHSrv**	The server item handles
In	**arrVal**	The list of OPCItemVQT structure. Each structure contains a value, quality and timestamp to be written to the corresponding ItemID.
Out	**arrErr**	Returned errors

Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 108: Parameters of the WriteVQT Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
OPC_E_NOTSUPPORTED	The requested (value, quality, timestamp) combination not supported by the server.
S_OK	The operation succeeded.

Table 109: Returned Codes of the WriteVQT Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_RANGE	The value was out of range
OPC_S_CLAMP	The value was accepted but was clamped.
OPC_E_BADRIGHTS	The item is not writable
OPC_E_BADTYPE	The passed data type cannot be accepted for this item

OPC_E_INVALIDHANDLE	The passed item handle was invalid.
E_FAIL	The Write failed for this item
OPC_E_UNKNOWNITEMID	The item is no longer available in the server address space.
S_xxx **E_xxx**	Vendor specific information

Table 110: Error Array Codes of the WriteVQT Function

8. Asynchronous Methods

8.1. IOPCAsyncIO2

8.1.1. Read

This function asynchronously reads the value, quality, and timestamp information for the specified items. The following table describes the parameters of this function.

```
int Read( int          serverindex,
          int          groupindex,
          int[]        arrHSrv,
          int          transactionID,
          out int      cancelID,
          out int[]    arrErr,
          out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**arrHSrv**	The server item handles
In	**transactionID**	An identifier created by the client and passed to the server in this call

Out	**cancelID**	The server-generated cancelID
Out	**arrErr**	This indicates to the client which of the items was successfully read
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 111: Parameters of the Read Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
CONNECT_E_NOCONNECTION	The client has not registered a callback through IConnectionPoint::Advise.
S_OK	The operation succeeded.

Table 112: Returned Codes of the Asynchronous Read Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_BADRIGHTS	The item is not readable
OPC_E_INVALIDHANDLE	The passed item handle was invalid.

E_FAIL	The Read failed for this item
OPC_E_UNKNOWNITEMID	The item is no longer available in the server address space.
S_xxx **E_xxx**	Vendor specific information

Table 113: Error Array Codes of the Asynchronous Read Function

8.1.2. Write

This function asynchronously writes values to the items specified. The following table describes the parameters of this function.

```
int Write( int         serverindex,
           int         groupindex,
           int[]       arrHSrv,
           object[]    arrVal,
           int         transactionID,
           out int     cancelID,
           out int[]   arrErr,
           out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**arrHSrv**	The server item handles
In	**arrVal**	The values to be written
In	**transactionID**	An identifier created by the client and passed to the server in this call
Out	**cancelID**	The server-generated cancelID
Out	**arrErr**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the

		returned error message. Otherwise, it will contain an empty string.

Table 114: Parameters of the Write Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
CONNECT_E_NOCONNECTION	The client has not registered a callback through IConnectionPoint::Advise.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
S_OK	The operation succeeded.

Table 115: Returned Codes of the Asynchronous Write Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_BADRIGHTS	The item is not writable
OPC_E_INVALIDHANDLE	The passed item handle was invalid.
E_FAIL	The Write failed for this item
OPC_E_UNKNOWNITEMID	The item is no longer available in the server address space.
S_xxx **E_xxx**	Vendor specific information

Table 116: Error Array Codes of the Asynchronous Write Function

8.1.3. Refresh2

This function forces OnDataChange callbacks for all active items. The following table describes the parameters of this function.

```
int Refresh2(int            serverindex,
             int            groupindex,
             int            sourceMode,
             int            transactionID,
             out int        cancelID,
             out string     strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**sourceMode**	The data source (CACHE or DEVICE). Refer to the "*values of the src parameter*" table
In	**transactionID**	An identifier created by the client and passed to the server in this call.
Out	**cancelID**	The server-generated cancelID
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 117: Parameters of the Refresh2 Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid

E_FAIL	The operation failed.
CONNECT_E_NOCONNECTION	The client has not registered a callback through IConnectionPoint::Advise.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
S_OK	The operation succeeded.

Table 118: Returned Codes of the Refresh2 Function

8.1.4. Cancel2

This function requests the server to cancel a transaction. The following table describes the parameters of this function.

```
int Cancel2( int       serverindex,
             int       groupindex,
             int       cancelID,
             out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**cancelID**	The server-generated cancelID
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 119: Parameters of the Cancel2 Function

Returned Codes

Return Code	Description

customerservice@integrationobjects.com

CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_OK	The operation succeeded.

Table 120: Returned Codes of the Cancel2 Function

8.1.5. SetEnable

This function enables or disables OnDataChange callback of the considered group. The following table describes the parameters of this function.

```
int SetEnable( int    serverindex,
               int    groupindex,
               bool   doEnable,
               out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Serverindex**	The identifier of the connection
In	**indexgroup**	The identifier of the group
In	**doEnable**	Enable or disable in the OnDataChange callback
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 121: Parameters of the SetEnable Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.

CONNECT_E_NOCONNECTION	The client has not registered a callback through IConnectionPoint::Advise.
S_OK	The operation succeeded.

Table 122: Returned Codes of the SetEnable Function

8.1.6. GetEnable

This function gives the enable value of the considered group. The following table describes the parameters of this function.

```
int GetEnable( int          serverindex,
               int          groupindex,
               out bool     isEnabled,
               out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**indexgroup**	The identifier of the group
Out	**isEnabled**	The returned result
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 123: Parameters of the GetEnable Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
CONNECT_E_NOCONNECTION	The client has not registered a callback through IConnectionPoint::Advise.

S_OK	The operation succeeded.

Table 124: Returned Codes of the GetEnable Function

8.2. IOPCAsyncIO3

8.2.1. Read3

This function asynchronously reads the value, quality and timestamp information for the items specified. The following table describes the parameters of this function.

```
int Read3(int          serverindex,
          int          groupindex,
          int[]        arrHSrv,
          int          transactionID,
          out int      cancelID,
          out int[]    arrErr,
          out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**arrHSrv**	The server item handles
In	**transactionID**	An identifier created by the client and passed to the server in this call
Out	**cancelID**	The server-generated cancelID
Out	**arrErr**	Indicates the result of the read operation for each item.
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 125: Parameters of the Read Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
CONNECT_E_NOCONNECTION	The client has not registered a callback through IConnectionPoint::Advise.
S_OK	The operation succeeded.

Table 126: Returned Codes of the Read3 Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_BADRIGHTS	The item is not readable
OPC_E_INVALIDHANDLE	The passed item handle was invalid.
E_FAIL	The Read failed for this item
OPC_E_UNKNOWNITEMID	The item is no longer available in the server address space.
S_xxx **E_xxx**	Vendor specific information

Table 127: Error Array Codes of the Read3 Function

8.2.2. Write3

This function asynchronously writes values to items of the considered group. The following table describes the parameters of this function.

```
int Write3(int        serverindex,
           int        groupindex,
           int[]      arrHSrv,
           object[]   arrVal,
           int        transactionID,
           out int    cancelID,
           out int[]  arrErr,
           out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**arrHSrv**	The server item handles
In	**arrVal**	The values to be written
In	**transactionID**	An identifier created by the client and passed to the server in this call
Out	**cancelID**	The server-generated cancelID
Out	**arrErr**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 128: Parameters of the Write Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.

S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
CONNECT_E_NOCONNECTION	The client has not registered a callback through IConnectionPoint::Advise.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
S_OK	The operation succeeded.

Table 129: Returned Codes of the Write3 Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_BADRIGHTS	The item is not writable
OPC_E_INVALIDHANDLE	The passed item handle was invalid.
E_FAIL	The write failed for this item
OPC_E_UNKNOWNITEMID	The item is no longer available in the server address space.
S_xxx **E_xxx**	Vendor specific information

Table 130: Error Array Codes of the Write3 Function

8.2.3. Refresh3

This function forces OnDataChange callbacks for all active items. The following table describes the parameters of this function.

```
int Refresh3( int          serverindex,
              int          groupindex,
              int          sourceMode,
              int          transactionID,
              out int      cancelID,
```

```
                    out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**sourceMode**	The data source (CACHE or DEVICE). Refer to the "*values of the src parameter*" table
In	**transactionID**	An identifier created by the client and passed to the server in this call.
Out	**cancelID**	The server-generated cancelID
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 131: Parameters of the Refresh3 Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
CONNECT_E_NOCONNECTION	The client has not registered a callback through IConnectionPoint::Advise.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
S_OK	The operation succeeded.

Table 132: Returned Codes of the Refresh3 Function

8.2.4. Cancel3

This function requests the server to cancel a transaction. The following table describes the parameters of this function.

```
int Cancel3( int      serverindex,
             int      groupindex,
             int      cancelID,
             out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**cancelID**	The server-generated cancelID
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 133: Parameters of the Cancel2 Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_OK	The operation succeeded.

Table 134: Returned Codes of the Cancel3 Function

8.2.5. SetEnable3

This function enables or disables OnDataChange callback of the considered group. The following table describes the parameters of this function.

```
int SetEnable3(int    serverindex,
               int    groupindex,
```

```
                bool    doEnable,
                out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Serverindex**	The identifier of the connection
In	**indexgroup**	The identifier of the group
In	**doEnable**	Enable or disable in the OnDataChange callback
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 135: Parameters of the SetEnable3 Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
CONNECT_E_NOCONNECTION	The client has not registered a callback through IConnectionPoint::Advise.
S_OK	The operation succeeded.

Table 136: Returned Codes of the SetEnable3 Function

8.2.6. GetEnable3

This function gives the enable value of the considered group. The following table describes the parameters of this function.

```
int GetEnable3(int          serverindex,
               int          groupindex,
```

```
                out bool     isEnabled,
                out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**indexgroup**	The identifier of the group
Out	**isEnabled**	The returned result
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 137: Parameters of the GetEnable3 Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
CONNECT_E_NOCONNECTION	The client has not registered a callback through IConnectionPoint::Advise.
S_OK	The operation succeeded.

Table 138: Returned Codes of the GetEnable3 Function

8.2.7. ReadMaxAge

This function asynchronously reads one or more values, qualities and timestamps for the items specified. This is functionally similar to the OPCASyncIO2::Read method except no source is specified (DEVICE or CACHE).

```
int ReadMaxAge(int serverindex,
               int groupindex,
```

```
int[]arrHSrv,
int[] pdwMaxAge,
int dwTransactionID,
 out int pdwCancelID,
out int[] ppErrors,
out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**arrHSrv**	The server item handles
In	**pdwMaxAge**	An indicator, requested in milliseconds, that determines where the data is obtained. If the information in the cache is within the MaxAge, then the data will be obtained from the cache, otherwise the server must access the device for the requested information.
In	**transactionID**	An identifier created by the client and passed to the server in this call
Out	**cancelID**	The server-generated cancelID
Out	**arrErr**	Indicates the result of the read operation for each item.
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 139: Parameters of the ReadMaxAge Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid

E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
CONNECT_E_NOCONNECTION	The client has not registered a callback through IConnectionPoint::Advise.
S_OK	The operation succeeded.

Table 140: Returned Codes of the Asynchronous ReadMaxAge Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_BADRIGHTS	The item is not readable
OPC_E_INVALIDHANDLE	The passed item handle was invalid.
E_FAIL	The Read failed for this item
OPC_E_UNKNOWNITEMID	The item is no longer available in the server address space.
S_xxx **E_xxx**	Vendor specific information

Table 141: Error Array Codes of the Asynchronous ReadMaxAge Function

8.2.8.WriteVQT

This function writes one or more values, qualities and timestamps for the items specified. Its functionally is similar to the IOPCAsyncIO2::Write method except that Quality and Timestamp may be written.

```
int WriteVQT(int serverindex,
             int groupindex,
```

```
int[] arrHSrv,
OPCITEMVQT[] arrVal,
int transactionID,
out int cancelID,
out int[] arrErr,
out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**arrHSrv**	The server item handles
In	**arrVal**	The list of OPCItemVQT structure. Each structure contains a value, quality and timestamp to be written to the corresponding ItemID.
In	**transactionID**	An identifier created by the client and passed to the server in this call
Out	**cancelID**	The server-generated cancelID
Out	**arrErr**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 142: Parameters of the WriteVQT Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to arrErr for more information.

E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
OPC_E_NOTSUPPORTED	The requested (value, quality, timestamp) combination not supported by the server.
CONNECT_E_NOCONNECTION	The client has not registered a callback through IConnectionPoint::Advise.
S_OK	The operation succeeded.

Table 143: Returned Codes of the Asynchronous WriteVQT Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_BADRIGHTS	The item is not writable
OPC_E_BADTYPE	The passed data type cannot be accepted for this item
OPC_E_INVALIDHANDLE	The passed item handle was invalid.
E_FAIL	The operation failed for this item
OPC_E_UNKNOWNITEMID	The item is no longer available in the server address space.
S_xxx **E_xxx**	Vendor specific information

Table 144: Error Array Codes of the Asynchronous WriteVQT Function

8.2.9. RefreshMaxAge

This function forces a callback to IOPCDataCallback::OnDataChange for all active items in the group.

```
int  RefreshMaxAge(   int serverindex,
                      int groupindex,
                      int pdwMaxAge,
                      int dwTransactionID,
```

```
                                    out int pdwCancelID,
                                    out string strError)
```

Parameters

In/Out	Parameter	Description
In	**serverindex**	The identifier of the connection
In	**groupindex**	The identifier of the group
In	**pdwMaxAge**	An indicator, requested in milliseconds, that determines where the data is obtained. If the information in the cache is within the MaxAge, then the data will be obtained from the cache, otherwise the server must access the device for the requested information.
In	**transactionID**	An identifier created by the client and passed to the server in this call
Out	**cancelID**	The server-generated cancelID
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 145: Parameters of the RefreshMaxAge Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
E_INVALIDARG	An argument to the function was invalid.
E_OUTOFMEMORY	Not enough memory
CONNECT_E_NOCONNECTION	The client has not registered a callback through IConnectionPoint::Advise.

S_OK	The operation succeeded.

Table 146: Returned Codes of the RefreshMaxAge Function

OPC DA CLIENT SAMPLES

This chapter describes the DA samples available within the installation of OPC .NET Client Toolkit.

1. Step 1: Create User Interface

Create the following user interface with respect to the indicated names of the components.

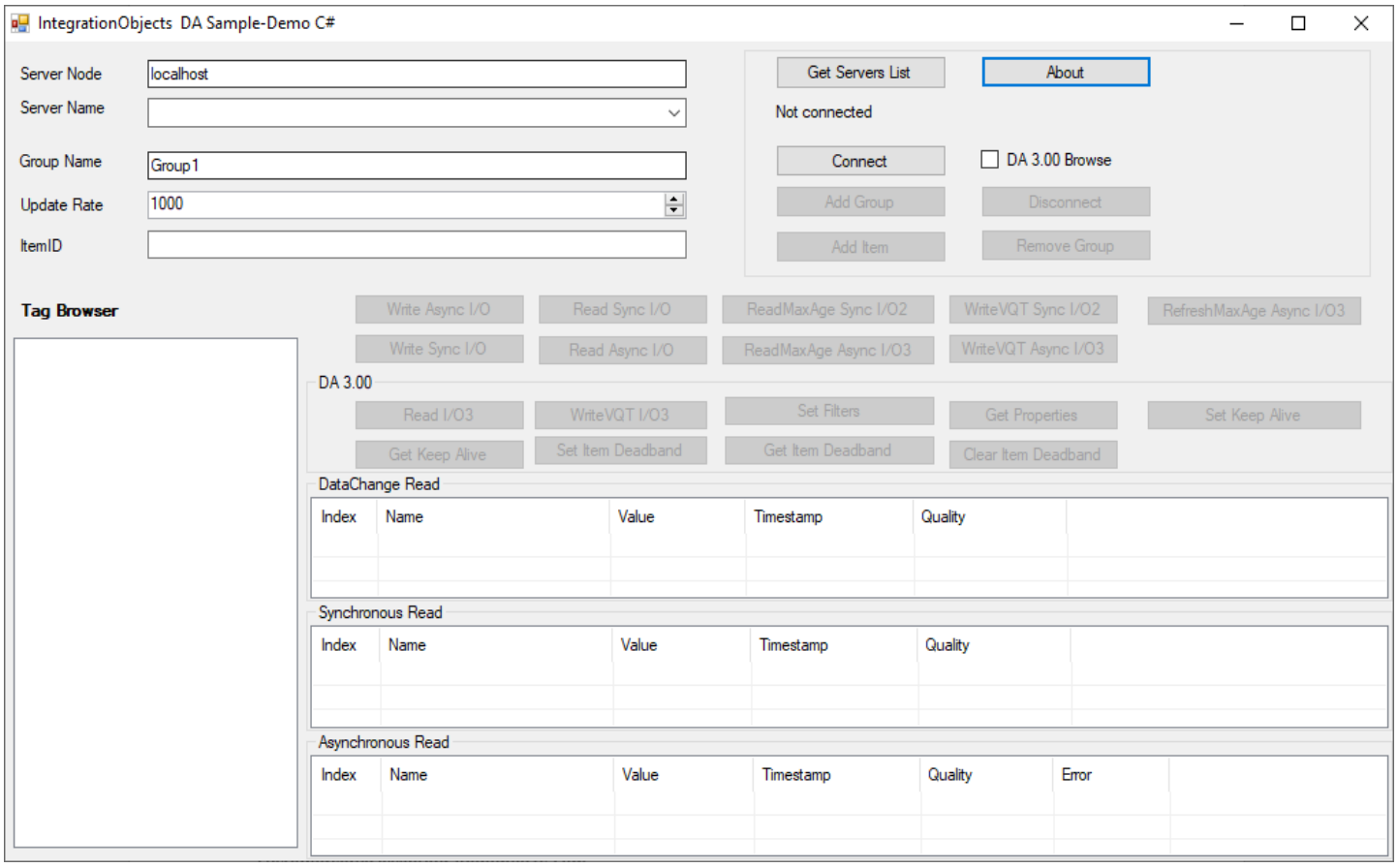

Figure 14: OPC DA Client - Creating a user interface window

2. Step 2: Initialize DA Manager and Subscribe to Callbacks

Perform the following declarations:

C#:
```
Private OPCDAManager objDaManager = null;
```

VB .NET:
```
Private objDaManager As OPC.DA.Manager.OPCDAManager = Nothing
```

Perform the following method's implementations:

VB .NET:
```
Public Sub display(ByVal GrpHandle As Integer, ByVal HandleClient() As Integer,
ByVal DataValue() As Object, ByVal TimeStamp() As Long, ByVal Quality() As
Short)


        If Me.listView1.InvokeRequired Then
            Dim delUpdateListView As New UpdateListViewSubs(AddressOf display2)
            Dim tmpobjArray As Object() = New Object() {GrpHandle, HandleClient,
DataValue, TimeStamp, Quality}
            Try
                'if it'll blocked change Invoke-->BeginInvoke
                Me.Invoke(delUpdateListView, tmpobjArray)
            Catch ec As Exception
            End Try
        Else
            display2(GrpHandle, HandleClient, DataValue, TimeStamp, Quality)
        End If
    End Sub
```

VB .NET:

```
Public Sub display2(ByVal GrpHandle As Integer, ByVal HandleClient() As Integer,
ByVal DataValue() As Object, ByVal TimeStamp() As Long, ByVal Quality() As
Short)
        Me.listView1.Items.Clear()
        Dim i As Integer = 0
        Do While (i < count)
            Try
                Dim Items() As String
                Dim list1 As List(Of String) = listItems(HandleClient(i))
                If list1(0).Equals(GrpHandle.ToString()) Then

                    If (Quality(i) = 192) Then
                        Items = New String() {HandleClient(i).ToString,
list1(1), DataValue(i).ToString, DateTime.FromFileTime(TimeStamp(i)).ToString,
"Good"}
                    Else
                        Items = New String() {HandleClient(i).ToString,
list1(1), DataValue(i).ToString, DateTime.FromFileTime(TimeStamp(i)).ToString,
Quality(i).ToString}
                    End If

                    Dim I1 As ListViewItem = New ListViewItem(Items)
                    Me.listView1.Items.Add(I1)
                End If
            Catch
            End Try
            i = (i + 1)
        Loop
    End Sub
Public Sub DataChange(ByVal transactionID As Integer, ByVal groupHandleClient As
Integer, ByVal masterQuality As Integer, ByVal masterError As Integer, ByVal
HandleClient() As Integer, ByVal DataValue() As Object, ByVal TimeStamp() As
Long, ByVal Quality() As Short, ByVal err() As Integer, ByVal GrpHandle As
Integer, ByVal SrvHandle As Integer)
        display(GrpHandle, HandleClient, DataValue, TimeStamp, Quality)
    End Sub
```

C#:

```
public void DisplayDatachange(int GrpHandle, int[] HandleClient,
                                object[] DataValue,
                                long[] TimeStamp,
                                short[] Quality)
        {


            if (this.listView1.InvokeRequired)
            {
                UpdateListDataChange delUpdateList = new
UpdateListDataChange(DisplayDatachange2);
                object[] tmpobjArray = new object[5];
                tmpobjArray[0] = GrpHandle;
                tmpobjArray[1] = HandleClient;
                tmpobjArray[2] = DataValue;
                tmpobjArray[3] = TimeStamp;
                tmpobjArray[4] = Quality;

                try
                {
                    this. BeginInvoke (delUpdateList, tmpobjArray);
                }
                catch (Exception ec) { }
            }
            else
                DisplayDatachange2(GrpHandle,HandleClient, DataValue, TimeStamp,
Quality);

        }
public void DisplayDatachange2(int GrpHandle ,
                            int[] HandleClient,
                            object[] DataValue,
                            long[] TimeStamp,
                            short[] Quality)
        {

            this.listView1.Items.Clear();


            for (int i = 0; i < count; i++)
            {
                    string[] Items;
                    List<string> list1 = listItems[HandleClient[i]];
                    if (list1[0].Equals(GrpHandle.ToString()))
                    {
                        if (Quality[i] == 192)
                            Items = new string[]{
                              HandleClient[i].ToString(),
                    list1[1],
                              DataValue[i].ToString(),
                              DateTime.FromFileTime( TimeStamp[i] ).ToString(),
                              "Good"};
```

C#:

```
                        else

                            Items = new string[]{
                              HandleClient[i].ToString(),
                    list1[1],
                              DataValue[i].ToString(),
                              DateTime.FromFileTime( TimeStamp[i] ).ToString(),
                              Quality[i].ToString()};

                        ListViewItem I1 = new ListViewItem(Items);
                        this.listView1.Items.Add(I1);
                    }
                }



        }

public void DataChange(
                  int                   transactionID,
                  int                   groupHandleClient,
                  int                   masterQuality,
                  int                   masterError,
                  int[]             HandleClient,
                  object[]          DataValue,
                  long[]                TimeStamp,
                  short[]               Quality,
                  int[]             Error,
                  int                   GrpHandle,
                  int                   SrvHandle )
        {
            try
            {
                DisplayDatachange(GrpHandle, HandleClient, DataValue, TimeStamp,
Quality);
            }
            catch(Exception ex)
            {
                string str = ex.Message;
            }
        }
```

Then, double-click the created form and add the following code to initialize DA Manager and to subscribe to callbacks. Here, we take the example of the DataChange callback:

C#:
```
objDaManager = new OPCDAManager();
if(objDaManager != null)
{objDaManager.MyDataChanged2+= new MyDataChangeEventHandler2 (this.DataChange);}
```

VB .NET:
```
 objDaManager = New OPC.DA.Manager.OPCDAManager
        If (Not (objDaManager) Is Nothing) Then
            objDaManager.MyDataChanged2 = AddressOf Me.DataChange
        End If
```

3. Step 3: List All Available DA Servers

Double-click the **server list button** and add the following code to list all available DA servers on the network:

C#:
```
try{
this.listserver.Items.Clear();
string host = txtHostName.Text.ToLower();
ArrayList listServer= new ArrayList();
string strError = string.Empty;
try{
if ((host == "localhost") || (host == "127.0.0.1"))
this.objDaManager.ListAllDA205ServersFromRegistry(txtHostName.Text, out
listServer,out strError);
else
this.objDaManager.ListAllDA205Servers(txtHostName.Text,out listServer,out
strError); }
catch (Exception e1){
try{
this.objDaManager.ListAllDA205ServersFromRegistry(txtHostName.Text, out
listServer ,out strError);}
catch (Exception e2){
string szlogmsg = string.Format("OPCNetClient:ListAllDAServers: " + e2.Message);
OPC_Log.TraceLog(OPC_Log.LOG_INFORMATION, szlogmsg);}}
for(int i=0 ;i<listServer.Count;i++){
this.listserver.Items.Add(listServer[i].ToString());}
try{
listserver.SelectedIndex = 0;}
catch (Exception exc){
MessageBox.Show("There is no available OPC DA Server", "Integration Objects' OPC
Net Client Toolkit", MessageBoxButtons.OK);
Return;}}
catch (Exception e2){throw new Exception(e2.Message,e2);}
this.connect.Enabled = true;
}
```

VB .NET:
```
Try
  Me.listserver.Items.Clear()
  Dim listServer As ArrayList = New ArrayList
  Dim strError As String = String.Empty
  Me.objDaManager.ListAllDA205Servers(txtHostName.Text, listServer,strError)
  Dim i As Integer = 0
  Do While (i < listServer.Count)
       Me.listserver.Items.Add(listServer(i).ToString)
       i = (i + 1)
   Loop
Catch ex As Exception
       Console.WriteLine(ex.ToString)
End Try
```

Perform the following declarations:

C#:
```
private string servername = null;
```

VB .NET:
```
Private servername As String = Nothing
```

Double-click the **listserver ComboBox** and add the following code:

C#:
```
servername=listserver.SelectedItem.ToString();
this.connect.Enabled=true;
```

VB .NET:
```
connect.Enabled = True
servername = listserver.SelectedItem.ToString
```

4. Step 4: Connect and Create Group

Perform the following declarations:

C#:
```
private string servername     =     null;
private int intServerIndex    =     -1;
private int servergroup       =     -1;
private int RevisedUpdateRate =     -1;
private int indexgroup        =     -1;
```

VB .NET:
```
Private servername As String = Nothing
Private intServerIndex As Integer = -1
Private servergroup As Integer = -1
Private RevisedUpdateRate As Integer = -1
Private indexgroup As Integer = -1
```

Double click the **connect button** and add the following code to connect to the selected server and to automatically create a new group.

C#:
```
if(servername!=null){
try{
      string strError = string.Empty;
      objDaManager.Connect(servername,txtHostName.Text,out intServerIndex,out
      strError);
}catch(Exception e3){
  throw new Exception(e3.Message,e3);}
try{
      string strError = string.Empty;
      objDaManager.AddGroup(intServerIndex,"group1",1,1000,1,null,null,0,out
servergroup,out RevisedUpdateRate,out indexgroup, out strError);}
catch(Exception e4){
  throw new Exception(e4.Message,e4);}
```

5. Step 5: Add Item

Perform the following declarations:

C#:
```
private string[] access = new string[100];
private int[]handle= new int[100];
private bool[] activestate = new bool[100];
private System.Runtime.InteropServices.VarEnum [] DataType = new
System.Runtime.InteropServices.VarEnum [100];
private int[] ARights = new int[100];
private System.Runtime.InteropServices.VarEnum [] CDataType = new
System.Runtime.InteropServices.VarEnum [100];
private int[] handleServer = new int[100];
private int[] err= new int[100];
private string[] Items = new string[100];
```

VB .NET:

```
Private access() As String = New String(100) {}
Private handle() As Integer = New Integer(100) {}
Private activestate() As Boolean = New Boolean(100) {}
Private DataType() As System.Runtime.InteropServices.VarEnum = New
System.Runtime.InteropServices.VarEnum(100) {}
Private ARights() As Integer = New Integer(100) {}
Private CDataType() As System.Runtime.InteropServices.VarEnum
Private handleServer() As Integer = New Integer(100) {}
Private err() As Integer = New Integer(100) {}
Private Items() As String = New String(100) {}
```

Double click the **AddItem button** and add the following code to add items:

C#:

```
Items[0]=this.ItemID.Text.ToString();
activestate[0]= true;
handle[0]=count;
access[0]="";
DataType[0]=System.Runtime.InteropServices.VarEnum.VT_EMPTY;
string strError = string.Empty;
objDaManager.AddItems(intServerIndex,indexgroup,1,Items,activestate,handle,acce
ss,DataType,out err,out handleServer,out CDataType,out ARights, out strError);
List<string> list1 = new List<string>();
list1.Add(indexgroup.ToString());
list1.Add(Items[0]);
listItems.Add(handle[0], list1);
count++;
```

VB .NET:

```
Items(0) = Me.ItemID.Text.ToString
activestate(0) = True
handle(0) = count
access(0) = ""
DataType(0) = System.Runtime.InteropServices.VarEnum.VT_EMPTY
string strError = string.Empty;
objDaManager.AddItems(intServerIndex, indexgroup, 1, Items, activestate,
handle, access, DataType, err, handleServer, CDataType, ARights, out strError)
Dim list1 As New List(Of String)()
            list1.Add(indexgroup.ToString())
            list1.Add(Items(0))
            listItems.Add(handle(0), list1)
count = (count + 1)
```

6. Read Data

C#:

```
string strError = string.Empty;
int[] arrErr = null;
int[] hClient = null;
long[] time = null;
short[] quality=null;
object[] data = null;
try
{
int ires= objDaManager.Read(intServerIndex, indexgroup, 1, handleServer, out arrErr, out
hClient, out data, out time, out quality,out strError);

       DisplaySync(ItemID.Text.ToString(),hClient, data, time, quality);
}catch{}

public void DisplaySync(string itemId,
                            int[] HandleClient,
                            object[] DataValue,
                            long[] TimeStamp,
                            short[] Quality)
        {
            this.listView3.Items.Clear();
            for (int i = 0; i < count; i++)
            {

                try
                {
                    string[] Items;
                    string strValue = string.Empty;
                    Type t = DataValue[i].GetType();
                    if (t.IsArray)
                    {

                        ArrayList arr = new ArrayList();

                        IEnumerable enumerable = DataValue[i] as IEnumerable;
                        if (enumerable != null)
                        {
                            foreach (object element in enumerable)
                            {
                                arr.Add(element);
                            }
                        }
                        strValue = string.Join(", ", Array.ConvertAll(arr.ToArray(), x =>
x.ToString()));

                    }
                    else
                    {
                        strValue = DataValue[i].ToString();
                    }
```

C#:

```
                    if (Quality[i] == 192)
                        Items = new string[]{
                                    HandleClient[i].ToString(),
                    itemId.ToString(),
                                    strValue,
                                    DateTime.FromFileTime( TimeStamp[i] ).ToString(),
                                    "Good"};

                    else

                    Items = new string[]{
                                    HandleClient[i].ToString(),
                    itemId.ToString(),
                                    strValue,
                                    DateTime.FromFileTime( TimeStamp[i] ).ToString(),
                                    Quality[i].ToString()};
                    ListViewItem I1 = new ListViewItem(Items);
                    this.listView3.Items.Add(I1);
                }
                catch
                {
                }
            }
        }
```

VB:

```
        Dim transactionID As Integer = 0
        Dim cancelID As Integer = -1
        Dim arrErr As Integer() = Nothing
        Dim hClient As Integer() = Nothing
        Dim time As Long() = Nothing
        Dim quality As Short() = Nothing
        Dim data As Object() = Nothing
        Dim strError As String = Nothing
        Try
            objDaManager.Read(intServerIndex, indexgroup, 1, handleServer, arrErr, hClient,
_
            data, time, quality, strError)

            DisplaySync(Me.ItemID.Text.ToString(), hClient, data, time, quality)
        Catch
        End Try


Public Sub DisplaySync(ByVal itemId As String, ByVal HandleClient As Integer(), ByVal
DataValue As Object(), ByVal TimeStamp As Long(), ByVal Quality As Short())
        Me.listView3.Items.Clear()
        For i As Integer = 0 To count - 1
            Try
                Dim Items As String()
                Dim strValue As String
                Dim t As Type
                t = DataValue(i).GetType()
                Dim enumerable As IEnumerable
                Dim arr As ArrayList
                arr = New ArrayList
                strValue = String.Empty

                If t.IsArray Then

                    enumerable = DataValue(i)
                    If enumerable Is Nothing Then
                    Else

                        For Each element As Object In enumerable
                            arr.Add(element)
                        Next

                    End If

                    strValue = String.Join(", ", Array.ConvertAll(arr.ToArray(), New
Converter(Of Object, String)(AddressOf ObjectToString)))


                End If
```

VB:

```
                If (Quality(i) = 192) Then

                    Items = New String() {HandleClient(i).ToString(), itemId.ToString(),
strValue, DateTime.FromFileTime(TimeStamp(i)).ToString(), "Good"}
                Else
                    Items = New String() {HandleClient(i).ToString(), itemId.ToString(),
strValue, DateTime.FromFileTime(TimeStamp(i)).ToString(), Quality(i).ToString()}
                End If
                Dim I1 As New ListViewItem(Items)
                Me.listView3.Items.Add(I1)
            Catch
            End Try
```

HOW TO INTERACT WITH THE OPC HDA .NET CLIENT TOOLKIT

1. Initialization of the API

After the DLL initialization, the client application shall hold the responsibility for properly initializing the API. Thus, the OPC .Net Client Toolkit exports the API function named OPCHDAManager that performs the basic initialization functions.

2. HDA Servers Auto-discovery

OPC .NET Client Toolkit provides a way to auto-discover all OPC HDA servers available on the network. The following table describes the parameters of the **ListAllHDAServers** function.

```
void ListAllHDAServers(string          host,
                       out ArrayList   lServers)
```

Parameters

In/Out	Parameter	Description
In	**host**	Name of the machine that hosts OPC HDA servers
Out	**lServers**	This parameter will contain the list of located OPC HDA servers

Table 147: Parameters of the ListAllHDAServers Function

3. Server Management

3.1. Connect To Server

a. Connect

This function establishes a connection to the server identified by **progidOPCserver** and located on the machine **strHostName**. The following table describes the parameters of the **Connect** function.

```
int Connect (string       progidOPCserver,
             string       strHostName,
             out int      index,
```

```
                out string      strError);
```

Parameters

In/Out	Parameter	Description
In	**progidOPCserver**	The server progID
In	**strHostName**	The server node name
Out	**index**	If the call succeeds, this parameter will contain a unique identifier for the connection. It should be passed in any further call to the server.
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 148: Parameters of the Connect Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_OK	The operation succeeded.

Table 149: Returned Codes of the Connect Function

b. Connect with Authentication

This function establishes a connection to the server with user authentication. The following table describes the parameters of the **ConnectWithAuthentication** function.

```
int ConnectWithAuthentication(string    progIDOPCserver,
                        string    strHostName,
                        string    strDomain,
                        string    strLogin,
                        string    strPassword,
                        out int     index,
                        out string    strError)
```


customerservice@integrationobjects.com

Parameters

In/Out	Parameter	Description
In	**progIDOPCserver**	The server progID
In	**strHostName**	The server node name
In	**strDomain**	The domain name. In case the server machine is in the workgroup, this parameter can be replaced with the machine name instead of the domain name.
In	**strLogin**	The login of the server's user.
In	**strPassword**	The password of the server's user.
Out	**index**	If the call succeeds, this parameter will contain a unique identifier for the connection. It should be passed in any further call to the server.
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 150: Parameters of the ConnectWithAuthentication Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_OK	The operation succeeded.

Table 151: Returned Codes of the ConnectWithAuthentication Function

3.2. Disconnect from Server

To disconnect from the server, the client may use the method called **disconnect**, providing the Server Handle returned by the method **connect**. The following table describes the parameters of this function.

```
int Disconnect(int    ServerIndex,
               int    BrowserIndex)
```

Parameters

In/Out	Parameter	Description
In	**ServerIndex**	The identifier of the connection
In	**BrowserIndex**	The identifier of the browser

Table 152: Parameters of the Disconnect Function

Return Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_OK	The operation succeeded.

Table 153: Returned Codes of the Disconnect Function

3.3. Subscription to Callbacks Functions

For the asynchronous requests, the user must provide callback functions used by the API to asynchronously inform the user that there is incoming data or to send back the results of requests. According to the OPC HDA specification, the client must provide 9 types of callback function:

1. OnReadComplete
2. OnDataChange
3. OnReadModifiedComplete
4. OnReadAttributeComplete
5. OnUpdateComplete
6. OnReadAnnotation
7. OnInsertAnnotation

8. OnPlayBack
9. OnCancelComplete

Before using these callbacks, they must be activated. This activation is made once for all connections.

4. Browsing

4.1. CreateBrowse

This function creates a new instance of a browser. The following table describes the parameters of this function.

```
void CreateBrowse(int                          intServerIndex,
                  int                          dwCount,
                  int[]                        pdwAttrID,
                  OPCHDA_OPERATORCODES[]       pOperator,
                  object[]                     vFilter,
                  out int[]                    ppErrors,
                  out int                      intBrowserIndex)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwCount**	The number of attributes in the filter
In	**pdwAttrID**	The filter attribute ID
In	**pOperator**	The filter operator
In	**vFilter**	The filter value
Out	**ppErrors**	The returned errors
Out	**intBrowserIndex**	The returned identifier of the browser

Table 154: Parameters of the CreateBrowse Function

The pOperator parameter values

Status	Value
OPCHDA_EQUAL	1

OPCHDA_LESS	2
OPCHDA_LESSEQUAL	3
OPCHDA_GREATER	4
OPCHDA_GREATEREQUAL	5
OPCHDA_NOTEQUAL	6

Table 155 : Values of the pOperator Parameter

4.2. InitBrowse

This function initializes the parameters of the browser. The following table describes the parameters of this function.

```
void initBrowse(int                          intServerIndex,
                int                          intBrowserIndex,
                OPCHDA_BROWSETYPE            type,
                OPCHDA_BROWSEDIRECTION       direction,
                out string[]                 strF,
                out int                      cft)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**intBrowserIndex**	The browser index
In	**type**	The browsing type
In	**direction**	The browsing direction.
Out	**strF**	The returned result list
Out	**cft**	The number of results

Table 156: Parameters of the InitBrowse Function

The type parameter values

Status	Value
OPC_BRANCH	1

customerservice@integrationobjects.com

OPC_LEAF	2
OPC_FLAT	3

Table 157 : Values of the type Parameter

The direction parameter values

Status	Value
OPC_BROWSE_UP	1
OPC_BROWSE_DOWN	2
OPC_BROWSE_TO	3

Table 158 : Values of the type Parameter

4.3. Browse

This function browses the server's items. The following table describes the parameters of this function.

```
void Browse(int             intServerIndex,
            int             intBrowserIndex,
            int             type,
            int             direction,
            string          source,
            out string[]    strF,
            out int         cft)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**BrowserIndex**	The browser index
In	**type**	The browsing type. *Refer to the "Values of the type parameter" table*
In	**direction**	The browsing direction. *Refer to the "Values of the direction parameter" table*
In	**source**	The source node name
Out	**strF**	The returned result

customerservice@integrationobjects.com

Out	**cft**	The number of results

Table 159: Parameters of the Browse Function

4.4. GetItemID

This function provides a way to get fully qualified item identification. The following table describes the parameters of this function.

```
void GetItemID(int          intServerIndex,
               int          intBrowserIndex,
               string       node,
               out string   result)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**BrowserIndex**	The browser index
In	**node**	The node name
Out	**result**	The returned ItemID

Table 160: Parameters of the GetItemID Function

5. HDA Items

5.1. GetItemAttributes

This function returns the items attributes supported by the considered server. The following table describes the parameters of this function.

```
void GetItemAttributes(int                  intServerIndex,
                       out  int             pdwCount,
                       out int[]            ppdwAttrID,
                       out string[]         ppszAttrName,
                       out string[]         ppszAttrDesc,
                       out VarEnum[]        ppvtAttrDataType)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection

Out	**pdwCount**	The number of attributes
Out	**ppdwAttrID**	The attribute ID
Out	**ppszAttrName**	The attribute Name
Out	**ppszAttrDesc**	The attribute Description
Out	**ppvtAttrDataType**	The attribute Data Type

Table 161: Parameters of the GetItemAttributes Function

5.2. GetAggregates

This function returns the aggregates supported by the server. The following table describes the parameters of this function.

```
int GetAggregates(int       intServerIndex,
                  out int       pdwCount,
                  out int[]     ppdwAggrID,
                  out string[]  ppszAggrName,
                  out string[]  ppszAggrDesc,
                 out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
Out	**pdwCount**	The number of aggregates
Out	**ppdwAggrID**	The aggregates IDs
Out	**ppszAggrName**	The aggregates Names
Out	**ppszAggrDesc**	The aggregates Descriptions
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 162: Parameters of the GetAggregates Function

The ppdwAggrID parameter values

Status	Value
OPCHDA_NOAGGREGATE	0
OPCHDA_INTERPOLATIVE	1
OPCHDA_TOTAL	2
OPCHDA_AVERAGE	3
OPCHDA_TIMEAVERAGE	4
OPCHDA_COUNT	5
OPCHDA_STDEV	6
OPCHDA_MINIMUMACTUALTIME	7
OPCHDA_MINIMUM	8
OPCHDA_MAXIMUMACTUALTIME	9
OPCHDA_MAXIMUM	10
OPCHDA_START	11
OPCHDA_END	12
OPCHDA_DELTA	13
OPCHDA_REGSLOPE	14
OPCHDA_REGCONST	15
OPCHDA_REGDEV	16
OPCHDA_VARIANCE	17
OPCHDA_RANGE	18
OPCHDA_DURATIONGOOD	19
OPCHDA_DURATIONBAD	20
OPCHDA_PERCENTGOOD	21

OPCHDA_PERCENTBAD	22
OPCHDA_WORSTQUALITY	23
OPCHDA_ANNOTATIONS	24

Table 163 : Values of the ppdwAggrID Parameter

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_OK	The operation succeeded.

Table 164: Returned Codes of the GetAggregates Function

5.3. GetHistorianStatus

This function returns information about the current server status. The following table describes the parameters of this function.

```
void GetHistorianStatus(int                          intServerIndex,
                        out OPCHDA_SERVERSTATUS      pwStatus,
                        out long                     pftCurrentTime,
                        out long                     pftStartTime,
                        out short                    pwMajorVersion,
                        out short                    pwMinorVersion,
                        out short                    pwBuildNumber,
                        out int                     pdwMaxReturnValues,
                        out string                   ppszStatusString,
                        out string                   ppszVendorInfo)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
Out	**pwStatus**	The current status of the historian

Out	**pftCurrentTime**	The current time at the historian location
Out	**pftStartTime**	The time when the historian was last started
Out	**pwMajorVersion**	The major version of the historian
Out	**pwMinorVersion**	The minor version of the historian
Out	**pwBuildNumber**	The build number of the historian
Out	**pdwMaxReturnValues**	The maximum number of values that can be returned by the server on a per item basis. A value of 0 indicates that the server forces no limit on the number of values it can return.
Out	**ppszStatusString**	A string explaining the historian status when the pwStatus value is OPCHDA_INDETERMINATE
Out	**ppszVendorInfo**	A vendor-specific informational string

Table 165: Parameters of the GetHistorianStatus Function

The pwStatus parameter values

Status	Value
OPCHDA_UP	1
OPCHDA_DOWN	2
OPCHDA_INDETERMINATE	3

Table 166 : Values of the pwStatus Parameter

5.4. GetItemHandles

Given an ItemID and a client handle, this function returns the server handle for this item. The returned server handle must be used in all requests to read or update history. The following table describes the parameters of this function.

```
int GetItemHandles(int                intServerIndex,
                   int                pdwCount,
                   string[]           pszItemID,
                   int[]              phClient,
```

```
                                out int[]           pphServer,
                                out int[]           ppErrors,
                                out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**pdwCount**	The number of items to be handled
In	**pszItemID**	The string that uniquely identifies the OPC HDA item
In	**phClient**	The handle of the client to be associated with the item
Out	**pphServer**	The returned handle of the server used to refer to this item
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 167: Parameters of the GetItemHandles Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
S_OK	The operation succeeded.

Table 168: Returned Codes of the GetItemHandles Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_BADRIGHTS	The item is not readable
OPC_E_INVALIDITEMID	The ItemId was specified incorrectly
E_FAIL	The Read failed for this item
OPC_E_UNKNOWNITEMID	The item is no longer available in the server address space.

Table 169: Error Array Codes of the GetItemHandles Function

5.5. ReleaseItemHandles

This function releases associations between the server handle and the client handle for a specific HDA item. The following table describes the parameters of this function.

```
int  ReleaseItemHandles(int       intServerIndex,
                        int           dwCount,
                        int[]         phServer,
                        out int[]     ppErrors,
                       out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwCount**	The number of item handles to be released
In	**phServer**	The handle of the server used to refer to this item
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 170: Parameters of the ReleaseItemHandles Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
S_OK	The operation succeeded.

Table 171: Returned Codes of the ReleaseItemHandles Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The server handle does not exist.
E_FAIL	The operation failed for this item

Table 172: Error Array Codes of the ReleaseItemHandles Function

5.6. ValidateItemIDs

This function validates that the server knows the specific HDA ItemID. The following table describes the parameters of this function.

```
int ValidateItemIDs( int          intServerIndex,
                     int          dwCount,
                     string[]     pszItemID,
                     out int[]    ppErrors,
                     out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwCount**	The number of items to be validated

In	**pszItemID**	The string that uniquely identifies the OPC HDA item
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 173: Parameters of the ValidateItemIDs Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
S_OK	The operation succeeded.

Table 174: Returned Codes of the ValidateItemIDs Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_UNKNOWNITEMID	The item does not exist in the server address space
OPC_E_INVALIDITEMID	The item ID specification is syntactically incorrect.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
E_FAIL	The operation failed for this item

Table 175: Error Array Codes of the ValidateItemIDs Function

6. Synchronous Read Methods

6.1. SyncReadRaw

This function synchronously reads the values, qualities, and timestamps from the historian for the specified time domain for one or more items. The following table describes the parameters of this function.

```
int SyncReadRaw(int                     intServerIndex,
                OPCHDA_TIME             htStartTime,
                OPCHDA_TIME             htEndTime,
                int                     dwNumValues,
                bool                    bBounds,
                int                     dwNumItems,
                int[]                   phServer,
                out OPCHDA_ITEM[]       ppItemValues,
                out int[]               ppErrors,
                out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**htStartTime**	The start of the history time period to be read
In	**htEndTime**	The end of the history time period to be read
In	**dwNumValues**	The maximum number of values returned for any item over the time range. If only one time is specified, the time range must extend to return this number of values.
In	**bBounds**	• TRUE if bounding values should be returned • FALSE otherwise
In	**dwNumItems**	The number of items to be read
In	**phServer**	The server item handle for the item to be read
Out	**ppItemValues**	The returned structure of item's values
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned

		error message. Otherwise, it will contain an empty string.

Table 176: Parameters of the SyncReadRaw Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
OPC_E_MAXEXCEEDED	The maximum number of values requested (dwNumItems) is greater than the server limit of maximum values returned.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 177: Returned Codes of the SyncReadRaw Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_S_NODATA	No data was found in the specified time range.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
OPC_S_MOREDATA	More data is available in the time range beyond the number of values requested.
E_FAIL	The operation failed for this item

Table 178: Error Array Codes of the SyncReadRaw Function

6.2. SyncReadProcessed

This function synchronously computes aggregate values, qualities, and timestamps from data in the history database for the specified time domain for one or more items. The following table describes the parameters of this function.

```
int SyncReadProcessed (int                      intServerIndex,
                       OPCHDA_TIME              htStartTime,
                       OPCHDA_TIME              htEndTime,
                       long                     ftResampleInterval,
                       int                      dwNumItems,
                       int[]                    phServer,
                       int[]                    haAggregate,
                       out OPCHDA_ITEM[]        ppItemValues,
                       out int[]                ppErrors,
                       out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**htStartTime**	The start of the history time period to be read
In	**htEndTime**	The end of the history time period to be read
In	**ftResampleInterval**	Interval between returned values
In	**dwNumItems**	The number of items to be read
In	**phServer**	The server item handle for the item to be read
In	**haAggregate**	The calculation to be performed on the raw data to create the values to be returned.
Out	**ppItemValues**	The returned structure of item's values
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 179: Parameters of the SyncReadProcessed Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
OPC_E_MAXEXCEEDED	The maximum number of values requested (dwNumItems) is greater than the server limit of maximum values returned.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 180: Returned Codes of the SyncReadProcessed Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_S_NODATA	No data was found in the specified time range.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
OPC_S_EXTRADATA	There is more data available than was returned.
OPC_E_NOT_AVAIL	The requested aggregate is not available for the specified item.
E_FAIL	The operation failed for this item

Table 181: Error Array Codes of the SyncReadProcessed Function

6.3. SyncReadAtTime

This function synchronously reads the values and qualities from the history database for the specified timestamps for one or more items. The following table describes the parameters of this function.

```
int SyncReadAtTime (int                 intServerIndex,
                    int                 dwNumTimeStamps,
                    long[]              ftTimeStamps,
                    int                 dwNumItems,
                    int[]               phServer,
                    out OPCHDA_ITEM[]   ppItemValues,
                    out int[]           ppErrors,
                    out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwNumTimeStamps**	The number of timestamps specified
In	**ftTimeStamps**	Array of timestamps for the requested data
In	**dwNumItems**	The number of items to be read
In	**phServer**	The server item handle for the item to be read
Out	**ppItemValues**	The returned structure of item's values
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 182: Parameters of the SyncReadAtTime Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.

S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
E_NOTIMPL	The server does not support this function.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 183: Returned Codes of the SyncReadAtTime Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_S_NODATA	No data was found in the specified time range.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
E_FAIL	The operation failed for this item

Table 184: Error Array Codes of the SyncReadAtTime Function

6.4. SyncReadModified

This function synchronously reads the values, qualities, and timestamps of the modification from the history database for the specified time domain for one or more items. The following table describes the parameters of this function.

```
int SyncReadModified (int                              intServerIndex,
                      OPCHDA_TIME                      htStartTime,
                      OPCHDA_TIME                      htEndTime,
                      int                              dwNumValues,
                      int                              dwNumItems,
                      int[]                            phServer,
                      out OPCHDA_MODIFIEDITEM[]        ppItemValues,
                      out int[]                        ppErrors,
                      out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**htStartTime**	The beginning of the history period to be read
In	**htEndTime**	The end of the history period to be read
In	**dwNumValues**	The maximum number of values returned for any item over the time range. If only one time is specified, the time range must extend to return this number of values.
In	**dwNumItems**	The number of items to be read
In	**phServer**	The server item handle for the item to be read
Out	**ppItemValues**	The returned structure of item's values
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 185: Parameters of the SyncReadModified Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
E_NOTIMPL	The server does not support this function.
E_INVALIDARG	An invalid parameter was passed.

S_OK	The operation succeeded.

Table 186: Returned Codes of the SyncReadModified Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_S_NODATA	No data was found in the specified time range.
OPC_S_MOREDATA	More data is available in the time range beyond the number of values requested.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
E_INVALIDARG	An invalid parameter was passed.
E_FAIL	The operation failed for this item

Table 187: Error Array Codes of the SyncReadModified Function

6.5. SyncReadAttribute

This function synchronously reads the attribute values and timestamps from the history database for the specified time domain for an item. The following table describes the parameters of this function.

```
int SyncReadAttribute(int                          intServerIndex,
                      OPCHDA_TIME                  htStartTime,
                      OPCHDA_TIME                  htEndTime,
                      int                          hServer,
                      int                          dwNumAttributes,
                      int[]                        pdwAttributeIDs,
                      out OPCHDA_ATTRIBUTE[]       ppItemValues,
                      out int[]                    ppErrors,
                      out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection

In	**htStartTime**	The beginning of the history period to be read
In	**htEndTime**	The end of the history period to be read
In	**hServer**	The server item handle for the item to be read
In	**dwNumAttributes**	The number of attributes to be read
In	**pdwAttributeIDs**	The attribute IDs to be read
Out	**ppItemValues**	The returned structure of item's values
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 188: Parameters of the SyncReadAttribute Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
OPC_E_INVALIDHANDLE	The handle is invalid.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 189: Returned Codes of the SyncReadAttribute Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.

customerservice@integrationobjects.com

OPC_E_INVALIDATTRID	The attribute ID is not valid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
OPC_S_CURRENTVALUE	No history available for attribute.
E_FAIL	The operation failed.

Table 190: Error Array Codes of the SyncReadAttribute Function

7. Synchronous Update Methods

7.1. SyncQueryCapabilities

This function specifies the synchronous update methods supported by the specified server. The following table describes the parameters of this function.

```
void SyncQueryCapabilities(
                int                                 intServerIndex,
                out OPCHDA_UPDATECAPABILITIES       pCapabilities)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
Out	**pCapabilities**	The synchronous update methods supported by the server

Table 191: Parameters of the SyncQueryCapabilities Function

The pCapabilities parameter values

Status	Value
OPCHDA_INSERTCAP	0x01
OPCHDA_REPLACECAP	0x02
OPCHDA_INSERTREPLACECAP	0x04
OPCHDA_DELETERAWCAP	0x08
OPCHDA_DELETEATTIMECAP	0x10

Table 192 : Values of the pCapabilities Parameter

7.2. SyncInsert

This function synchronously inserts values and qualities into the history database at the specified timestamps for one or more items. The following table describes the parameters of this function.

```
int SyncInsert(int              intServerIndex,
               int              dwNumItems,
               int[]            phServer,
               long[]           ftTimeStamps,
               object[]         vDataValues,
               int[]            pdwQualities,
               out int[]        ppErrors,
               out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwNumItems**	The number of items to be updated
In	**phServer**	The server item handle for the item to be inserted
In	**ftTimeStamps**	The item timestamp for the value to insert
In	**vDataValues**	The item value to insert
In	**pdwQualities**	The quality flag for the value to insert
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 193: Parameters of the SyncInsert Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid

E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
E_NOTIMPL	The server does not support this function.
S_OK	The operation succeeded.

Table 194: Returned Codes of the SyncInsert Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
E_INVALIDARG	An invalid parameter was passed.
OPC_E_DATAEXISTS	Unable to insert – data already present.
E_FAIL	The operation failed.

Table 195: Error Array Codes of the SyncInsert Function

7.3. SyncInsertReplace

This function synchronously inserts or replaces values and qualities into the history database at the specified timestamps for one or more items. The following table describes the parameters of this function.

```
int  SyncInsertReplace(int             intServerIndex,
                       int             dwNumItems,
                       int[]           phServer,
                       long[]          ftTimeStamps,
                       object[]        vDataValues,
                       int[]           pdwQualities,
                       out int[]       ppErrors,
                       out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwNumItems**	The number of items to be updated
In	**phServer**	The server item handle for the item to be updated
In	**ftTimeStamps**	The item timestamp for the value to insert
In	**vDataValues**	The item value to insert
In	**pdwQualities**	The quality flag for the value to insert
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 196: Parameters of the SyncInsertReplace Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
E_NOTIMPL	The server does not support this function.
S_OK	The operation succeeded.

Table 197: Returned Codes of the SyncInsertReplace Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
E_INVALIDARG	An invalid parameter was passed.
OPC_S_INSERTED	The requested insert occurred.
OPC_S_REPLACED	The requested replace occurred.
E_FAIL	The operation failed.

Table 198: Error Array Codes of the SyncInsertReplace Function

7.4. SyncReplace

This function synchronously replaces values and qualities into the history database at the specified timestamps for one or more items. The following table describes the parameters of this function.

```
int SyncReplace( int              intServerIndex,
                 int              dwNumItems,
                 int[]            phServer,
                 long[]           ftTimeStamps,
                 object[]         vDataValues,
                 int[]            pdwQualities,
                 out int[]        ppErrors,
                 out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwNumItems**	The number of items to be replaced
In	**phServer**	The server item handle for the item to be updated

customerservice@integrationobjects.com

In	**ftTimeStamps**	The item timestamp for the value to insert
In	**vDataValues**	The item value to insert
In	**pdwQualities**	The quality flag for the value to insert
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 199: Parameters of the SyncReplace Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
E_NOTIMPL	The server does not support this function.
S_OK	The operation succeeded.

Table 200: Returned Codes of the SyncReplace Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.

E_INVALIDARG	An invalid parameter was passed.
OPC_E_NODATAEXISTS	Unable to replace – no data exists.
E_FAIL	The operation failed.

Table 201: Error Array Codes of the SyncReplace Function

7.5. SyncDeleteRaw

This function synchronously deletes the values, qualities, and timestamps from the history database for the specified time domain for one or more items. The following table describes the parameters of this function.

```
int  SyncDeleteRaw(int                  intServerIndex,
                   OPCHDA_TIME          htStartTime,
                   OPCHDA_TIME          htEndTime,
                   int                  dwNumItems,
                   int[]                phServer,
                   out int[]            ppErrors,
                  out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**htStartTime**	The beginning of the history period
In	**htEndTime**	The end of the history period
In	**dwNumItems**	The number of items to be deleted
In	**phServer**	The server item handle for the item to be updated
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 202: Parameters of the SyncDeleteRaw Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
E_NOTIMPL	The server does not support this function.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 203: Returned Codes of the SyncDeleteRaw Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
OPC_S_NODATA	No values to delete for the item in the specified time range.
E_FAIL	The operation failed.

Table 204: Error Array Codes of the SyncDeleteRaw Function

7.6. SyncDeleteAtTime

This function synchronously deletes the values and qualities in the history database for the specified timestamps for one or more items. The following table describes the parameters of this function.

```
int   SyncDeleteAtTime(int              intServerIndex,
                       int              dwNumItems,
                       int[]            phServer,
```

```
long[]           ftTimeStamps,
out int[]        ppErrors,
out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**phServer**	The server item handle for the item values to be deleted
In	**dwNumItems**	The number of items to be deleted
In	**ftTimeStamps**	The timestamps for the data to be deleted
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 205: Parameters of the SyncDeleteAtTime Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
E_NOTIMPL	The server does not support this function.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 206: Returned Codes of the SyncDeleteAtTime Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
OPC_S_NODATA	No values to delete for the item in the specified time range.
E_FAIL	The operation failed.

Table 207: Error Array Codes of the SyncDeleteAtTime Function

8. Synchronous Annotations Methods

8.1. SAQueryCapabilities

This function specifies the synchronous annotations methods supported by the specified server. The following table describes the parameters of this function.

```
void SAQueryCapabilities(
          int                                  intServerIndex,
          out OPCHDA_ANNOTATIONCAPABILITIES    pCapabilities)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
Out	**pCapabilities**	The synchronous annotations methods supported by the server

Table 208: Parameters of the SAQueryCapabilities Function

The pCapabilities parameter values

Status	Value
OPCHDA_READANNOTATIONCAP	0x01

customerservice@integrationobjects.com

OPCHDA_INSERTANNOTATIONCAP	0x02

Table 209 : Values of the pCapabilities Parameter

8.2. SARead

This function synchronously reads the annotations from the history database in the specified time domain for one or more item. The following table describes the parameters of this function.

```
int SARead(int                              intServerIndex,
           OPCHDA_TIME                      htStartTime,
           OPCHDA_TIME                      htEndTime,
           int                              dwNumItems,
           int[]                            phServer,
           out OPCHDA_ANNOTATION[]          ppAnnotationValues,
           out int[]                        ppErrors,
           out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**htStartTime**	The beginning of the history period
In	**htEndTime**	The end of the history period
In	**dwNumItems**	The number of items to be read
In	**phServer**	The server item handle for the annotation values to be read
Out	**ppAnnotationValues**	The returned structure of annotation's values
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 210: Parameters of the SARead Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 211: Returned Codes of the SARead Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
OPC_S_NODATA	No values to delete for the item in the specified time range.
E_FAIL	The operation failed.

Table 212: Error Array Codes of the SARead Function

8.3. SAInsert

This function synchronously inserts the annotations into the history database in the specified time domain for one or more item. The following table describes the parameters of this function.

```
int SAInsert(int                    intServerIndex,
             int                    dwNumItems,
             int[]                  phServer,
             long[]                 ftTimeStamps,
             OPCHDA_ANNOTATION[]    ppAnnotationValues,
```

```
                out int[]                   ppErrors,
                 out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwNumItems**	The number of the annotations to be inserted
In	**phServer**	The server item handle for the annotations values to be insert
In	**ftTimeStamps**	Array of timestamps for the requested data
In	**ppAnnotationValues**	The structure of annotation's values
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 213: Parameters of the SAInsert Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 214: Returned Codes of the SAInsert Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
E_FAIL	The operation failed.

Table 215: Error Array Codes of the SAInsert Function

9. Asynchronous Annotations Methods

9.1. AAQueryCapabilities

This function specifies the asynchronous annotations methods supported by the specified server. The following table describes the parameters of this function.

```
void AAQueryCapabilities(
                int   intServerIndex,
                 out OPCHDA_ANNOTATIONCAPABILITIES pCapabilities)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
Out	**pCapabilities**	The asynchronous annotations methods supported by the server.

Table 216: Parameters of the AAQueryCapabilities Function

9.2. AARead

This function synchronously reads the annotations from the history database in the specified time domain for one or more item. The following table describes the parameters of this function.

```
int AARead(int                  intServerIndex,
           int                  dwTransactionID,
           OPCHDA_TIME          htStartTime,
           OPCHDA_TIME          htEndTime,
```

```
int                 dwNumItems,
int[]               phServer,
out int             pdwCancelID,
out int[]           ppErrors,
out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwTransactionID**	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
In	**htStartTime**	The beginning of the history period
In	**htEndTime**	The end of the history period
In	**dwNumItems**	The number of items to be read
In	**phServer**	The server item handle for the annotations values to be read
Out	**pdwCancelID**	The server-generated cancelID
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 217: Parameters of the AARead Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.

E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 218: Returned Codes of the AARead Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
E_INVALIDARG	An Invalid parameter was passed.

Table 219: Error Array Codes of the AARead Function

9.3. AAInsert

This function asynchronously inserts the annotations into the history database in the specified time domain for one or more item. The following table describes the parameters of this function.

```
int AAInsert(int                        intServerIndex,
             int                        dwTransactionID,
             int                        dwNumItems,
             int[]                      phServer,
             long[]                     ftTimeStamps,
             OPCHDA_ANNOTATION[]        pAnnotationValues,
             out int                    pdwCancelID,
             out int[]                  ppErrors,
            out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwTransactionID**	An identifier created by the client and passed to the server in this call. The server shall

		return this identifier along with the results of this call.
In	**dwNumItems**	The number of the annotations to be inserted
In	**phServer**	The server item handle for the annotations values to be inserted
In	**ftTimeStamps**	Array of timestamps for the requested data
In	**ppAnnotationValues**	The structure of annotation's values
Out	**pdwCancelID**	The server-generated cancelID
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 220: Parameters of the AAInsert Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 221: Returned Codes of the AAInsert Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.

OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
E_INVALIDARG	An Invalid parameter was passed.

Table 222: Error Array Codes of the AAInsert Function

9.4. AACancel

This function cancels the outstanding operation. The following table describes the parameters of this function.

```
void AACancel(int           intServerIndex,
              int           dwCancelID)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwCancelID**	The server-generated cancelID which was returned from the original method call

Table 223: Parameters of the AACancel Function

10. Asynchronous Read Methods

10.1. AsyncReadRaw

This function asynchronously reads the values and qualities from the history database for the specified timestamps for one or more items. The following table describes the parameters of this function.

```
int AsyncReadRaw(int              intServerIndex,
                 int              dwTransactionID,
                 OPCHDA_TIME      htStartTime,
                 OPCHDA_TIME      htEndTime,
                 int              dwNumValues,
                 bool             bBounds,
                 int              dwNumItems,
                 int[]            phServer,
                 out int          pdwCancelID,
                 out int[]        ppErrors,
               out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwTransactionID**	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
In	**htStartTime**	The beginning of the history period
In	**htEndTime**	The end of the history period
In	**dwNumValues**	The maximum number of values returned for any item over the time range. If only one time is specified, the time range must extend to return this number of values.
In	**bBounds**	• TRUE if bounding values should be returned • FALSE otherwise
In	**dwNumItems**	The number of items to be read
In	**phServer**	The server item handle for the item to be read
Out	**pdwCancelID**	The server-generated cancelID
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 224: Parameters of the AsyncReadRaw Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.

S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
OPC_E_MAXEXCEEDED	The maximum number of values requested (dwNumItems) is greater than the server limit of maximum values returned.
E_NOTIMPL	The server does not support this function.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 225: Returned Codes of the AsyncReadRaw Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.

Table 226: Error Array Codes of the AsyncReadRaw Function

10.2. AsyncAdviseRaw

This function asynchronously reads the values, qualities, and timestamps from the history database from the specified start time at the update interval for one or more items.

```
int AsyncAdviseRaw(int                  intServerIndex,
                   int                  dwTransactionID,
                   OPCHDA_TIME          htStartTime,
                   long                 ftUpdateInterval,
                   int                  dwNumItems,
                   int[]                phServer,
                   out int              pdwCancelID,
                   out int[]            ppErrors,
                  out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwTransactionID**	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
In	**htStartTime**	The beginning of the history period
In	**ftUpdateInterval**	Update interval to send new data
In	**dwNumItems**	The number of items to be read
In	**phServer**	The server item handle for the item to be read
Out	**pdwCancelID**	The server-generated cancelID
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 227: Parameters of the AsyncAdviseRaw Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
OPC_S_UNSUPPORTEDRATE	The requested update interval is not supported by the server.
E_NOTIMPL	The server does not support this function.
E_INVALIDARG	An invalid parameter was passed.

S_OK	The operation succeeded.

Table 228: Returned Codes of the AsyncAdviseRaw Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
E_INVALIDARG	An Invalid parameter was passed

Table 229: Error Array Codes of the AsyncAdviseRaw Function

10.3. AsyncReadProcessed

This function asynchronously computes aggregate values, qualities, and timestamps from data in the history database for the specified time domain for one or more items. The following table describes the parameters of this function.

```
int AsyncReadProcessed(int             intServerIndex,
                       int             dwTransactionID,
                       OPCHDA_TIME     htStartTime,
                       OPCHDA_TIME     htEndTime,
                       long            ftResampleInterval,
                       int             dwNumItems,
                       int[]           phServer,
                       int[]           haAggregate,
                       out int         pdwCancelID,
                       out int[]       ppErrors,
                       out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection

In	**dwTransactionID**	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
In	**htStartTime**	The beginning of the history period
In	**htEndTime**	The end of the history period
In	**ftResampleInterval**	Interval between returned values
In	**dwNumItems**	The number of items to be read
In	**phServer**	The server item handle for the item to be read
In	**haAggregate**	The calculation to be performed on the raw data to create the values to be returned.
Out	**pdwCancelID**	The server-generated cancelID
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 230: Parameters of the AsyncReadProcessed Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
OPC_E_MAXEXCEEDED	The maximum number of values returnable by the server was exceeded.
E_NOTIMPL	The server does not support this function.
E_INVALIDARG	An invalid parameter was passed.

S_OK	The operation succeeded.

Table 231: Returned Codes of the AsyncReadProcessed Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.

Table 232: Error Array Codes of the AsyncReadProcessed Function

10.4. AsyncAdviseProcessed

This function asynchronously computes aggregate values, qualities, and timestamps from data in the history from the specified start time at the update interval for one or more items. The following table describes the parameters of this function.

```
int AsyncAdviseProcessed(int            intServerIndex,
                         int            dwTransactionID,
                         OPCHDA_TIME    htStartTime,
                         long           ftUpdateInterval,
                         int            dwNumItems,
                         int[]          phServer,
                         int[]          haAggregate,
                         int            dwNumIntervals,
                         out int        pdwCancelID,
                         out int[]      ppErrors,
                        out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwTransactionID**	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.

customerservice@integrationobjects.com

In	**htStartTime**	The beginning of the history period
In	**ftUpdateInterval**	Update interval to send new data
In	**dwNumItems**	The number of items to be read
In	**phServer**	The server item handle for the item to be read
In	**haAggregate**	The calculation to be performed on the raw data to create the values to be returned.
In	**dwNumIntervals**	Number of resample intervals between updates.
Out	**pdwCancelID**	The server-generated cancelID
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 233: Parameters of the AsyncAdviseProcessed Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
OPC_S_UNSUPPORTEDRATE	The requested resample interval is not supported by the server.
E_NOTIMPL	The server does not support this function.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 234: Returned Codes of the AsyncAdviseProcessed Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.

Table 235: Error Array Codes of the AsyncAdviseProcessed Function

10.5. AsyncReadAtTime

This function asynchronously reads the values and qualities from the history database for the specified timestamps for one or more items. The following table describes the parameters of this function.

```
int AsyncReadAtTime(int          intServerIndex,
                    int          dwTransactionID,
                    int          dwNumTimeStamps,
                    long[]       ftTimeStamps,
                    int          dwNumItems,
                    int[]        phServer,
                    out int      pdwCancelID,
                    out int[]    ppErrors,
                    out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwTransactionID**	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
In	**dwNumTimeStamps**	The number of timestamps specified
In	**ftTimeStamps**	Array of timestamps for the requested data
In	**dwNumItems**	The number of items to be read

In	**phServer**	The server item handle for the item to be read
Out	**pdwCancelID**	The server-generated cancelID
Out	**ppErrors**	Returned errors
Out	**StrError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 236: Parameters of the AsyncReadAtTime Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information..
E_NOTIMPL	The server does not support this function.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 237: Returned Codes of the AsyncReadAtTime Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.

Table 238: Error Array Codes of the AsyncReadAtTime Function

10.6. AsyncReadModified

This function asynchronously reads the values, qualities and timestamps of the modification from the history database for the specified time domain for one or more items. The following table describes the parameters of this function.

```
int AsyncReadModified(int              intServerIndex,
                      int              dwTransactionID,
                      OPCHDA_TIME      htStartTime,
                      OPCHDA_TIME      htEndTime,
                      int              dwNumValues,
                      int              dwNumItems,
                      int[]            phServer,
                      out int          pdwCancelID,
                      out int[]        ppErrors,
                     out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwTransactionID**	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
In	**htStartTime**	The beginning of the history period
In	**htEndTime**	The end of the history period
In	**dwNumValues**	The maximum number of values returned for any item over the time range. If only one time is specified, the time range must extend to return this number of values.
In	**dwNumItems**	The number of items to be read
In	**phServer**	The server item handle for the item to be read
Out	**pdwCancelID**	The server-generated cancelID
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 239: Parameters of the AsyncReadModified Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
E_NOTIMPL	The server does not support this function.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 240: Returned Codes of the AsyncReadModified Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.

Table 241: Error Array Codes of the AsyncReadModified Function

10.7. AsyncReadAttribute

This function asynchronously reads the attribute values and timestamps from the history database for the specified time domain for an item. The following table describes the parameters of this function.

```
int AsyncReadAttribute(int             intServerIndex,
                       int             dwTransactionID,
                       OPCHDA_TIME     htStartTime,
                       OPCHDA_TIME     htEndTime,
                       int             phServer,
```

```
int             dwNumAttributes,
int[]           dwAttributeIDs,
out int         pdwCancelID,
out int[]       ppErrors,
out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwTransactionID**	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
In	**htStartTime**	The beginning of the history period
In	**htEndTime**	The end of the history period
In	**phServer**	The server item handle for the item to be read
In	**dwNumAttributes**	The number of attributes to be read
In	**dwAttributeIDs**	The attribute IDs to be read
Out	**pdwCancelID**	The server-generated cancelID
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 242: Parameters of the AsyncReadAttribute Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.

S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
OPC_E_INVALIDHANDLE	The handle is invalid.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 243: Returned Codes of the AsyncReadAttribute Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDATTRID	The attribute ID is not valid.
E_FAIL	The attribute read was unsuccessful.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
OPC_S_CURRENTVALUE	No history available for attribute.

Table 244: Error Array Codes of the AsyncReadAttribute Function

10.8. AsyncCancel

This function cancels the outstanding synchronous update operations. The following table describes the parameters of this function.

```
void AsyncCancel(int        intServerIndex,
                 int        dwCancelID)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection

In	**dwCancelID**	The server-generated cancelID which was returned from the original method call

Table 245: Parameters of the AsyncCancel Function

11. Asynchronous Update Methods

11.1. AsyncQueryCapabilities

This function specifies the asynchronous update methods supported by the specified server. The following table describes the parameters of this function.

```
void AsynQueryCapabilities(
                int                                 intServerIndex,
                out   OPCHDA_UPDATECAPABILITIES     pCapabilities)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
Out	**pCapabilities**	The asynchronous update methods supported by the server.

Table 246: Parameters of the AsyncQueryCapabilities Function

11.2. AsyncUpdateInsert

This function asynchronously inserts values and qualities into the history database at the specified timestamps for one or more items. The following table describes the parameters of this function.

```
int AsyncUpdateInsert(int           intServerIndex,
                      int           dwTransactionID,
                      int           dwNumItems,
                      int[]         phServer,
                      long[]        ftTimeStamps,
                      object[]      vDataValues,
                      int[]         pdwQualities,
                      out int       pdwCancelID,
                      out int[]     ppErrors,
                      out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwTransactionID**	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
In	**dwNumItems**	The number of items to be updated
In	**phServer**	The server item handle for the item to be updated
In	**ftTimeStamps**	The item timestamp for the value to insert
In	**vDataValues**	The item value to insert
In	**pdwQualities**	The quality flag for the value to insert
Out	**pdwCancelID**	The server-generated cancelID
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 247: Parameters of the AsyncUpdateInsert Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
E_NOTIMPL	The server does not support this function.

E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 248: Returned Codes of the AsyncUpdateInsert Function

Error Array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.

Table 249: Error Array Codes of the AsyncUpdateInsert Function

11.3. AsyncUpdateReplace

This function asynchronously replaces values and qualities into the history database at the specified timestamps for one or more items. The following table describes the parameters of this function.

```
int AsyncUpdateReplace(int             intServerIndex,
                       int             dwTransactionID,
                       int             dwNumItems,
                       int[]           phServer,
                       long[]          ftTimeStamps,
                       object[]        vDataValues,
                       int[]           pdwQualities,
                       out int         pdwCancelID,
                       out int[]       ppErrors,
                      out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwTransactionID**	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.

In	**dwNumItems**	The number of items to be updated
In	**phServer**	The server item handle for the item to be updated
In	**ftTimeStamps**	The item timestamp for the value to replace
In	**vDataValues**	The item value to replace
In	**pdwQualities**	The quality flag for the value to replace
Out	**pdwCancelID**	The server-generated cancelID
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 250: Parameters of the AsyncUpdateReplace Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
E_NOTIMPL	The server does not support this function.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 251: Returned Codes of the AsyncUpdateReplace Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.

OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.

Table 252: Error Array Codes of the AsyncUpdateReplace Function

11.4. AsyncUpdateInsertReplace

This function synchronously inserts or replaces values and qualities into the history database at the specified timestamps for one or more items. The following table describes the parameters of this function.

```
int AsyncUpdateInsertReplace(int        intServerIndex,
                             int        dwTransactionID,
                             int        dwNumItems,
                             int[]      phServer,
                             long[]     ftTimeStamps,
                             object[]   vDataValues,
                             int[]      pdwQualities,
                             out int    pdwCancelID,
                             out int[]  ppErrors,
                             out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwTransactionID**	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
In	**dwNumItems**	The number of items to be updated
In	**phServer**	The server item handle for the item to be updated
In	**ftTimeStamps**	The item timestamp for the value to insert
In	**vDataValues**	The item value to insert
In	**pdwQualities**	The quality flag for the value to insert
Out	**pdwCancelID**	The server-generated cancelID

customerservice@integrationobjects.com

Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 253: Parameters of the AsyncUpdateInsertReplace Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
E_NOTIMPL	The server does not support this function.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 254: Returned Codes of the AsyncUpdateInsertReplace Function

Error Array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.

Table 255: Error Array Codes of the AsyncUpdateInsertReplace Function

11.5. AsyncUpdateDeleteRaw

This function asynchronously deletes the values, qualities, and timestamps from the history database for the specified time domain for one or more items. The following table describes the parameters of this function.

```
int AsyncUpdateDeleteRaw(int            intServerIndex,
                         int            dwTransactionID,
                         OPCHDA_TIME    htStartTime,
                         OPCHDA_TIME    htEndTime,
                         int            dwNumItems,
                         int[]          phServer,
                         out int        pdwCancelID,
                         out int[]      ppErrors,
                         out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwTransactionID**	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
In	**htStartTime**	The beginning of the history period
In	**htEndTime**	The end of the history period
In	**dwNumItems**	The number of items to be updated
In	**phServer**	The server item handle for the item to be updated
Out	**pdwCancelID**	The server-generated cancelID
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 256: Parameters of the AsyncUpdateDeleteRaw Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
E_NOTIMPL	The server does not support this function.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 257: Returned Codes of the AsyncUpdateDeleteRaw Function

Error Array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.

Table 258: Error Array Codes of the AsyncUpdateDeleteRaw Function

11.6. AsyncUpdateDeleteAtTime

This function asynchronously deletes the values and qualities in the history database for the specified timestamps for one or more items. The following table describes the parameters of this function.

```
int AsyncUpdateDeleteAtTime(int         intServerIndex,
                            int         dwTransactionID,
                            int         dwNumItems,
                            int[]       phServer,
                            long[]      ftTimeStamps,
                            out int     pdwCancelID,
```

```
                                    out int[]  ppErrors,
                                    out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwTransactionID**	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
In	**dwNumItems**	The number of items to be updated
In	**phServer**	The server item handle for the item to be updated
In	**ftTimeStamps**	Array of timestamps for the requested data
Out	**pdwCancelID**	The server-generated cancelID
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 259: Parameters of the AsyncUpdateDeleteAtTime Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
E_NOTIMPL	The server does not support this function.
E_INVALIDARG	An invalid parameter was passed.

S_OK	The operation succeeded.

Table 260: Returned Codes of the AsyncUpdateDeleteAtTime Function

Error Array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.

Table 261: Error Array Codes of the AsyncUpdateDeleteAtTime Function

11.7. AsyncUpdateCancel

This function cancels the outstanding asynchronous update operations. The following table describes the parameters of this function.

```
void AsyncUpdateCancel(int        intServerIndex,
                       int        dwCancelID)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwCancelID**	The server-generated cancelID that was returned from the original method call

Table 262: Parameters of the AsyncUpdateCancel Function

customerservice@integrationobjects.com

12. Asynchronous Playback Methods

12.1. ReadRawWithUpdate

This function initially retrieves data from the start time to the end time. After the initial response, it periodically (every ftUpdateInterval) responds with an ftUpdateDuration amount of data. The time of the last value returned in the initial response is used as the start time for the first update. After that, the time of the last value returned in an update is used as the start time for the next update. The following table describes the parameters of this function.

```
int ReadRawWithUpdate(int              intServerIndex,
                      int              dwTransactionID,
                      OPCHDA_TIME      htStartTime,
                      OPCHDA_TIME      htEndTime,
                      int              dwNumValues,
                      long             ftUpdateDuration,
                      long             ftUpdateInterval,
                      int              dwNumItems,
                      int[]            phServer,
                      out int          pdwCancelID,
                      out int[]        ppErrors,
                      out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwTransactionID**	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
In	**htStartTime**	The beginning of the history period
In	**htEndTime**	The end of the history period
In	**dwNumValues**	The maximum number of values returned for any item over the time range. If only one time is specified, the time range must extend to return this number of values.
In	**ftUpdateDuration**	The amount of time the update covers in.

In	**ftUpdateInterval**	Update interval to send new data
In	**dwNumItems**	The number of items to be read
In	**phServer**	The server item handle for the item to be updated
Out	**pdwCancelID**	The server-generated cancelID
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 263: Parameters of the ReadRawWithUpdate Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
OPC_E_MAXEXCEEDED	The maximum number of values returnable by the server was exceeded.
E_INVALIDARG	An invalid parameter was passed.
S_OK	The operation succeeded.

Table 264: Returned Codes of the ReadRawWithUpdate Function

Error Array Returned Codes

Result	Description
S_OK	The function was successful.

S_NODATA	There is no data to be returned.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
E_INVALIDARG	An Invalid parameter was passed.
E_FAIL	The operation failed.

Table 265: Error Array Codes of the ReadRawWithUpdate Function

12.2. ReadProcessedWithUpdate

This function initially retrieves historical processed data within the start time to the end time interval. After the initial response, it periodically (every ftUpdateInterval) responds with dwNumIntervals worth of data divided into ftResampleInterval sized bins. The time of the last value returned in the initial response is used as the start time for the first update. After that, the time of the last value returned in an update is used as the start time for the next update. The following table describes the parameters of this function.

```
int ReadProcessedWithUpdate(int              intServerIndex,
                            int              dwTransactionID,
                            OPCHDA_TIME      htStartTime,
                            OPCHDA_TIME      htEndTime,
                            long             ftResampleInterval,
                            int              dwNumIntervals,
                            long             ftUpdateInterval,
                            int              dwNumItems,
                            int[]            phServer,
                            int[]            haAggregate,
                            out int          pdwCancelID,
                            out int[]        ppErrors,
                            out string strError)
```

Parameters

In/Out	Parameter	Description
In	**intServerIndex**	The identifier of the connection
In	**dwTransactionID**	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
In	**htStartTime**	The beginning of the history period

In	**htEndTime**	The end of the history period
In	**ftResampleInterval**	Interval between returned values
In	**dwNumIntervals**	The number of the update intervals
In	**ftUpdateInterval**	Update interval to send new data
In	**dwNumItems**	The number of items to be read
In	**phServer**	The server item handle for the item to be updated
In	**haAggregate**	The calculation to be performed on the raw data to create the values to be returned. . *Refer to the "Values of the ppdwAggrID parameter"table*
Out	**pdwCancelID**	The server-generated cancelID
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 266: Parameters of the ReadProcessedWithUpdate Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The HDA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
OPC_E_MAXEXCEEDED	The maximum number of values returnable by the server was exceeded.
E_NOTIMPL	This server does not support this function.
E_INVALIDARG	An invalid parameter was passed.

S_OK	The operation succeeded.

Table 267: Returned Codes of the ReadProcessedWithUpdate Function

Error Array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_E_INVALIDHANDLE	The handle is invalid.
OPC_E_BADRIGHTS	Insufficient rights for this operation.
E_FAIL	The operation failed.

Table 268: Error Array Codes of the ReadProcessedWithUpdate Function

12.3. PlaybackCancel

This function cancels the outstanding playbacks operations. The following table describes the parameters of this function.

```
void PlayBackCancel(int      intServerIndex,
                    int      dwCancelID)
```

Parameters

In/Out	Parameter	Description
In	intServerIndex	The identifier of the connection
In	dwCancelID	The server-generated cancelID which was returned from the original method call

Table 269: Parameters of the PlaybackHDACancel Function

customerservice@integrationobjects.com

OPC HDA CLIENT SAMPLES

The present chapter describes the HDA samples available within the installation of OPC .NET Client Toolkit.

1. Step 1: Create User Interface

Create the following user interface with respect to the indicated names of the components.

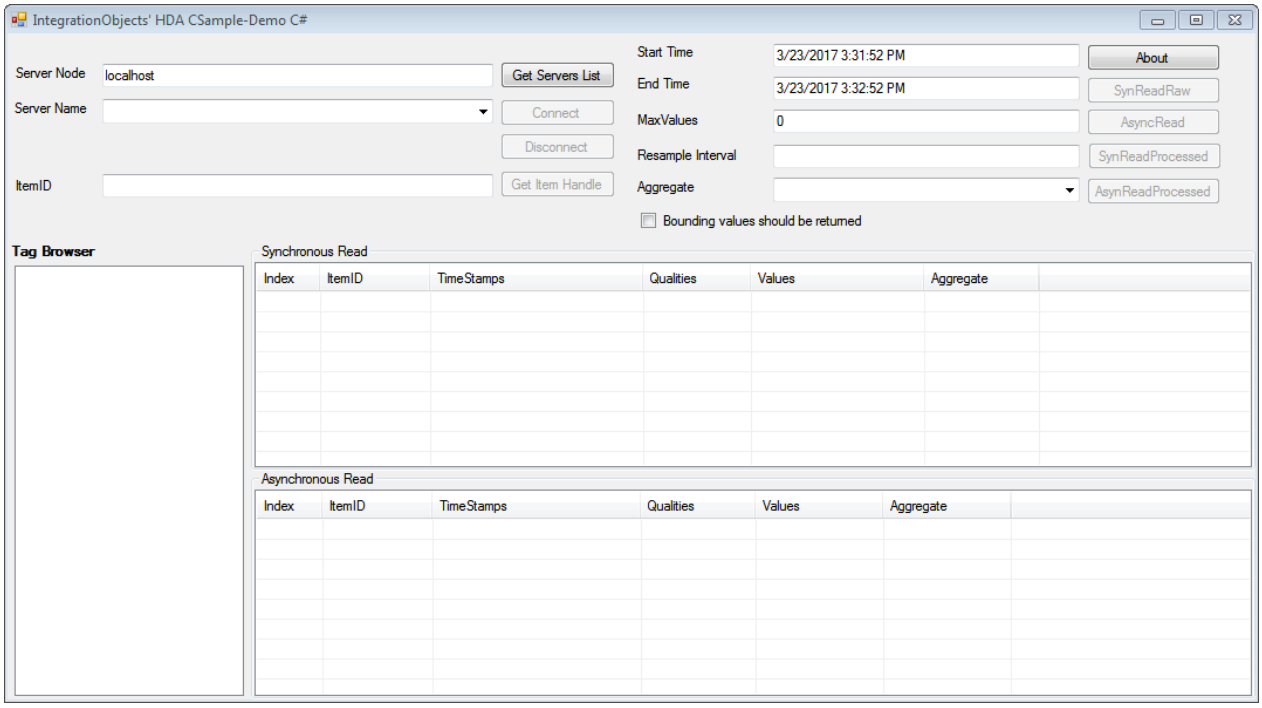

Figure 15: OPC HDA Client - Creating a user interface window

2. Step 2: Initialize HDA Manager and Subscribe to Callbacks

Perform the following declarations:

C#:

```
Private OPCHDAManager objHdaManager  = null;
```

VB .NET:

```
Private objHdaManager As OPC.HDA.Manager.OPCHDAManager = Nothing
```

Perform the following method's implementations:

C#:

```
public void display(OPCHDA_ITEM[] resultAtt){

this.listView4.Items.Clear();
this.resultAtt= resultAtt;
int nb= this.resultAtt[0].dwCount;

for(int i=0;i<nb;i++){
      try{
            string[] Items = new string[]
            {
            i.ToString(),
            itemID[0],
            DateTime.FromFileTime(resultAtt[0].pftTimeStamps[i]).ToString(),
            resultAtt[0].pdwQualities[i].ToString(),
            resultAtt[0].pvDataValues[i].ToString()
                        };
      ListViewItem I1 = new ListViewItem(Items);
      this.listView4.Items.Add(I1);
      }
      catch{
            }
        }
}

public void ReadComplete(
                  int    SrvHandle,
                  int dwTransactionID,
                  int hrStatus,
                  int dwNumItems,
                  OPCHDA_ITEM[] objItemValues,
                  int[] phrErrors)
            {
                  this.display(objItemValues);
            }
```

VB .NET:

```
Public Sub display(ByVal resultAtt() As OPC.HDA.Interface.OPCHDA_ITEM)
            Me.resultAtt = resultAtt
            Dim nb As Integer = Me.resultAtt(0).dwCount
            Dim i As Integer = 0
            Do While (i < nb)
                Try
                    Dim Items() As String = New String() {itemID(0),
DateTime.FromFileTime(resultAtt(0).pftTimeStamps(i)).ToString,
resultAtt(0).pdwQualities(i).ToString, resultAtt(0).pvDataValues(i).ToString}
                    Dim I1 As ListViewItem = New ListViewItem(Items)
                    Me.listView4.Items.Add(I1)
                Catch
                End Try
                i = (i + 1)
            Loop
        End Sub


Public Sub ReadComplete(ByVal SrvHandle As Integer, ByVal dwTransactionID As
Integer, ByVal hrStatus As Integer, ByVal dwNumItems As Integer, ByVal
objItemValues() As OPC.HDA.Interface.OPCHDA_ITEM, ByVal phrErrors() As Integer)
      Me.display(objItemValues)
End Sub
```

Then, double click on the created form and add the following code to initialize HDA Manager and to subscribe to callbacks (here we take the example of ReadComplete callback):

C#:

```
objHdaManager = new OPCHDAManager();
if(objHdaManager != null)
{
objHdaManager.MgtReadComplete= new
MgtReadCompleteEventHandler(this.ReadComplete);
}
```

VB .NET:

```
objHdaManager = New OPC.HDA.Manager.OPCHDAManager
If (Not (objHdaManager) Is Nothing) Then
   objHdaManager.MgtReadComplete = AddressOf Me.ReadComplete
End If
Me.startT.Text = DateTime.Now.ToString
Me.EndT.Text = DateTime.Now.ToString
Me.MaxVal.Text = "0"
```

3. Step 3: List All Available HDA Servers

Double click the **server list button** and add the following code to list all available HDA servers on the network:

C#:

```
try{
this.listserver.Items.Clear();
string host = txtHostName.Text.ToLower();
ArrayList listServer = new ArrayList();
try{
if ((host == "localhost") || (host == "127.0.0.1"))
this.objHdaManager.ListHDAServersFromRegistry(host, out listServer);
else
this.objHdaManager.ListAllHDAServers(host, out listServer);
}
catch
{
}
for(int i=0 ;i<listServer.Count;i++){
this.listserver.Items.Add(listServer[i].ToString());}
if (listServer.Count != 0)
listserver.SelectedIndex = 0;
}
catch
{
}
```

VB .NET:

```
Try
Me.listserver.Items.Clear()
Dim listServer As ArrayList = New ArrayList
Dim host As String = txtHostName.Text.ToLower()
If ((host = "localhost") Or (host = "127.0.0.1")) Then
Me.objHdaManager.ListHDAServersFromRegistry(txtHostName.Text, listServer)
Else
Me.objHdaManager.ListAllHDAServers(txtHostName.Text, listServer)
End If
Dim i As Integer = 0
Do While (i < listServer.Count)
Me.listserver.Items.Add(listServer(i).ToString)
i = (i + 1)
Loop
If Me.listserver.Items.Count > -1 Then
Me.listserver.SelectedIndex = 0
End If
Catch
End Try
```

Perform the following declarations:

C#:
```
private string servername = null;
```

VB .NET:
```
Private servername As String = Nothing
```

Double click the **listserver ComboBox** and add the following code:

C#:
```
   servername=listserver.SelectedItem.ToString();
   this.connect.Enabled=true;
```

VB .NET:
```
   servername = listserver.SelectedItem.ToString
   Me.connect.Enabled = True
```

4. Step 4: Connect and Create Browser

Perform the following declarations:

C#:
```
private int intServerIndex    = -1;
private int intBrowserIndex   = -1;

int dwCount       =      0;
int[]pdwAttrID    =     new int[dwCount];
OPCHDA_OPERATORCODES[] pOperator    =     new OPCHDA_OPERATORCODES[dwCount];
object[]vFilter                     =     new object[dwCount];
int[]ppErrors                       =     new int[dwCount];
string[]msg                         =     new string[dwCount];

string[]str =     new string[100];
string[]str1 =    new string[100];
int length;
int length1;
```

VB .NET:

```
Dim dwCount As Integer = 0
Dim pdwAttrID() As Integer = New Integer(dwCount) {}
Dim pOperator() As OPC.HDA.Interface.OPCHDA_OPERATORCODES
pOperator = New OPC.HDA.Interface.OPCHDA_OPERATORCODES(dwCount) {}
Dim vFilter() As Object = New Object(dwCount) {}
Dim ppErrors() As Integer = New Integer(dwCount) {}

Dim str() As String = New String(100) {}
Dim str1() As String = New String(100) {}
Dim length As Integer
Dim length1 As Integer
```

Double click the **connect button** and add the following code to connect to the selected server and to create the browser

C#:

```
if(servername!=null){
try{
string strError = string.Empty;
      objHdaManager.Connect(servername,txtHostName.Text,out intServerIndex,out
      strError);
}
catch(Exception e1){
throw new Exception(e1.Message,e1); }
}

if (intServerIndex > -1)
{
TreeNode ServerNameNode = new TreeNode(servername, 0, 1);
Browse(intServerIndex,ServerNameNode);
this.treeView1.Nodes.Add(ServerNameNode);
this.button1.Enabled = true;


//Get Aggregates
int iCount = 0;
int[] arrAggregatesID = null;
string[] arrAggregatesNames = null;
string[] ArrDesc = null;
string strError = string.Empty;
int iResult=  objHdaManager.GetAggregates(intServerIndex, out iCount, out arrAggregatesID,
out arrAggregatesNames, out ArrDesc,out strError);

int intAggLength = arrAggregatesNames.Length;
for (int i = 0; i < intAggLength; i++)
{
sourceTypeNbr.Add(arrAggregatesID[i], arrAggregatesNames[i]);
}
```

```
if (sourceTypeNbr.Count > 0)
{
AggregatecomboBox.DataSource = new BindingSource(sourceTypeNbr, null);
AggregatecomboBox.DisplayMember = "Value";
AggregatecomboBox.ValueMember = "Key";
}
}
}
```

```
private void Browse(int index, TreeNode NodeToExpand)
{
try
{
int dwCount = 0;
int[] pdwAttrID = new int[dwCount];
OPCHDA_OPERATORCODES[] pOperator = new OPCHDA_OPERATORCODES[dwCount];
object[] vFilter = new object[dwCount];
int[] ppErrors = new int[dwCount];
string[] msg = new string[dwCount];
int length1 = 0;
try
{
objHdaManager.CreateBrowse(index, dwCount, pdwAttrID, pOperator, vFilter, out
ppErrors, out intBrowserIndex);
string[] strSep1 = new string[0];
string[] strSep1Leaf = new string[0];
string[] strSep2 = new string[0];
string[] strSep3 = new string[0];
string[] strSep4 = new string[0];

int length1Leaf = 0;

int numberOfTries = 0;
while (length1 == 0)
{
numberOfTries++;
objHdaManager.initBrowse(index, intBrowserIndex, OPCHDA_BROWSETYPE.OPCHDA_LEAF,
OPCHDA_BROWSEDIRECTION.OPCHDA_BROWSE_DIRECT, out strSep1Leaf, out length1Leaf);
for (int k = 0; k < length1Leaf; k++)
{
if (strSep1Leaf[k] != null)
{
TreeNode subNode = new TreeNode(strSep1Leaf[k], 0, 1);
subNode.ImageIndex = 0;
NodeToExpand.Nodes.Add(subNode);
}

}
```

C#:

```
int length1Leaf = 0;
int numberOfTries = 0;
while (length1 == 0)
{
numberOfTries++;
objHdaManager.initBrowse(index, intBrowserIndex, OPCHDA_BROWSETYPE.OPCHDA_LEAF,
OPCHDA_BROWSEDIRECTION.OPCHDA_BROWSE_DIRECT, out strSep1Leaf, out length1Leaf);
for (int k = 0; k < length1Leaf; k++)
{
if (strSep1Leaf[k] != null)
{
TreeNode subNode = new TreeNode(strSep1Leaf[k], 0, 1);
subNode.ImageIndex = 0;
NodeToExpand.Nodes.Add(subNode);
}

}
objHdaManager.initBrowse(index, intBrowserIndex, OPCHDA_BROWSETYPE.OPCHDA_BRANCH,
OPCHDA_BROWSEDIRECTION.OPCHDA_BROWSE_DIRECT, out strSep1, out length1);
for (int i = 0; i < length1; i++)
{
if (strSep1[i] != null)
{
              string ItemIDR;
objHdaManager.GetItemID(index, intBrowserIndex, strSep1[i], out ItemIDR);
              TreeNode subNode = new TreeNode(strSep1[i], 0, 1);
subNode.ImageIndex = 0;
subNode = browseBranch(index, intBrowserIndex, strSep1[i], subNode);
objHdaManager.ChangeBrowsePosition(index, intBrowserIndex,
int)OPCHDA_BROWSEDIRECTION.OPCHDA_BROWSE_UP, ItemIDR);
strSep2 = new string[100];
strSep3 = new string[100];
NodeToExpand.Nodes.Add(subNode);
}
}
if (numberOfTries > 10)
{
break;
}
}
}
catch (Exception)
{
// ignored
}
}
catch
{
// ignored
}
}
```

C#:

```
private TreeNode browseBranch(int intServerInd, int intBrowserInd, string
BranchName, TreeNode node){
int LenghtOfLeafs = 0;
string[] ListIemsBranche = new string[0];
string[] ListItemsLeaf = new string[0];
int lengthOfBranches = 0;
string ItemFullName;
objHdaManager.GetItemID(intServerIndex, intBrowserIndex, BranchName, out
ItemFullName);
string test2;
objHdaManager.GetBranchPosition(intServerIndex, intBrowserIndex, out test2);
string ItemName = BranchName;
if (ItemFullName != null){
ItemName = ItemFullName;}
objHdaManager.Browse(intServerInd, intBrowserInd,
(int)OPCHDA_BROWSETYPE.OPCHDA_LEAF,
(int)OPCHDA_BROWSEDIRECTION.OPCHDA_BROWSE_DIRECT, ItemName, out ListItemsLeaf,
out LenghtOfLeafs);
for (int l1 = 0; l1 < LenghtOfLeafs; l1++){
string ItemIDR3;
objHdaManager.GetItemID(intServerInd, intBrowserInd, ListItemsLeaf[l1], out
ItemIDR3);
string FullNameI = ListItemsLeaf[l1];
if (ItemIDR3 != null){
FullNameI = ItemIDR3;}
TreeNode subNode2 = new TreeNode(FullNameI, 0, 1);
node.Nodes.Add(subNode2);}
objHdaManager.Browse(intServerInd, intBrowserInd,
(int)OPCHDA_BROWSETYPE.OPCHDA_BRANCH,
(int)OPCHDA_BROWSEDIRECTION.OPCHDA_BROWSE_DOWN, ItemName, out ListIemsBranche,
out lengthOfBranches);
```

C#:

```
for (int j = 0; j < lengthOfBranches; j++){
string ItemIDR2;
objHdaManager.GetItemID(intServerIndex, intBrowserIndex, ListIemsBranche[j], out
ItemIDR2);
string ItemNameModified = ListIemsBranche[j];
if (ItemIDR2 != null){
ItemNameModified = ItemIDR2;}
TreeNode subNode2 = new TreeNode(ItemNameModified, 0, 1);
objHdaManager.ChangeBrowsePosition(intServerIndex, intBrowserIndex,
(int)OPCHDA_BROWSEDIRECTION.OPCHDA_BROWSE_DOWN, ItemNameModified);
subNode2 = browseBranch(intServerIndex, intBrowserIndex, ListIemsBranche[j],
subNode2);
node.Nodes.Add(subNode2);
objHdaManager.ChangeBrowsePosition(intServerIndex, intBrowserIndex,
(int)OPCHDA_BROWSEDIRECTION.OPCHDA_BROWSE_UP, ItemNameModified);
}
return node;
}
```

VB .NET:

```
If (Not (servername) Is Nothing) Then
Dim strError As String = String.Empty
objHdaManager.Connect(servername, txtHostName.Text, intServerIndex,strError)
Dim ServerNameNode As New TreeNode(servername, 0, 1)
Dim dwCount As Integer = 0
Dim pdwAttrID As Integer() = New Integer(dwCount - 1) {}
Dim pOperator As OPCHDA_OPERATORCODES() = New OPCHDA_OPERATORCODES(dwCount - 1) {}
Dim vFilter As Object() = New Object(dwCount - 1) {}
Dim ppErrors As Integer() = New Integer(dwCount - 1) {}
Dim msg As String() = New String(dwCount - 1) {}
Dim length1 As Integer = 0
Try
objHdaManager.CreateBrowse(intServerIndex, dwCount, pdwAttrID, pOperator, vFilter,
ppErrors, _intBrowserIndex)
Dim strSep1 As String() = New String(-1) {}
Dim strSep1Leaf As String() = New String(-1) {}
Dim strSep2 As String() = New String(-1) {}
Dim strSep3 As String() = New String(-1) {}
Dim strSep4 As String() = New String(-1) {}
Dim length2 As Integer = 0
Dim length1Leaf As Integer = 0
Dim length3 As Integer = 0
Dim index1 As Integer = -1
Dim index2 As Integer = -1
Dim numberOfTries As Integer = 0
While length1 = 0
index1 = -1
numberOfTries += 1
```

VB .NET:

```
objHdaManager.initBrowse(intServerIndex, intBrowserIndex, OPCHDA_BROWSETYPE.OPCHDA_LEAF,
OPCHDA_BROWSEDIRECTION.OPCHDA_BROWSE_DIRECT, strSep1Leaf, length1Leaf)
For k As Integer = 0 To length1Leaf - 1
Dim subNode As New TreeNode(strSep1Leaf(k), 0, 1)
ServerNameNode.Nodes.Add(subNode)
Next
objHdaManager.initBrowse(intServerIndex, intBrowserIndex, OPCHDA_BROWSETYPE.OPCHDA_BRANCH,
OPCHDA_BROWSEDIRECTION.OPCHDA_BROWSE_DIRECT, strSep1, length1)
For i As Integer = 0 To length1 - 1
Dim ItemIDR As String
objHdaManager.GetItemID(intServerIndex, intBrowserIndex, strSep1(i), ItemIDR)
Dim subNode As New TreeNode(strSep1(i), 0, 1)
subNode = browseBranch(intServerIndex, intBrowserIndex, strSep1(i), subNode)
objHdaManager.ChangeBrowsePosition(intServerIndex, intBrowserIndex,
CInt(OPCHDA_BROWSEDIRECTION.OPCHDA_BROWSE_UP), ItemIDR)
strSep2 = New String(99) {}
strSep3 = New String(99) {}
ServerNameNode.Nodes.Add(subNode)
Next
If numberOfTries > 10 Then
Exit While
End If
End While
Catch ex As Exception
End Try
Me.treeView1.Nodes.Add(ServerNameNode)
```

VB .NET:

```
Private Function browseBranch(ByVal intServerInd As Integer, ByVal intBrowserInd As
Integer, ByVal BranchName As String, ByVal node As TreeNode) As TreeNode
Dim LenghtOfLeafs As Integer = 0
Dim ListIemsBranche As String() = New String(-1) {}
Dim ListItemsLeaf As String() = New String(-1) {}
Dim lengthOfBranches As Integer = 0
Dim ItemFullName As String
objHdaManager.GetItemID(intServerIndex, intBrowserIndex, BranchName, ItemFullName)
Dim test2 As String
objHdaManager.GetBranchPosition(intServerIndex, intBrowserIndex, test2)
Dim ItemName As String = BranchName
If ItemFullName IsNot Nothing Then
ItemName = ItemFullName
End If
objHdaManager.Browse(intServerInd, intBrowserInd, CInt(OPCHDA_BROWSETYPE.OPCHDA_LEAF),
CInt(OPCHDA_BROWSEDIRECTION.OPCHDA_BROWSE_DIRECT), ItemName, ListItemsLeaf, _LenghtOfLeafs)
For l1 As Integer = 0 To LenghtOfLeafs - 1
Dim ItemIDR3 As String
objHdaManager.GetItemID(intServerInd, intBrowserInd, ListItemsLeaf(l1), ItemIDR3)
Dim FullNameI As String = ListItemsLeaf(l1)
If ItemIDR3 IsNot Nothing Then
FullNameI = ItemIDR3
End If
Dim subNode2 As New TreeNode(FullNameI, 0, 1)
node.Nodes.Add(subNode2)
Next
objHdaManager.Browse(intServerInd, intBrowserInd, CInt(OPCHDA_BROWSETYPE.OPCHDA_BRANCH),
CInt(OPCHDA_BROWSEDIRECTION.OPCHDA_BROWSE_DOWN), ItemName, ListIemsBranche,
_lengthOfBranches)
For j As Integer = 0 To lengthOfBranches - 1
Dim ItemIDR2 As String
objHdaManager.GetItemID(intServerIndex, intBrowserIndex, ListIemsBranche(j), ItemIDR2)
Dim ItemNameModified As String = ListIemsBranche(j)
If ItemIDR2 IsNot Nothing Then
ItemNameModified = ItemIDR2
End If
Dim subNode2 As New TreeNode(ItemNameModified, 0, 1)
objHdaManager.ChangeBrowsePosition(intServerIndex, intBrowserIndex,
CInt(OPCHDA_BROWSEDIRECTION.OPCHDA_BROWSE_DOWN), ItemNameModified)
subNode2 = browseBranch(intServerIndex, intBrowserIndex, ListIemsBranche(j), subNode2)
node.Nodes.Add(subNode2)
objHdaManager.ChangeBrowsePosition(intServerIndex, intBrowserIndex,
CInt(OPCHDA_BROWSEDIRECTION.OPCHDA_BROWSE_UP), ItemNameModified)
Next
Return node
End Function
```

5. Step 5: Get Item Handle

Perform the following declarations:

C#:
```
private string[]        itemID          = new string[100];
private int[]           hclient         = new int[100];
private int[]           pphserver       = new int[100];
private int[]           ppErrors        = new int[100];
```

VB .NET:
```
Private itemID() As String = New String(100) {}
Private hclient() As Integer = New Integer(100) {}
Private ppErrors() As Integer = New Integer(100) {}
Private pphserver() As Integer = New Integer(100) {}
```

Double click the **treeview** and add the following code to get the ID of the selected item:

C#:
```
this.ItemName.Text= e.Node.Text;
```

VB .NET:
```
Me.ItemName.Text = e.Node.Text
```

Double click the **GetItemHandle button** and add the following code to get the properties of the select item:

C#:
```
if (ItemName.Text != null && ItemName.Text != ""){
itemID[0] = ItemName.Text;
hclient[0] = 1;
objHdaManager.GetItemHandles(intServerIndex, 1, itemID, hclient, out pphserver,
out ppErrors);
MessageBox.Show(itemID[0] + " is connected);
this.AsyncRead.Enabled = true;
this.ReadRaw.Enabled = true;}
else{
MessageBox.Show("Item Name should not be empty. Please select an Item.");}
```

VB .NET:

```
If (Me.ItemName.Text <> "") Then
itemID(0) = Me.ItemName.Text
hclient(0) = 1
objHdaManager.GetItemHandles(intServerIndex, 1, itemID, hclient, pphserver,
ppErrors)
MessageBox.Show((itemID(0) + " is connected"))
Me.AsyncRead.Enabled = True
Me.ReadRaw.Enabled = True
Else
MessageBox.Show("The ItemName should not be empty! Please select an Item.")
  End If
```

6. Step 6: SyncReadRaw

Perform the following declarations:

C#:

```
private OPCHDA_TIME     startTime          = new OPCHDA_TIME();
private OPCHDA_TIME     endTime            = new OPCHDA_TIME() ;
private OPCHDA_ITEM[]   resultAtt          = new OPCHDA_ITEM[100];
private bool            bound              = false;
private int             nbval              =0;
private long            start              =0;
private long            end                =0;
private int[]           err                = new int[100];
```

VB .NET:

```
Private startTime As OPC.HDA.Interface.OPCHDA_TIME = New
OPC.HDA.Interface.OPCHDA_TIME
Private endTime As OPC.HDA.Interface.OPCHDA_TIME = New
OPC.HDA.Interface.OPCHDA_TIME

Private resultAtt() As OPC.HDA.Interface.OPCHDA_ITEM
Private bound As Boolean = False
Private nbval As Integer = 0
Private start As Long = 0
Private TEnd As Long = 0
Private err() As Integer = New Integer(100) {}
```

Double click the **SynReadRaw button** and add the following code to synchronously read raw data:

C#:

```
string msgSR = null;
            this.listView1.Items.Clear();
            try
            {
                start = DateTime.Parse(this.startT.Text).ToFileTime();
                end = DateTime.Parse(this.EndT.Text).ToFileTime();
            }
            catch (Exception e1)
            {
                MessageBox.Show("Please verify the date format : MM/DD/YYYY
HH:MM:SS AM or MM/DD/YYYY HH:MM:SS PM ", "Integration Objects' OPC .Net Client
Toolkit", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }
                  nbval=Int32.Parse(MaxVal.Text);

                  if(checkBox1.Checked)
                  {
                        bound= true;
                  }
                  else{bound= false;}

                  startTime.bString=false;
                  startTime.ftTime=start;
                  startTime.szTime="";

                  endTime.bString=false;
                  endTime.ftTime=end;
                  endTime.szTime="";
                  try
                  {

      objHdaManager.SyncReadRaw(intServerIndex,startTime,endTime,nbval,bound,1,p
phserver,out resultAtt,out err);
                        objHdaManager.getErrors(err[0],out msgSR);
                OPC_Log.TraceLog(OPC_Log.LOG_CONTROLE, "Warning: " + msgSR);

                if (resultAtt[0].dwCount == 0)
                    MessageBox.Show("There is no data archived in the period
spedified", "Integration Objects' OPC .Net Client Toolkit",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                else
                        this.display(resultAtt);
                  }
                  catch
            {
            }
```

VB .NET:

```
Me.listView4.Items.Clear()
            Try
                start = DateTime.Parse(Me.startT.Text).ToFileTime
                TEnd = DateTime.Parse(Me.EndT.Text).ToFileTime
            Catch e1 As Exception
                MessageBox.Show("Please verify the date format : MM/DD/YYYY
HH:MM:SS AM or MM/DD/YYYY HH:MM:SS PM ", "Integration Objects' OPC .Net Client
Toolkit", MessageBoxButtons.OK, MessageBoxIcon.Warning)
                Return
            End Try
            nbval = Int32.Parse(MaxVal.Text)
            If checkBox1.Checked Then
                bound = True
            Else
                bound = False
            End If
            startTime.bString = False
            startTime.ftTime = start
            startTime.szTime = ""
            endTime.bString = False
            endTime.ftTime = TEnd
            endTime.szTime = ""
            Try
                objHdaManager.SyncReadRaw(intServerIndex, startTime, endTime,
nbval, bound, 1, pphserver, resultAtt, err)
                If (resultAtt(0).dwCount = 0) Then

                    MessageBox.Show("There is no data archived in the period
spedified", "Integration Objects' OPC .Net Client Toolkit",
MessageBoxButtons.OK, MessageBoxIcon.Information)

                Else
                    Me.display(resultAtt)
                End If
            Catch
            End Try
```

7. Step 7: AsyncReadRaw

Perform the following declarations:

C#:
```
Private int pdwCancelID =0;
```

VB .NET:
```
Private pdwCancelID As Integer = 0
```

Double click the **AsynReadRaw button** and add the following code to asynchronously read raw data:

VB .NET:
```
Me.listView2.Items.Clear()
            Try
                start = DateTime.Parse(Me.startT.Text).ToFileTime()
                endd = DateTime.Parse(Me.EndT.Text).ToFileTime()
            Catch e1 As Exception
                MessageBox.Show("Please verify the date format : MM/DD/YYYY
HH:MM:SS AM or MM/DD/YYYY HH:MM:SS PM ", "Integration Objects' OPC .Net Client
Toolkit", MessageBoxButtons.OK, MessageBoxIcon.Warning)
                Return
            End Try
            nbval = Int32.Parse(MaxVal.Text)

            If checkBox1.Checked Then
                bound = True
            Else
                bound = False
            End If

            startTime.bString = False
            startTime.ftTime = start
            startTime.szTime = ""

            endTime.bString = False
            endTime.ftTime = endd
            endTime.szTime = ""
            Dim aggr As Integer() = New Integer(0) {}
            aggr(0) = 0
            Try
                objHdaManager.AsyncReadRaw(intServerIndex, 1, startTime,
endTime, nbval, bound, 1, pphserver, pdwCancelID, err)
            Catch
            End Try
```

C#:

```
this.listView2.Items.Clear();
            try
            {
                start = DateTime.Parse(this.startT.Text).ToFileTime();
                  end = DateTime.Parse(this.EndT.Text).ToFileTime();
        }
        catch (Exception e1)
        {
            MessageBox.Show("Please verify the date format : MM/DD/YYYY HH:MM:SS
AM or MM/DD/YYYY HH:MM:SS PM ", "Integration Objects' OPC .NET Client Toolkit",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }
                  nbval=Int32.Parse(MaxVal.Text);

                  if(checkBox1.Checked)
                  {
                        bound= true;
                  }
                  else{
                  bound= false;}

                  startTime.bString=false;
                  startTime.ftTime=start;
                  startTime.szTime="";
            intTest = 0;

                  endTime.bString=false;
                  endTime.ftTime=end;
                  endTime.szTime="";
            int[] aggr=new int[1];
            aggr[0] = 0;
                  try
                  {

      objHdaManager.AsyncReadRaw(intServerIndex,1,startTime,endTime,nbval,bound,
1,pphserver,out pdwCancelID,out err);
                  }
                  catch
                  {
                  }
```

HOW TO INTERACT WITH THE OPC AE .NET CLIENT TOOLKIT

1. Initialization of the API

After the DLL initialization, the client application has the responsibility to properly initialize the API. Thus, the OPC .Net client Toolkit exports the API function named OPCAEManager that performs the basic initialization functions.

2. AE Servers Auto-discovery

OPC .NET Client Toolkit provides the way to auto-discover of all OPC AE servers available on the network. The following table describes the parameters of the **ListAllAEServers** function.

```
void ListAllAEServers(string              host,
                      out ArrayList       lServers)
```

Parameters

In/Out	Parameter	Description
In	**Host**	Name of the machine that hosts OPC DA servers
Out	**lServers**	This parameter will contain the list of located OPC DA servers

Table 270: Parameters of the ListAllAEServers Function

3. Server Management

3.1. Connect To Server

a. Connect

This function establishes a connection to the server identified by `progidOPCserver` and located on the machine strHost. The following table describes the parameters of the **Connect** function.

```
int Connect(string          progidOPCserver,
            string          strHost ,
            out int         intIndex,
            out string strError)
```


customerservice@integrationobjects.com

Parameters

In/Out	Parameter	Description
In	**progidOPCserver**	The server progID
In	**strHost**	The server node name
Out	**intIndex**	If the call succeeds, this parameter will contain a unique identifier for the connection. It should be passed in any further call to the server.
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 271: Parameters of the Connect Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_OK	The operation succeeded.

Table 272: Returned Codes of the Connect Function

b. Connect with Authentication

This function establishes a connection to the server with user authentication. The following table describes the parameters of the **ConnectWithAuthentication** function.

```
int ConnectWithAuthentication(string    progIDOPCserver,
                        string    strHostName,
                        string    strDomain,
                        string    strLogin,
                        string    strPassword,
                        out int    index,
                        out string    strError)
```

Parameters

In/Out	Parameter	Description
In	**progIDOPCserver**	The server progID
In	**strHostName**	The server node name
In	**strDomain**	The domain name. In case the server machine is in the workgroup, this parameter can be replaced with the machine name instead of the domain name.
In	**strLogin**	The login of the server's user.
In	**strPassword**	The password of the server's user.
Out	**index**	If the call succeeds, this parameter will contain a unique identifier for the connection. It should be passed in any further call to the server.
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 273: Parameters of the ConnectWithAuthentication Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The DA feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_OK	The operation succeeded.

Table 274: Returned Codes of the ConnectWithAuthentication Function

3.2. Remove server

This function removes the specified server from the server list. The following table describes the parameters of this function.

```
void RemoveServer(int intIndex)
```

Parameters

In/Out	Parameter	Description
In	**Index**	The identifier of the connection

Table 275: Parameters of the RemoveServer Function

3.3. Disconnect from Server

To disconnect from the server, the client may use the method called **disconnect**, providing the Server Handle returned by the method **connect**. The following table describes the parameters of this function.

```
int Disconnect(int intIndex)
```

Parameters

In/Out	Parameter	Description
In	**Index**	The identifier of the connection

Table 276: Parameters of the Disconnect Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed.
S_OK	The operation succeeded.

Table 277: Returned Codes of the Disconnect Function

3.4. GetStatus

This function returns information about the current server status. The following table describes the parameters of this function.

```
int GetStatus(int                          indexserver,
              out OPCEVENTSERVERSTATUS serverStatus,
              out string strError )
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
Out	**serverStatus**	The current status of the server
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 278: Parameters of the GetStatus Function

The serverStatus parameter values

Status	Value
OPCAE_STATUS_RUNNING	1
OPCAE_STATUS_FAILED	2
OPCAE_STATUS_NOCONFIG	3
OPCAE_STATUS_SUSPENDED	4
OPCAE_STATUS_TEST	5
OPCAE_STATUS_COMM_FAULT	6

Table 279 : Values of the serverStatus Parameter

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid

E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_OUTOFMEMORY	Not enough memory
E_INVALIDARG	An argument to the function was invalid.
S_OK	The operation succeeded.

Table 280: Returned Codes of the GetStatus Function

3.5. QueryAvailableFilters

This function indicates to the clients which filter criteria are supported by a given server. The following table describes the parameters of this function.

```
int QueryAvailableFilters (int                indexserver,
                           out int            pdwFilterMask,
                           out string strError)
```

Parameters

In/Out	Parameter	Description
In	Indexserver	The identifier of the connection
Out	pdwFilterMask	Indicates the types of masks supported by the server. Refer to the Filter Mask Values table.
Out	strError	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 281: Parameters of the QueryAvailableFilters Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.

S_OK	The operation succeeded.

Table 282: Returned Codes of the QueryAvailableFilters Function

Filter Mask Values

Status	Value
OPC_FILTER_BY_EVENT	1
OPC_FILTER_BY_CATEGORY	2
OPC_FILTER_BY_SEVERITY	4
OPC_FILTER_BY_AREA	8
OPC_FILTER_BY_SOURCE	16

Table 283: Filter Mask Values of the QueryAvailableFilters Function

3.6. QueryEventCategories

This function indicates to the clients the specific categories of events supported by a given server. The following table describes the parameters of this function.

```
int QueryEventCategories(int                indexserver,
                         int                dwBufferType,
                         out int            pdwCount,
                         out int[]          ppdwEventCategories,
                         out string[]       ppEventCategoryDescs,
                          out string strError)
```

Parameters

In/Out	Parameter	Description
In	Indexserver	The identifier of the connection
In	dwBufferType	The event types which can be one of following: • OPC_SIMPLE_EVENT =1 • OPC_TRACKING_EVENT=2 • OPC_CONDITION_EVENT=4

		• OPC_ALL_EVENTS=7
Out	pdwCount	The number of event categories
Out	ppdwEventCategories	The vendor-specific event categories implemented by the server
Out	ppEventCategoryDescs	Text names and description for the event categories
Out	strError	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 284: Parameters of the QueryEventCategories Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_OUTOFMEMORY	Not enough memory
E_INVALIDARG	An argument to the function was invalid.
S_OK	The operation succeeded.

Table 285: Returned Codes of the QueryEventCategories Function

3.7. QueryConditionNames

This function gives to the clients the specific condition names which the event server supports for the specified event category. The following table describes the parameters of this function.

```
int QueryConditionNames(int       indexserver,
                        int           dwEventCategory,
                        out string[]  ppszConditionNames,
                        out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**dwEventCategory**	The vendor-specific event categories implemented by the server
Out	**ppszConditionNames**	The condition names for the specified event categories
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 286: Parameters of the QueryConditionNames Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	A bad parameter was passed.
E_OUTOFMEMORY	Not enough memory
E_NOTIMPL	The server does not support this function
S_OK	The operation succeeded.

Table 287: Returned Codes of the QueryConditionNames Function

3.8. QuerySubConditionNames

This function gives the clients the specific sub-condition names which are associated with the specified condition name. The following table describes the parameters of this function.

```
int QuerySubConditionNames(int                indexserver,
                           string             szConditionName,
                           out string[]       ppszSubConditionNames,
                           out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**szConditionName**	Condition name
Out	**ppszSubConditionNames**	SubCondition names associated with the condition name
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 288: Parameters of the SubConditionNames Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	A bad parameter was passed.
E_OUTOFMEMORY	Not enough memory
E_NOTIMPL	The server does not support this function
S_OK	The operation succeeded.

Table 289: Returned Codes of the QuerySubConditionNames Function

3.9. QuerySourceConditions

This function gives the clients the specific condition names associated with the specified source. The following table describes the parameters of this function.

```
int QuerySourceConditions(int             indexserver,
                          string          szSource,
                          out string []   ppszConditionNames,
                           out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**szSource**	The source name
Out	**ppszConditionNames**	The condition names for the specified source
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 290: Parameters of the QuerySourceConditions Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	A bad parameter was passed.
E_OUTOFMEMORY	Not enough memory
E_NOTIMPL	The server does not support this function
S_OK	The operation succeeded.

Table 291: Returned Codes of the QuerySourceConditions Function

3.10. QueryEventAttributes

This function gives to the clients the vendor-specific attributes the server can provide as part of an event notification for an event within the specified event category. The following table describes the parameters of this function.

```
int QueryEventAttributes(int             indexserver,
                         int             dwEventCategory,
                         out int []      ppdwAttrIDs,
                         out string []   ppszAttrDescs,
                         out VarEnum[]   ppvtAttrTypes,
                          out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**dwEventCategory**	The event category
Out	**ppdwAttrIDs**	Vendor-specific event attributes ID associated with the event category
Out	**ppszAttrDescs**	The event attributes description
Out	**ppvtAttrTypes**	The event attributes type
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 292: Parameters of the QueryEventAttributes Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	A bad parameter was passed.

E_OUTOFMEMORY	Not enough memory
S_OK	The operation succeeded.

Table 293: Returned Codes of the QueryEventAttributes Function

3.11. TranslateToItemIDs

This function is useful in a case where the client wishes to use the OPC Data Access interface to subscribe to real-time data associated with a given event or alarm. The following table describes the parameters of this function.

```
int TranslateToItemIDs(
          int                    indexserver,
          string                 szSource,
          int                    dwEventCategory,
          string                 szConditionName,
          string                 szSubconditionName,
          int                    dwCount,
          int[]                  pdwAssocAttrIDs,
          out string[]           ppszAttrItemIDs,
          out string[]           ppszNodeNames,
          out int[]              ppCLSIDs,
          out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**szSource**	The source name
In	**dwEventCategory**	The event category
In	**szConditionName**	The condition name
In	**szSubconditionName**	The subcondition name
In	**dwCount**	The number of event attributes
In	**pdwAssocAttrIDs**	Associated attribute ID
Out	**ppszAttrItemIDs**	Item ID available for the associated attribute
Out	**ppszNodeNames**	Network node names of the associated OPC Data Access Servers

Out	**ppCLSIDs**	CLSID of the associated OPC Data Access server
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 294: Parameters of the TranslateToItemIDs Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_NOTIMPL	This capability not implemented by this server.
E_INVALIDARG	A bad parameter was passed.
E_OUTOFMEMORY	Not enough memory
S_OK	The operation succeeded.

Table 295: Returned Codes of the TranslateToItemIDs Function

3.12. GetConditionState

This function returns the current state information for the condition instance corresponding to the szSource and szConditionName. The following table describes the parameters of this function.

```
int GetConditionState(int                    indexEvent,
                      string                 szSource,
                      string                 szConditionName,
                      int                    dwNumberEventAttrs,
                      int[]                  pdwAttributeIDs,
                      out CONDITIONSTATE     condState,
                      out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**szSource**	The source name
In	**szConditionName**	The condition name
In	**dwNumberEventAttrs**	The requested number of event attributes
In	**pdwAttributeIDs**	Attribute IDs
Out	**condState**	The returned condition state structure
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 296: Parameters of the GetConditonState Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_NOTIMPL	This capability not implemented by this server.
E_INVALIDARG	A bad parameter was passed.
E_OUTOFMEMORY	Not enough memory
OPC_E_NOINFO	No information is currently available for the specified condition.
S_OK	The operation succeeded.

Table 297: Returned Codes of the GetConditonState Function

3.13. EnableConditionByArea

This function places the specified process areas into the enabled state. The following table describes the parameters of this function.

```
int EnableConditionByArea(int          serverindex,
                          int          dwNumAreas,
                          string[]     pszAreas,
                          out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**dwNumAreas**	The number of areas
In	**pszAreas**	The areas names
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 298: Parameters of the EnableConditionByArea Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_NOTIMPL	This capability not implemented by this server.
E_INVALIDARG	A bad parameter was passed.
S_OK	The operation succeeded.

Table 299: Returned Codes of the EnableConditionByArea Function

3.14. EnableConditionBySource

This function places all conditions for the specified event sources into the enabled state. The following table describes the parameters of this function.

```
int EnableConditionBySource(int              Indexserver,
                             int             dwNumSrc,
                            string[]         pszSrc,
                            out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**dwNumSrc**	The number of source
In	**pszSrc**	The sources names
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 300: Parameters of the EnableConditionBySource Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_NOTIMPL	This capability not implemented by this server.
E_INVALIDARG	A bad parameter was passed.
S_OK	The operation succeeded.

Table 301: Returned Codes of the EnableConditionBySource Function

3.15. DisableConditionByArea

This function places the specified process areas into the disabled state. The following table describes the parameters of this function.

```
int DisableConditionByArea(int          serverindex,
                           int          dwNumAreas,
                           string[]     pszAreas,
                           out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**dwNumAreas**	The number of areas
In	**pszAreas**	The areas names
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 302: Parameters of the DisableConditionByArea Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_NOTIMPL	This capability not implemented by this server.
E_INVALIDARG	A bad parameter was passed.
S_OK	The operation succeeded.

Table 303: Returned Codes of the DisableConditionByArea Function

3.16. DisableConditionBySource

This function places all conditions for the specified event sources into the disabled state. The following table describes the parameters of this function.

```
int DisableConditionBySource(int       serverindex,
                             int       dwNumSrc,
                             string[]  pszSrc,
                             out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**dwNumSrc**	The number of sources
In	**pszSrc**	The sources names
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 304: Parameters of the DisableConditionBySource Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_NOTIMPL	This capability not implemented by this server.
E_INVALIDARG	A bad parameter was passed.
S_OK	The operation succeeded.

Table 305: Returned Codes of the DisableConditionBySource Function

3.17. AckCondition

This function enables the client to acknowledge one or more conditions in the Event Server. The following table describes the parameters of this function.

```
int AckCondition(int             indexserver,
                 int             dwCount,
                 string          szAcknowledgerID,
                 string          szComment,
                 string[]        pszSource,
                 string[]        pszConditionName,
                 long[]          pftActiveTime,
                 int[]           pdwCookie,
                 out int[]       ppErrors,
               out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**dwCount**	The number of acknowledgements
In	**szAcknowledgerID**	The acknowledgement ID
In	**szComment**	Comment associated with acknowledgements
In	**pszSource**	Event source
In	**pszConditionName**	Condition name
In	**pftActiveTime**	The active time of the sources and conditions names
In	**pdwCookie**	Server supplied cookies of the sources and the conditions names
Out	**ppErrors**	Returned errors
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 306: Parameters of the AckCondition Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_NOTIMPL	This capability not implemented by this server.
E_INVALIDARG	A bad parameter was passed.
E_OUTOFMEMORY	Not enough memory.
S_FALSE	The operation completed with one or more errors. Refer to ppErrors for more information.
S_OK	The operation succeeded.

Table 307: Returned Codes of the AckCondition Function

Error array Returned Codes

Result	Description
S_OK	The function was successful.
OPC_S_ALREADYACKED	The condition has already been acknowledged.
OPC_E_INVALIDTIME	The pftActiveTime did not match the current SubCondLastActive attribute of the condition.
E_INVALIDARG	An Invalid parameter was passed.

Table 308: Error Array Codes of the AckCondition Function

3.18. CreateAreaBrowser

This function creates a new instance of a browser. The following table describes the parameters of this function.

```
int   CreateAreaBrowser(int              indexserver,
                        out object       ppUnk,
                        out  BROWSER     objBrowserStruct,
                        out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
Out	**ppUnk**	The returned interface

Out	**objBrowserStruct**	The returned browser structure whose parameter are described as following:
		`string strBranchName;`
		`int intAreaNumber;`
		`int intSrcName;`
		`string[] objSrcList;`
		`BROWSER[] objAreaList;`
		You need to implement your own algorithm to retrieve the areas and sources lists from the returned **objBrowserStruct BROWSER** class.

Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 309: Parameters of the CreateAreaBrowser Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_NOTIMPL	This capability not implemented by this server.

E_INVALIDARG	A bad parameter was passed.
E_OUTOFMEMORY	Not enough memory.
E_NOINTERFACE	The interface IOPCEventAreaBrowser is not supported by the server.
S_OK	The operation succeeded.

Table 310: Returned Codes of the CreateAreaBrowser Function

3.19. CreateEventSubscription

This function adds an Event Subscription object to an Event Server. The following table describes the parameters of this function.

```
int CreateEventSubscription(int         serverindex,
                            string      strSubsName,
                            bool        bActive,
                            int         dwBufferTime,
                            int         dwMaxSize,
                            int         hClientSubscription,
                            out int     pdwRevisedBufferTime,
                            out int     pdwRevisedMaxSize,
                            out int     indexsubs,
                            out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**strSubsName**	Subscription name
In	**bActive**	Subscription active state
In	**dwBufferTime**	The requested buffer time
In	**dwMaxSize**	The requested maximum number of events
In	**hClientSubscription**	Client provided handle for this subscription
Out	**pdwRevisedBufferTime**	The buffer time actually provided by the server

Out	**pdwRevisedMaxSize**	The maximum number of events actually sent by the server
Out	**Indexsubs**	The returned subscription identifier
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 311: Parameters of the CreateEventSubscription Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	A bad parameter was passed.
E_OUTOFMEMORY	Not enough memory.
E_NOINTERFACE	The interface IOPCEventAreaBrowser is not supported by the server.
OPC_S_INVALIDBUFFERTIME	The buffer time parameter was invalid.
OPC_S_INVALIDMAXSIZE	The max size parameter was invalid.
S_OK	The operation succeeded.

Table 312: Returned Codes of the CreateEventSubscription Function

4. AE Subscriptions Management

4.1. ActivateSubscription

This function activates the specified subscription. The following table describes the parameters of this function.

```
int ActivateSubscription(int          indexserver,
                         int          indexsubs,
                         out string   strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**Indexsubs**	Subscription identifier
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 313: Parameters of the ActivateSubscription Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	A bad parameter was passed.
OPC_S_INVALIDBUFFERTIME	The buffer time parameter was invalid.
OPC_S_INVALIDMAXSIZE	The max size parameter was invalid.
S_OK	The operation succeeded.

Table 314: Returned Codes of the ActivateSubscription Function

4.2. RemoveSubscription

This function removes the specified subscription. The following table describes the parameters of this function.

```
int RemoveSubscription(int        indexserver,
```

```
                int       indexsubs,
              out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**Indexsubs**	Subscription identifier
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 315: Parameters of the RemoveSubscription Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_OK	The operation succeeded.

Table 316: Returned Codes of the RemoveSubscription Function

4.3. DeactivateSubscription

This function deactivates the specified subscription. The following table describes the parameters of this function.

```
int DeactivateSubscription(int          indexserver,
                           int          indexsubs,
                           out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection

In	**Indexsubs**	Subscription identifier
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 317: Parameters of the DeactivateSubscription Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	A bad parameter was passed.
OPC_S_INVALIDBUFFERTIME	The buffer time parameter was invalid.
OPC_S_INVALIDMAXSIZE	The max size parameter was invalid.
S_OK	The operation succeeded.

Table 318: Returned Codes of the ActivateSubscription Function

4.4. SetFilter

This function sets the filtering criteria to be used for the event subscription. The following table describes the parameters of this function.

```
int SetFilter(int           indexserver,
              int           indexsubs,
              int           dwEventType,
              int           dwNumCategories,
              int[]         pdwEventCategories,
              int           dwLowSeverity,
              int           dwHighSeverity,
              int           dwNumAreas,
              string[]      pszAreaList,
              int           dwNumSources,
              string[]      pszSourceList,
             out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection : You can get this value from the connect method server index output parameter as shown below : `objAeManager.Connect(servername, txtHostName.Text, out intServerIndex);`
In	**Indexsubs**	Subscription identifier: You can get this value from the CreateEventSubscription method subscription index output parameter as shown below : `objAeManager.CreateEventSubscription(intServerIndex, name, true, 100, 100, count, out RBufferTime, out RMaxSize, out indexsub);`
In	**dwEventType**	The event types which can be one of the following: • OPC_SIMPLE_EVENT =1 • OPC_TRACKING_EVENT=2 • OPC_CONDITION_EVENT=4 • OPC_ALL_EVENTS=7
In	**dwNumCategories**	The number of event categories: You need to call the method QueryEventCategories to get the categories number as shown below: `objAeManager.QueryEventCategories(intServerIndex, 7, out iEventCategoryCount, out iEventCategories, out strCategoryDescs)`
In	**pdwEventCategories**	The event categories: You need to call the method QueryEventCategories to get the event categories as shown below: `objAeManager.QueryEventCategories(intServerIndex, 7, out iEventCategoryCount, out iEventCategories, out strCategoryDescs)`
In	**dwLowSeverity**	Lowest security of interest. The value is between 1 and 1000
In	**dwHighSeverity**	Highest security of interest. The value is between 1 and 1000

In	**dwNumAreas**	The number of the selected areas to be filtered
In	**pszAreaList**	The areas names : You need to call the method CreateAreaBrowser to retrieve the areas list from the returned Browser struct as shown below: `objAeManager.CreateAreaBrowser(intServerIndex, out ppUnk, out ObjBrowserStruct)`
In	**dwNumSources**	The number of the selected sources to be filtered
In	**pszSourceList**	The sources names. You need to call the method CreateAreaBrowser to retrieve the sources list from the returned Browser struct as shown below: `objAeManager.CreateAreaBrowser(intServerIndex, out ppUnk, out ObjBrowserStruct)`
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 319: Parameters of the SetFilter Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	A bad parameter was passed.
E_OUTOFMEMORY	Not enough memory
OPC_E_BUSY	A refresh operation is currently in progress on this event subscription object.
S_OK	The operation succeeded.

S_FALSE	One or more of the specified filter criteria were ignored.

Table 320: Returned Codes of the SetFilter Function

4.5. GetFilter

This function returns the filter currently in use for the event subscriptions. The following table describes the parameters of this function.

```
int GetFilter(int                indexserver,
              int                indexsubs,
              out int            pdwEventType,
              out int[]          ppdwEventCategories,
              out int            pdwLowSeverity,
              out int            pdwHighSeverity,
              out string []      ppszAreaList,
              out string []      ppszSourceList,
               out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**Indexsubs**	Subscription identifier
Out	**pdwEventType**	The event types
Out	**ppdwEventCategories**	The event categories
Out	**pdwLowSeverity**	Lowest security of interest
Out	**pdwHighSeverity**	Highest security of interest
Out	**ppszAreaList**	The areas names
Out	**ppszSourceList**	The sources names
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 321: Parameters of the GetFilter Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	A bad parameter was passed.
E_OUTOFMEMORY	Not enough memory
S_OK	The operation succeeded.

Table 322: Returned Codes of the GetFilter Function

4.6. SelectReturnedAttributes

This function sets the attributes to be returned for each Event Category. The following table describes the parameters of this function.

```
int SelectReturnedAttributes(int        indexserver,
                             int        indexsubs,
                             int        dwEventCategory,
                             int []     dwAttributeIDs,
                             out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**Indexsubs**	Subscription identifier
In	**dwEventCategory**	The event category
In	**dwAttributeIDs**	The selected attribute IDs
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned

		error message. Otherwise, it will contain an empty string.

Table 323: Parameters of the SelectReturnedAttributes Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	A bad parameter was passed.
S_OK	The operation succeeded.

Table 324: Returned Codes of the SelectReturnedAttributes Function

4.7. GetReturnedAttributes

This function retrieves the attributes that are currently specified to be returned for each Event Category. The following table describes the parameters of this function.

```
int GetReturnedAttributes(int           indexserver,
                          int           indexsubs,
                          int           dwEventCategory,
                          out int[]     pdwAttributeIDs,
                           out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**Indexsubs**	Subscription identifier
In	**dwEventCategory**	The event category
Out	**pdwAttributeIDs**	Returned attribute IDs
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned

		error message. Otherwise, it will contain an empty string.

Table 325: Parameters of the GetReturnedAttributes Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	A bad parameter was passed.
S_OK	The operation succeeded.

Table 326: Returned Codes of the GetReturnedAttributes Function

4.8. Refresh

This function forces a refresh for all conditions. The following table describes the parameters of this function.

```
int Refresh(int       indexserver,
            int       indexsubs,
            int       dwConnection,
            out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection
In	**Indexsubs**	Subscription identifier
In	**dwConnection**	The OLE connection number
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 327: Parameters of the Refresh Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
OPC_E_BUSY	There is currently another refresh in progress on this event subscription.
E_NOTIMPL	The server does not support this method
S_OK	The operation succeeded.

Table 328: Returned Codes of the Refresh Function

4.9. CancelRefresh

This function cancels a refresh in progress for the event subscription. The following table describes the parameters of this function.

```
int CancelRefresh(int       indexserver,
                  int       indexsubs,
                  int       dwConnection,
                  out string strError)
```

Parameters

In/Out	Parameter	Description
In	Indexserver	The identifier of the connection
In	Indexsubs	Subscription identifier
In	dwConnection	The OLE connection number
Out	strError	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 329: Parameters of the CancelRefresh Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	dwConnection is not valid.
E_NOTIMPL	The server does not support this method
S_OK	The operation succeeded.

Table 330: Returned Codes of the CancelRefresh Function

4.10. GetState

This function gets the current state of the subscription. The following table describes the parameters of this function.

```
int GetState(int      indexserver,
             int          indexsubs,
             out bool     pbActive,
             out int      pdwBufferTime,
             out int      pdwMaxSize,
             out int      phClientSubscription,
              out string strError)
```

Parameters

In/Out	Parameter	Description
In	Indexserver	The identifier of the connection
In	Indexsubs	Subscription identifier
Out	pbActive	The active state
Out	pdwBufferTime	The buffer time
Out	pdwMaxSize	The requested maximum number of events

Out	phClientSubscription	Client provided handle for this subscription
Out	strError	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 331: Parameters of the GetState Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
S_OK	The operation succeeded.

Table 332: Returned Codes of the GetState Function

4.11. SetState

This function sets the properties of the subscription. The following table describes the parameters of this function.

```
int SetState(int            indexserver,
             int            indexsubs,
             bool[]         pbActive,
             int[]          pdwBufferTime,
             int[]          pdwMaxSize,
             int            phClientSubscription,
             out int        pdwRevisedBufferTime,
             out int        pdwRevisedMaxSize,
            out string strError)
```

Parameters

In/Out	Parameter	Description
In	**Indexserver**	The identifier of the connection

In	**Indexsubs**	Subscription identifier
In	**pbActive**	The active state
In	**pdwBufferTime**	The buffer time
In	**pdwMaxSize**	The requested maximum number of events
In	**phClientSubscription**	Client provided handle for this subscription
Out	**pdwRevisedBufferTime**	The buffer time actually provided by the server
Out	**pdwRevisedMaxSize**	The maximum number of events actually sent by the server
Out	**strError**	If the call fails at the OPC .NET Client Toolkit level, this parameter will contain the returned error message. Otherwise, it will contain an empty string.

Table 333: Parameters of the SetState Function

Returned Codes

Return Code	Description
CLASS_E_NOTLICENSED	The AE feature license for the OPC .NET Client Toolkit is not valid
E_FAIL	The operation failed. Refer to the strError parameter for more details about the error.
E_INVALIDARG	A bad parameter was passed.
OPC_S_INVALIDBUFFERTIME	The buffer time parameter was invalid.
OPC_S_INVALIDMAXSIZE	The max size parameter was invalid.
S_OK	The operation succeeded.

Table 334: Returned Codes of the SetState Function

OPC AE CLIENT SAMPLES

The chapter describes the AE samples available within the installation of OPC .NET Client Toolkit.

1. Step 1: Create User Interface

Create the following user interface with respect to the indicated names of the components.

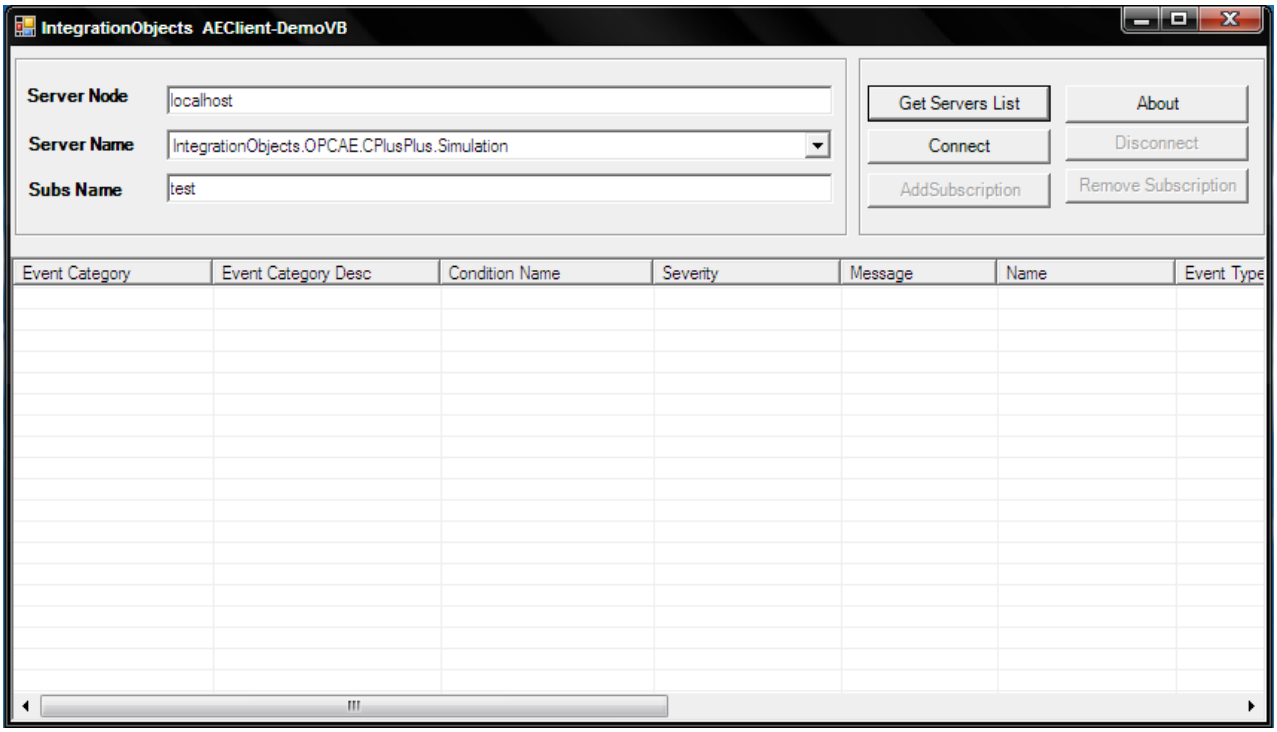

Figure 16: OPC AE Client - Create User Interface Window

customerservice@integrationobjects.com

2. Step 2: Initialize AE Manager and Subscribe to Callbacks

Perform the following declarations:

C#:
```
Private OPCAEManager objAeManager   = null;
```

VB .NET:
```
Private objAeManager As OPC.AE.Manager.OPCAEManager = Nothing
```

Perform the following method's implementations:

C#:
```
public void displayAE(ONEVENTSTRUCT[] resultAtt)
        {


            if (this.listViewSubs.InvokeRequired)
            {
                UpdateListViewSubs delUpdateListView = new
UpdateListViewSubs(displayAE2);
                object[] tmpobjArray = new object[1];
                tmpobjArray[0] = resultAtt;

                try
                {
                    this.Invoke(delUpdateListView, tmpobjArray); //if it'll
block change Invoke-->BeginInvoke
                }
                catch (Exception ec) { }
            }
            else
                displayAE2(resultAtt);

        }
```

C#:

```
public void displayAE2(ONEVENTSTRUCT[] resultAtt)
        {
int nb = resultAtt[0].dwNumEventAttrs;
int EventCategory = resultAtt[0].dwEventCategory;
string EventCatDesc = "";
string strEventType = "";
string szConditionName="";
int dwSeverity = resultAtt[0].dwSeverity;
string szMessage = "";
string szName = "";
string szServerProgID = "";
string szSource = "";
string szSubconditionName = "";
try{
if(resultAtt[0].strEventCatDesc!= null){
   EventCatDesc = resultAtt[0].strEventCatDesc.ToString();}
if (resultAtt[0].szConditionName != null){
   szConditionName = resultAtt[0].szConditionName.ToString();}
if (resultAtt[0].szMessage != null){
   szMessage = resultAtt[0].szMessage.ToString();}
if (resultAtt[0].szName != null){
   szName = resultAtt[0].szName.ToString();}
if (resultAtt[0].strEventType != null){
   strEventType = resultAtt[0].strEventType.ToString();}
if (resultAtt[0].szServerProgID != null){
   szServerProgID = resultAtt[0].szServerProgID.ToString();}
if (resultAtt[0].szSource != null){
   szSource = resultAtt[0].szSource.ToString();}
if (resultAtt[0].szSubconditionName != null){
   szSubconditionName = resultAtt[0].szSubconditionName.ToString();}
string[] Items = new string[]{EventCategory.ToString(),EventCatDesc,
szConditionName,dwSeverity.ToString(),szMessage,szName,strEventType,szServerPr
ogID,szSource,szSubconditionName,DateTime.FromFileTime(resultAtt[0].ftActiveTi
me).ToString(),DateTime.FromFileTime(resultAtt[0].ftTime).ToString()};
ListViewItem I1 = new ListViewItem(Items);
this.listViewSubs.Items.Add(I1);}
catch (Exception e20){string str = e20.Message;}}

public void OnEvent(
                  int                     indexserver,
                  int                     indxsubs,
                  int                     hGroup,
                  bool              bRefresh,
                  bool              bLastRefresh,
                  int                     dwCount,
                  ONEVENTSTRUCT[]   pEvents)
            {
                  displayAE(pEvents);
            }
```

VB .NET:

```
Public Sub display(ByVal resultAtt As ONEVENTSTRUCT())
Try
If Me.listViewSubs.InvokeRequired Then
Dim delUpdateListView As New UpdateListViewSubs(AddressOf display2)
Dim tmpobjArray As Object() = New Object(0) {}
tmpobjArray(0) = resultAtt
Try
'if it'll blocked change Invoke-->BeginInvoke
Me.Invoke(delUpdateListView, tmpobjArray)
Catch ec As Exception
End Try
Else
display2(resultAtt)
End If
Catch
End Try
End Sub

Public Sub display2(ByVal resultAtt() As OPC.AE.Interface.ONEVENTSTRUCT)

Dim nb As Integer = resultAtt(0).dwNumEventAttrs
Try
Dim Items() As String = New String(){
resultAtt(0).dwEventCategory.ToString, resultAtt(0).strEventCatDesc.ToString,
resultAtt(0).szConditionName.ToString, resultAtt(0).dwSeverity.ToString,
resultAtt(0).szMessage.ToString, resultAtt(0).szName.ToString,
resultAtt(0).strEventType.ToString, resultAtt(0).szServerProgID.ToString,
resultAtt(0).szSource.ToString, resultAtt(0).szSubconditionName.ToString,
DateTime.FromFileTime(resultAtt(0).ftActiveTime).ToString,
DateTime.FromFileTime(resultAtt(0).ftTime).ToString}
Dim I1 As ListViewItem = New ListViewItem(Items)
Me.listViewSubs.Items.Add(I1)
Catch
End Try

Public Sub OnEvent(ByVal indexserver As Integer, ByVal indxsubs As Integer,
ByVal hGroup As Integer, ByVal bRefresh As Boolean, ByVal bLastRefresh As
Boolean, ByVal dwCount As Integer, ByVal pEvents() As
OPC.AE.Interface.ONEVENTSTRUCT)
   display(pEvents)
End Sub
```

Then, double-click on the created form and add the following code to initialize AE manager and to subscribe to OnEvent callback:

C#:
```
objAeManager      =     new OPCAEManager();
if(objAeManager != null)
      {
      objAeManager.MyEvent2 += new MyEventEventHandler2(this.OnEvent);
      }
this.startT.Text= DateTime.Now.ToString() ;
```

VB .NET:
```
objAeManager = New OPC.AE.Manager.OPCAEManager
If (Not (objAeManager) Is Nothing) Then
     objAeManager.MyEvent2 = AddressOf Me.OnEvent
End If
```

3. Step 3: List All Available AE Servers

Double click the **server list button** and add the following code to list all available AE servers on the network:

C#:
```
try{
this.listserver.Items.Clear()
ArrayList listServer= new ArrayList();
string host = txtHostName.Text.ToLower();
if ((host == "localhost")||(host == "127.0.0.1"))
this.objAeManager.ListAEServersFromRegistry(txtHostName.Text,out listServer);
else
this.objAeManager.ListAllAEServers(txtHostName.Text, out listServer);
for(int i=0 ;i<listServer.Count;i++)
{
this.listserver.Items.Add(listServer[i].ToString());
}
if (listServer.Count != 0)
listserver.SelectedIndex = 0;
}
```

VB .NET:
```
Try
Me.listserver.Items.Clear()
Dim listServer As ArrayList = New ArrayList
Dim host As String = txtHostName.Text.ToLower()
If ((host = "localhost") Or (host = "127.0.0.1")) Then
Me.objAeManager.ListAEServersFromRegistry(txtHostName.Text, listServer)
Else
Me.objAeManager.ListAllAEServers(txtHostName.Text, listServer)
End If
Dim i As Integer = 0
Do While (i < listServer.Count)
Me.listserver.Items.Add(listServer(i).ToString)
i = (i + 1)
Loop
If (listServer.Count > 0) Then
Me.listserver.SelectedIndex = 0
End If
Catch
End Try
```

Perform the following declarations:

C#:
```
private string servername = null;
```

VB .NET:
```
Private servername As String = Nothing
```

Double-click the **listserver ComboBox** and add the following code:

C#:
```
servername=listserver.SelectedItem.ToString();
this.connect.Enabled=true;
```

VB .NET:
```
connect.Enabled = True
servername = listserver.SelectedItem.ToString
```

4. Step 4: Connect to an AE Server

Perform the following declarations:

C#:
```
private int  intServerIndex        =     -1;
```

VB.NET:
```
Private intServerIndex As Integer   =     -1
```

Double click the **connect button** and add the following code to connect to the selected server:

C#:
```
if(servername!=null)
{
string strError = string.Empty;
objAeManager.Connect(servername,txtHostName.Text,out intServerIndex,out
strError);
this.AddSub.Enabled=true;
}
```

VB .NET:
```
If (Not (servername) Is Nothing) Then
   objAeManager.Connect(servername, txtHostName.Text, intServerIndex, strError)
   Me.AddSub.Enabled = True
End If
```

5. Step 5: Add Subscription

Perform the following declarations:

C#:
```
private int                  RBufferTime       =     -1;
private int                  RMaxSize          =     -1;
private int                  indexsub          =     -1;
```

VB.NET:
```
private int                  RBufferTime       =     -1;
private int                  RMaxSize          =     -1;
private int                  indexsub          =     -1;
```

Double-click the **Addsubscription button** and add the following code to Add subscriptions:

C#:

```
objAeManager.CreateEventSubscription(intServerIndex,"test",true,100,100,1,out
RBufferTime,out RMaxSize,out indexsub, out strError);
```

VB .NET:

```
objAeManager.CreateEventSubscription(intServerIndex, "test", True, 100, 100, 1,
RBufferTime, RMaxSize, indexsub, strError)
```

OPC CLIENT Sample

The chapter describes the DA, HDA and AE samples available within the installation of OPC .NET Client Toolkit.

1. Step 1: OPC Client User Interface

To use OPC DA, HDA and AE, launch the OPC Client Sample.

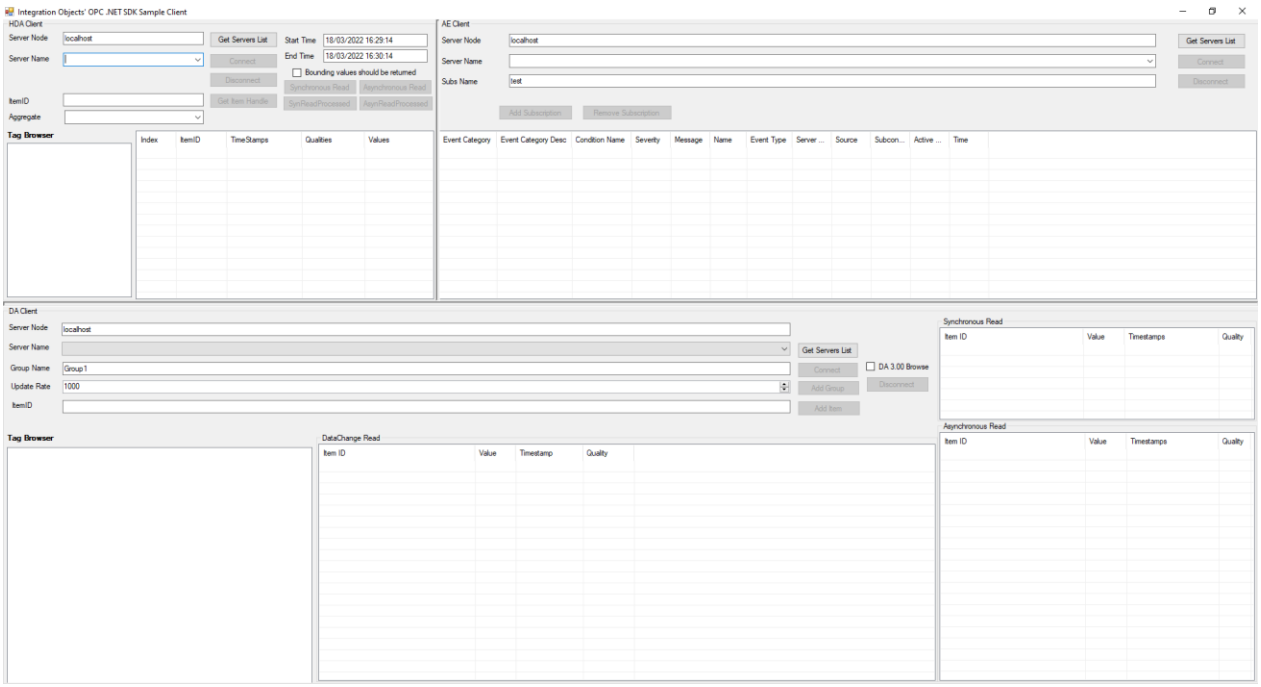

Figure 17: OPC Client - User Interface Window

2. Step 2: List All Available DA Servers

Click the **Get Servers List button** and select the server name to use from all available DA servers on the network.

3. Step 3: Connect and browse

You can browse the address space including all the branches and items for any OPC DA Server that supports OPC DA browsing. To browse your OPC server, check **DA 3.00 Browse Checkbox** if you will use OPC DA version 3.00 and Click **Connect button**.

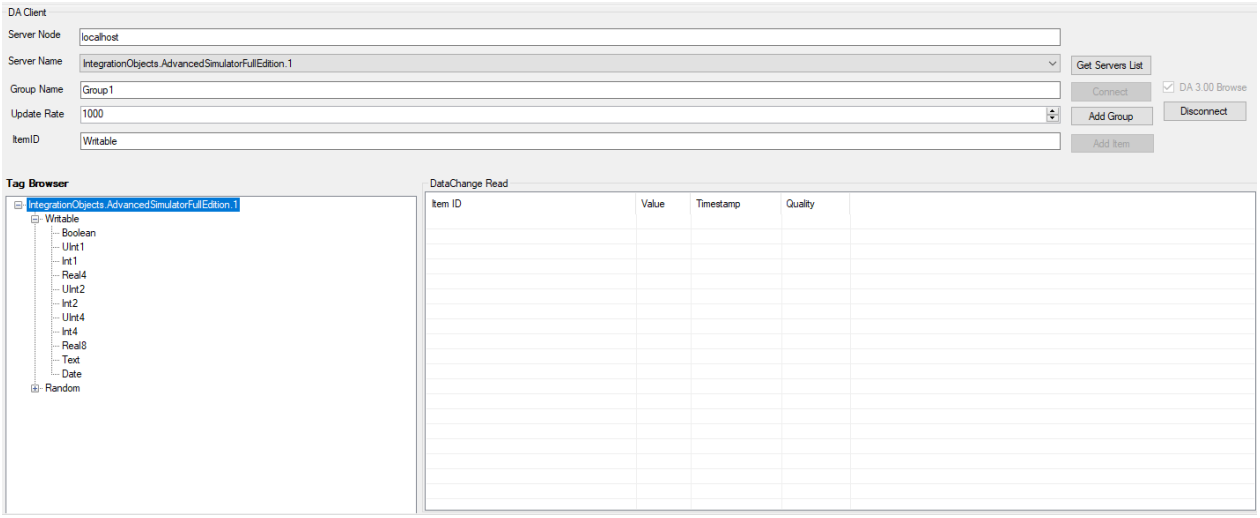

Figure 18: OPC Client – Connect and browse

4. Step 4: Read, Write and Get Properties

Right click on the **Tag Name** to read, write or get properties using specification 3.00.
Click **Read I/O3 button** to read tag value, Timestamp and Quality.

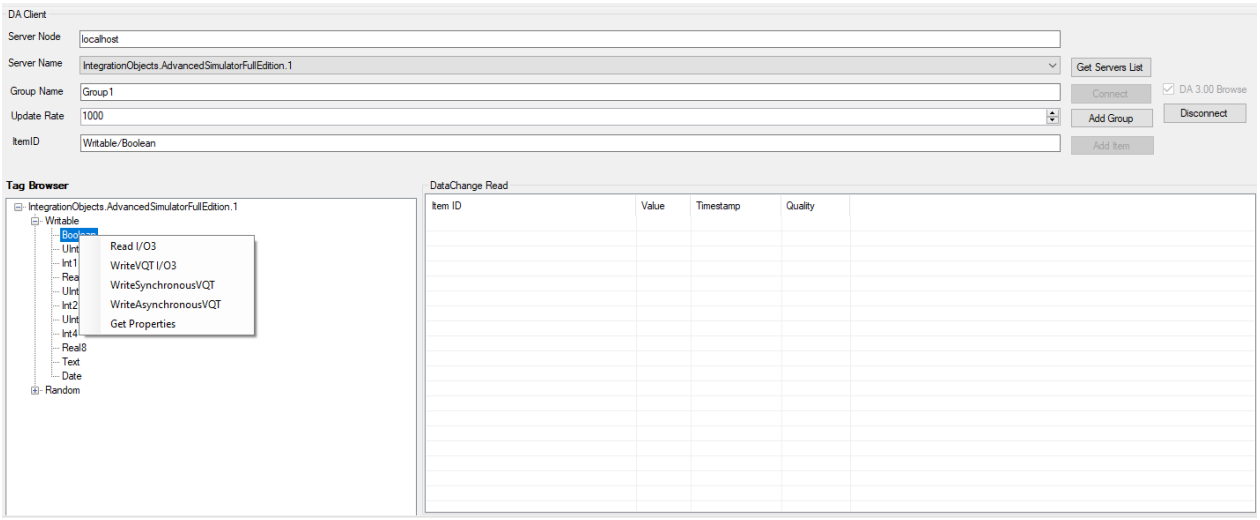

Figure 19: OPC Client – Read, Write and Get Properties

TOOLKIT TRACING CAPABLITIES

The toolkit has tracing capabilities. Developers can record the toolkit errors and debugging information (COM and OPC messages) in a log file named `OPCNetClientSDKLog.LOG`. If difficulties occur with the toolkit, the log file can be extremely valuable for troubleshooting.

This log file is generated at start-up where the server executable file is located. The toolkit incorporates a configuration file `ConfigOPCClientSDK.ini` that 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 `ConfigOPCClientSDK.ini` in a text editor.
2. Edit any of the parameters listed in the following tables:

Log Setting	Description	Default Value
CreateNew	true to create a new event log or to append the old log, by default it's true	true
Level	There are five log levels: 1. Control: logs only control messages generated by the .NET Client Toolkit. 2. Error: Logs error and control messages generated by .NET Client Toolkit. 3. Warning: Logs warning, error and control messages generated by .NET Client Toolkit. 4. Inform: Logs information, warning, error and control messages generated by the .NET Client Toolkit. 5. Debug: Logs all messages generated by the .NET Client Toolkit. The higher the log level, the more information is recorded. We recommend using level “Control” for a better performance of the Toolkit. The other levels are dedicated for troubleshooting purposes.	Error

Source	The Window event log source name	OPCNetClientSDK
LogName	The OPC .NET Client Toolkit log file name	OPCNetClientSDKLog
AutoAppend	Set to true to continue writing log messages in the existed log file or to false to create a new file.	True
BufferSize	The maximum number of messages to be stored in the runtime memory before launching a write action to the hard disk. The specified value must be greater than 100.	100
MaximumFiles	Set to 0 means that log files will be created in an unlimited way.	0
PacketIntegrityEnabled	**True:** the activation authentication level = RPC_C_AUTHN_LEVEL_PKT_INTEGRIT. This is the configuration required to support the “DCOM network Packet Level Integrity Security Change”. **False:** the activation authentication level = RPC_C_AUTHN_LEVEL_DEFAULT.	false

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

Sample Configuration File:

```
[LogConfiguration]
CreateNew=True
Level=Error
LogName=OPCNetClientSDK
Source=OPCNetClientSDK
[FileLogConfiguration]
AutoAppend=False
BufferSize=100
FileName=OPCNetClientSDKLog
MaximumFiles=0
[DCOMSecurityConfiguration]
PacketIntegrityEnabled=false
```

GLOSSARY

C

CLSID

A CLSID is a globally unique identifier that identifies a COM class object. If your server or container allows linking to its embedded objects, you need to register a CLSID for each supported class of objects.

COM

Component Object Model is a specification for writing reusable software components. COM is an infrastructure that allows objects to communicate between processes and computers.

D

DCOM

Distributed Component Object Model is a protocol that enables software components to communicate directly over a network in a reliable, secure, and efficient manner.

O

OPC

Ole for Process Control is based on Microsoft's OLE/COM technology. OPC is open connectivity in industrial automation and the enterprise systems that support industry. Interoperability is assured through the creation and maintenance of open standards specifications.

P

ProgID

A ProgID or Programmatic Identifier.
The format of a ProgID is <Program>.<Component>.<Version>, separated by periods and with no spaces. Like the CLSID, the ProgID identifies a class but with less precision because it is not guaranteed to be globally unique.

TROUBLESHOOTING

1. Problem 1: Access Denied Error

You have an "Access denied" error on the client machine. The client and server are not part of the same domain.

Let's assume that the OPC client is running on machine A and the OPC server on machine B. When the OPC client and server are on different computers, you have to give each computer access to the other by giving access permissions. The permission issue is crucial to proper DCOM configurations.

Here the server is running on a standalone machine. So the ONLY user accounts it will trust are those it finds in its own "local" security database. Here is where you can get into trouble in setting up an OPC client to server connection.

To allow remote client to access the DCOM server, the DCOM utility uses the Windows Security database. For this reason, you cannot give access to a user account that is not found in this database.

Here is the issue:

1. You can add Machine B into the same domain as Machine A (or in a trusted domain), which is the safest way to correctly set up the communication between the OPC client and the OPC server.
2. You need to create the *exact same* user account name AND password on BOTH machines (for example User1 (login), PWD1 (password)). Once you have that set up, when Machine A calls on Machine B with an OPC request and identifies himself as User1 with PWD1 password, Machine B will look in its database, see the same account name, the same password, and same "come on in request from Machine A". When Machine B returns its data from the OPC server to the OPC client on machine A, the OPC server will go call Machine A as User1 with a password – Machine A will look in its database, see that it has that account, and accept the call. This workaround should resolve the communication problem between the OPC client and server.

2. Problem 2: Operating System Upgrade

You have been running your OPC client on a Windows XP machine. When upgrading the machine to XP Service Pack 2, the OPC client becomes unable to connect to the OPC server.

This is a common problem when using OPC via DCOM with Microsoft Windows XP Service Pack 2. In fact, when Service Pack 2 is installed with its default configuration settings, OPC communication via DCOM will cease to work.

To resolve this issue, reconfigure your settings for:

1. The windows XP firewall
2. DCOM

You can find the OPC Foundation document named "**Using OPC via DCOM with SP2.pdf**" that describes all the steps needed to apply new settings under the OPC NET Client Toolkit installation folder.

3. Problem 3: "This is not a development machine" Error

You have a full version of the toolkit and when you run the application on the runtime machine, the following error message is prompted: "This is not a development machine"

In order to run your application properly using a full runtime version of the Toolkit, make sure that the OPC .NET Client Toolkit is not installed as a demo version in the deployment machine. If it is the case, you will need to uninstall it.

Also, verify the following criteria are met:

- The path of the application folder does not include the words "Debug" or "Release".
- The application deployment folder should contain the following files:
  - ConfigOPCClientSDK.ini
  - IntegrationObjects.Logger.SDK.dll
  - IntegrationObjects.OPCNetClientSDK.dll
  - License.dll
  - The application executable and any other custom depending assembly

4. Problem 4: "This is not a valid license" Error

When you run the application, the following error message is prompted: "This is not a valid license".

Open the license authorization tool and check the license status. In case the license is valid, check that the License.dll exists in your application output folder.

5. Problem 5: Unable to Connect to Local OPC Server

You are not able to connect to a local OPC Server.

You should check if the OPC Core Components are installed in your machine.
The OPC Core components to be installed depend on the target platform of your application.
The installation program is located in the installation folder under:
..\Integration Objects\Integration Objects' OPC NET Client Toolkit\Components

6. Problem 6: Unable to run the OPC Client as Windows service:

You are not able to run the OPC Client Sample as Windows service.
The current directory of a Windows service is set to ".:\WINDOWS\system32" by default. You need to set the current directory to the directory where your application files are located in order to load the required files of the OPC Client application.

Verify the following items to be able to run the OPC Client Sample as Windows service:

- Add the line below before initializing the OPCDAManager/OPCHDAManager/ OPCAEManager:
  `Directory.SetCurrentDirectory(AppDomain.CurrentDomain.BaseDirectory);`
- Verify that you are running your OPC Client application service with a user account that has access rights to connect to the OPC Server and read data.

customerservice@integrationobjects.com

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