# Integration Objects'

## OPC DA/DX/HDA Toolkit
## For Server Application Development

**OPC Server Toolkit**
Version 3.0 Rev.0

# USER GUIDE

Integration Objects' OPC Server Toolkit User Guide Version 3.0 Rev.0
Published December 2017

# TABLE OF CONTENTS

# TABLE OF FIGURES

# LIST OF TABLES

# PREFACE

## About This User Guide

This guide:

- Presents the OPC Server Toolkit,
- Explains how to use this API (**A**pplication **P**rogramming **I**nterface) to generate out-of-process server specific applications,
- Describes in detail the functionalities exported by this toolkit and explains returned error codes.

## Target Audience

This user guide is intended for developers of OPC DA/DX/HDA compliant servers. It assumes that developers have a working knowledge of programming with Visual C++, Visual Basic, Visual C# .NET, and Visual Basic .NET languages. It also assumes that they have an idea about the latest OPC Data Access, Data eXchange, and Historical Data Access specifications.

## Related Documents

This user guide is based on the following OPC specifications:

- OPC Common Definitions and Interfaces 1.0
- OPC Data Access Custom Interface Standard 1.0a
- OPC Data Access Custom Interface Standard 2.05
- OPC Data Access Custom Interface Standard 3.0
- OPC Historical Data Access Standard 1.1
- OPC Historical Data Access Standard 1.2
- OPC Data eXchange Standard 1.0

# Customer Support Service

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

# OVERVIEW OF THE OPC SERVER TOOLKIT

## 1. Overview

Integration Objects' OPC Server Toolkit is a Windows DLL (Dynamic Link Library) that implements all required and most of the optional OPC Data Access interfaces, OPC Data eXchange interfaces, and OPC Historical Data Access interfaces. Refer to the list of supported OPC interfaces. It is a tool for fast and easy programming of OPC DA/DX/HDA servers.

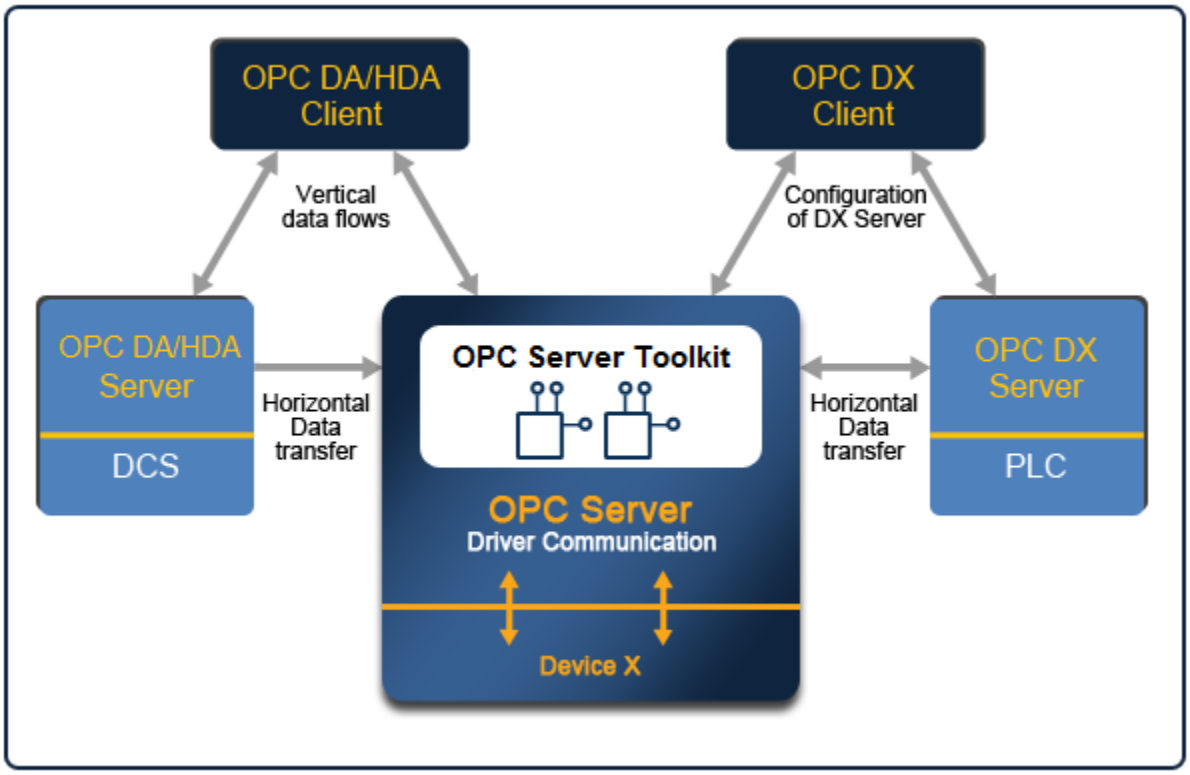


**Figure 1: Deployment of the OPC Server Toolkit in an OPC Server**

Any OPC server based on this API can be accessed locally or remotely via DCOM by OPC DA/DX/HDA clients. It can be configured by any OPC DX client using Data eXchange services and can be supervised and controlled by an OPC DA client.

The OPC Server Toolkit is licensed per development machine and is royalty-free for the runtime

# 2. OPC Server Toolkit Capabilities

All OPC DA/DX/HDA servers share an important part of their code. This part covers COM/DCOM communications and the implementation of OPC DA/DX/HDA interfaces. The OPC Server Toolkit was created and designed in order to hide the complexity of handling these operations and provide an easy and flexible interface for OPC server specific applications development.

**The OPC Server Toolkit handles a maximum number of 100000 tags.**

## 2.1. Registering/Unregistering the OPC server

Giving all the necessary information such as server name (known as PROGID) and server CLSID, this toolkit is able to add/remove COM entries in the Windows registry for the OPC server, and so allowing developers to use this API with minimal knowledge of COM/DCOM.

Note that an OPC server is a COM server, so it needs to be registered in the Windows Registry to allow OPC clients to identify it.

## 2.2. Providing OPC Data Access Services

Using OPC Server Toolkit, OPC server application can communicate with OPC DA compliant clients. In fact, the toolkit implements all OPC Data Access functionalities including synchronous/asynchronous reads and writes from/to the cache/device. The module that manages the cache is encapsulated into the DLL. Whereas, whenever the server needs to poll the equipment for device transactions, it just invokes simple callbacks functions offered by this API. These callbacks have been designed to handle the communication with the real equipment.

Building and browsing the DA hierarchy of the server address space are treated internally. The server has, by default, a hierarchical namespace that it constructs using the delimiter provided in its initialization phase.

## 2.3. Providing OPC Data eXchange Services

### 2.3.1. Overview of OPC Data eXchange Standard

In addition to capabilities provided by a DA server module, an OPC DX server has an OPC-DA client part with a DX interface. This client module consumes data from other OPC DA/DX servers known as **source servers** and performs updates to the target tags (DA items of your DX server). That being said, a DX server is

similar to a data pipeline from source servers to your system. This kind of data transfer from a source item to a target item is called a **DXConnection**.

A **DXDatabase** stores all information about configured DXConnections, source servers, and DXServerStatus.

- DXServerStatus: Includes the maximum number of supported DXConnections, the maximum queue size that holds the DX write actions to target items, and the list of supported COM DA servers.
- DXConnection: Includes information like target item name (a tag that we want to update), source server name (the OPC DA/DX server from which we want to collect data), and source item name.

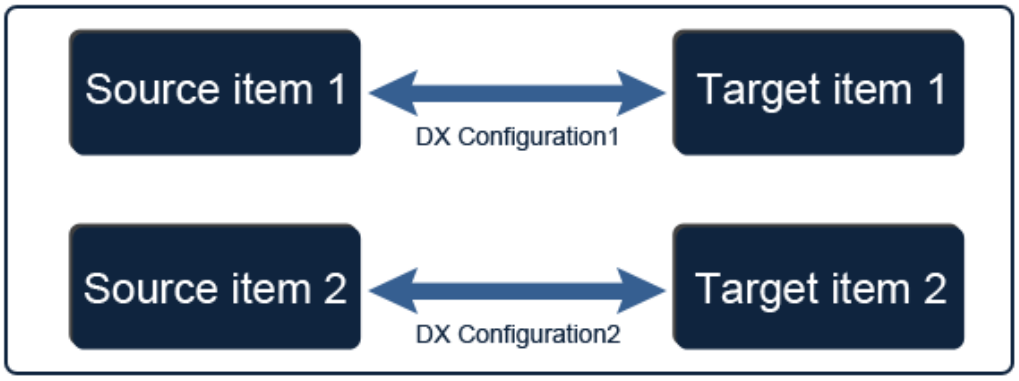


**Figure 2: DX Configuration**

- Source Servers: Contains information about configured OPC servers that are OPC DA 2.0 compliant.

### 2.3.2. Toolkit Features

The toolkit encapsulates all OPC Data eXchange services. DX clients can configure the server to allow data transfer from other OPC DA/DX servers to itself (and from/to itself for internal connections).

The DXDatabase stores configuration information and is managed by this toolkit as an XML document (file). Its manipulation is performed internally. Developers don't need great knowledge to manipulate XML documents. This function is delegated to the DLL.

**Item IDs and Browsing**

The OPC DX server has a hierarchical namespace. It uses a reserved subtree of its address space for DX server items. In the server address space, you will find an additional default branch named "DX". Under this branch, there is, at minimum, the "DXServerStatus" branch that describes the server status.

Using configuration services (add/remove/update DXConnection/SourceServers), you may also find the "DXConnectionRoot" branch and "SourceServers" branch under the "DX" root branch.

In case the server will support DX features, it is imperative to use the slash ('/') as a path delimiter for all items including DA items, according to the OPC Foundation. Nevertheless, in such cases, we may encounter some inconvenience because item names can also contain the slash as a basic character.

To resolve this issue, the developer should make character replacements in the item names to be in accordance with the DX specification before calling any routine to add tags to the server address space. That means items' names are passed to the DLL with '/' as a separator after all necessary modifications (① ).
The DLL will expose Item IDs with these modifications to the server code which should reverse replacements made initially to regenerate true Items paths known by the system (② ).

To clarify this reasoning, let's take the following example. The problem occurs when the server supports DX features and the OPC server developer chooses a delimiter for tags that differs from slash character. Let's assume that there are some Item IDs that contain '/' and the path separator of the vendor is the '#' character.

A tag will have the following fully qualified ItemID syntax (for example):

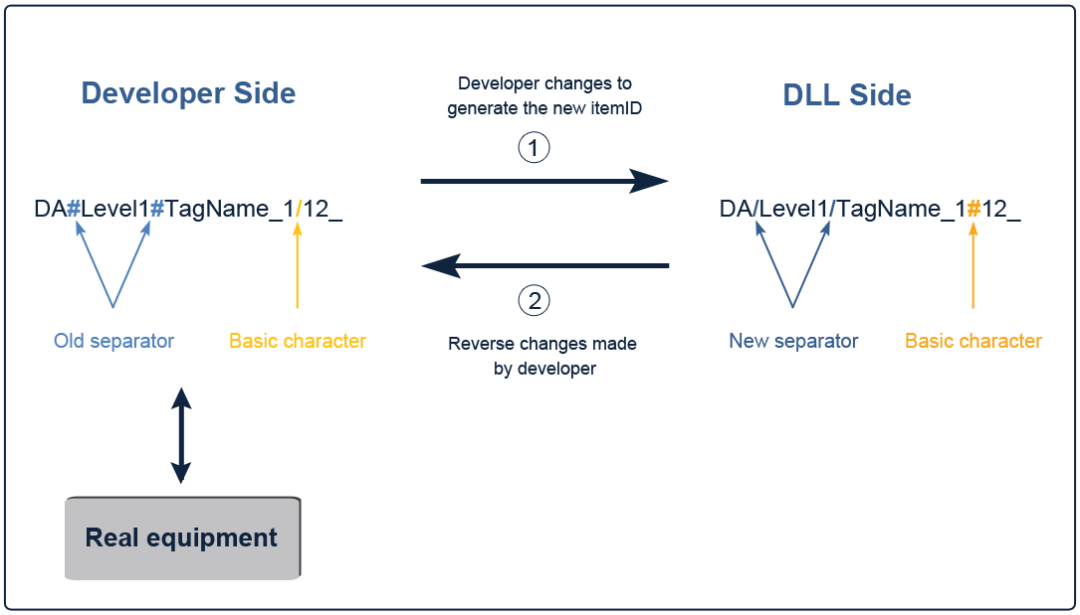


**Figure 3: Item ID Syntax**

The '**/**' character is used to separate levels in the OPC ItemID.

The '#' character became like any other character in the ItemID.

All these changes are made outside the DLL. The developer of this OPC server should implement a piece of code that performs all required character replacements for all communications from/to real equipment.

If the separator passed to the DLL is not '/', then a flat namespace would be imposed.

**Providing Data Transfers from OPC servers**

Once the server is configured, the toolkit begins the transfer of data. It offers an interface to the driver to manage updated values. New values should be written to the device.

## 2.4. Providing OPC Historical Data Access Services

If developers choose to implement an OPC HDA server using this toolkit, they will find it very easy as most of the HDA server functionalities are encapsulated into the DLL.

The server will be able to:

- Build the OPC HDA hierarchy of the server namespace.
- Collect historical data from the data source.
- Calculate aggregates over a time range, such as average, minimum, maximum, etc. Most of the standard aggregates, defined by the OPC Foundation, are implemented inside the DLL but developers can add their specific aggregates if needed (refer to io_hdasetitemaggregates function description for more details).
- Perform synchronous/asynchronous reads of attributes' values and their timestamps from the history database for a specified time domain and a given item. Like aggregates, developers can add their own specific attributes.
- Insert, replace, or delete entries of the historical database.

**During server implementation, developers will only need to focus on device communication.**

# 3. OPC Server Toolkit Architecture

This API offers a set of functions that OPC server programs can use to handle OPC Client requests.

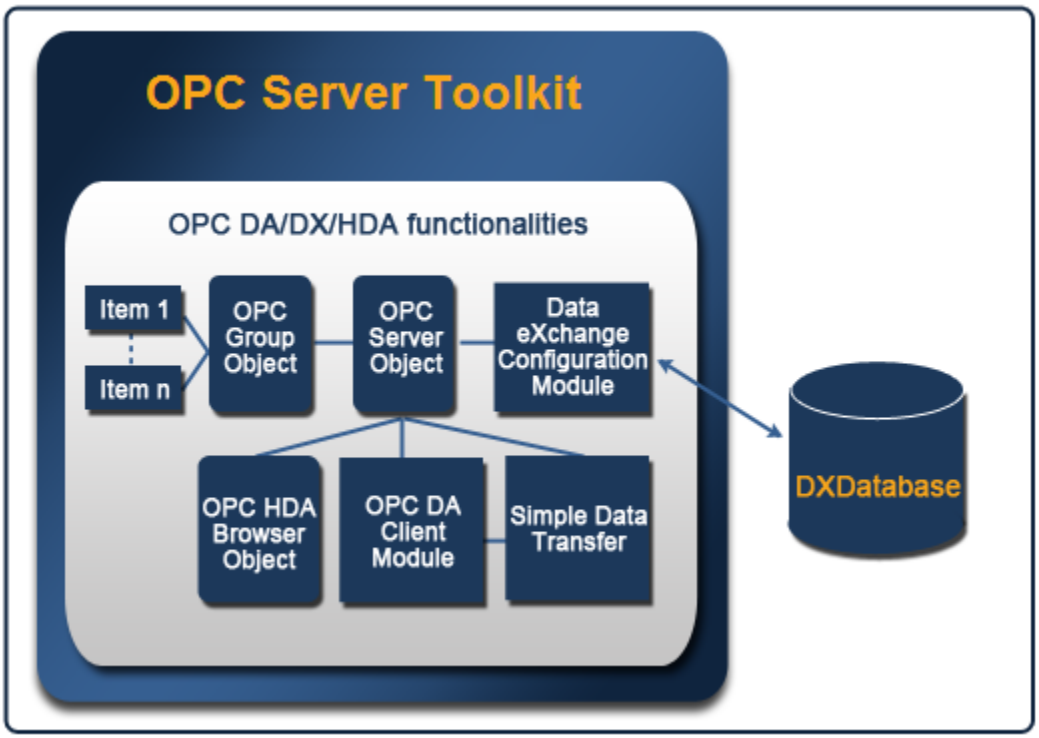


**Figure 4: OPC Server Toolkit Architecture**

The OPC Server Toolkit includes the following main modules: OPC Server Object, OPC Group Object, OPC DA Client, Data eXchange configuration, OPC HDA Browser Object, and Simple Data Transfer module.

The configuration module interacts with the DXDatabase to update the lists of source servers and DXConnections and to update the DXServerStatus.
The OPC DA Client module allows the DX server to send requests to OPC source servers and begin the data transfer by making a simple copy of source values.

# 4. Supported OPC Interfaces

The OPC Server Toolkit is compliant with the latest released specifications. It supports OPC Data Access (server side) 1.0, 2.05, and 3.0 specifications; OPC Data Access (client side) 1.0 and 2.05, OPC Data eXchange 1.0 and OPC Historical Data Access 1.1 and 1.2.

The supported interfaces are listed in the tables below:

**OPC Data Access interfaces version 1.0**

| Data Access Server Required Interfaces | 1.0 | Implemented |
|---|---|---|
| OPCServer | | |
| IUnknown | Required | Yes |
| IOPCServer | Required | Yes |
| IOPCServerPublicGroups | Optional | No |
| IOPCBrowseServerAddressSpace | Optional | Yes |
| | | |
| OPCGroup | | |
| IUnknown | Required | Yes |
| IOPCItemMgt | Required | Yes |
| IOPCGroupStateMgt | Required | Yes |
| IOPCPublicGroupStateMgt | Optional | No |
| IOPCSyncIO | Required | Yes |
| IOPCAsyncIO | Required | Yes |
| IdataObject | Required | Yes |

**Table 1: OPC DA Interfaces Version 1.0**

**Added OPC Data Access interfaces version 2.05**

| Data Access Server Required Interfaces | 2.0 | Implemented |
|---|---|---|
| OPCServer | | |
| IOPCCommon | Required | Yes |
| IConnectionPointContainer | Required | Yes |
| IOPCItemProperties | Required | Yes |
| | | |

| | | |
|---|---|---|
| OPCGroup | | |
| IOPCAsyncIO2 | Required | Yes |
| IConnectionPointContainer | Required | Yes |

**Table 2: OPC DA Interfaces Version 2.05**

**Added OPC Data Access interfaces version 3.0**

| Data Access Server Required Interfaces | 3.0 | Implemented |
|---|---|---|
| OPCServer | | |
| IOPCBrowse | Required | Yes |
| IOPCItemIO | Required | Yes |
| | | |
| OPCGroup | | |
| IOPCSyncIO2 | Required | Yes |
| IOPCAsyncIO3 | Required | Yes |
| IOPCItemDeadbandMgt | Required | Yes |
| IOPCItemSamplingMgt | Optional | No |
| IOPCGroupStateMgt2 | Required | Yes |

**Table 3: OPC DA Interfaces Version 3.0**

**OPC Data Access Custom Interface (Client side)**

The DX server is also an OPC DA client version 2.05.

| Data Access Client | Implemented |
|---|---|
| IOPCDataCallback | Yes |
| IOPCShutDown | Yes |

**Table 4: OPC DA Custom Interface**

**OPC Data eXchange interfaces version 1.0**

| Data eXchange Server | 1.0 | Implemented |
|---|---|---|
| IOPCConfiguration | Required | Yes |

**Table 5: OPC DX Interfaces Version 1.0**

**OPC Historical Data Access interfaces version 1.1 and 1.2**

| Historical Data Access Server Required Interfaces | 1.1/1.2 | Implemented |
|---|---|---|
| OPCHDAServer | | |
| IOPCCommon | Required | Yes |
| IOPCHDA_Server | Required | Yes |
| IOPCHDA_SyncRead | Required | Yes |
| IOPCHDA_SyncUpdate | Optional | Yes |
| IOPCHDA_SyncAnnotations | Optional | No |
| IOPCHDA_AsyncRead | Optional | Yes |
| IOPCHDA_AsyncUpdate | Optional | Yes |
| IOPCHDA_AsyncAnnotations | Optional | No |
| IOPCHDA_Playback | Optional | No |
| IConnectionPointContainer | (required if any Async interface is supported) | Yes |
| | | |
| OPCHDABrowser | | |
| IOPCHDA_Browser | Required | Yes |

**Table 6: OPC HDA Interfaces Version 1.1 and 1.2**

This toolkit does not support the HDA annotations and playback interfaces, which are optional interfaces.

# 5. Supported Types

Integration Objects' OPC Server Toolkit supports the following simple VARIANT data types:

VT_I1
VT_UI1
VT_I2
VT_UI2
VT_I4
VT_UI4
VT_R4
VT_R8
VT_DATE
VT_BSTR
VT_BOOL

# 6. Supported Item Properties

The following is the set of standard property IDs supported by Integration Objects' OPC Server Toolkit. For more details, see Appendix A.

| ID | DATATYPE of returned VARIANT | STANDARD DESCIPTION |
|---|---|---|
| 1 | VT_I2 | "Item Canonical DataType" (VARTYPE stored in an I2) |
| 2 | <varies> | "Item Value"<br>(VARIANT)<br>Note: the type of the returned value is, as indicated by the "Item Canonical DataType", above and depends on the item. This will behave like a read from DEVICE. |
| 3 | VT_I2 | "Item UshoQuality"<br>(OPCUSHOQUALITY stored in an I2). This will behave like a read from DEVICE. |
| 4 | VT_DATE | "Item Timestamp"<br>(Will be converted from FILETIME). This will behave like a read from DEVICE. |
| 5 | VT_I4 | "Item Access Rights"<br>(OPCACCESSRIGHTS stored in an I4) |

| | | |
|---|---|---|
| 6 | VT_R4 | "Server Scan Rate"<br>In Milliseconds. This represents the fastest rate at which the server could obtain data from the underlying data source. The nature of this source is not defined but is typically a DCS system, a SCADA system, a PLC via a COMM port or network, a Device Network, etc. This value generally represents the 'best case' fastest RequestedUpdateRate which could be used if this item were added to an OPCGroup.<br>The accuracy of this value (the ability of the server to attain 'best case' performance) can be greatly affected by system load and other factors. |
| 7 | VT_I4 | "Item EU Type"<br>(OPCEUTYPE stored in an I4) |
| 8 | VT_BSTR \| VT_ARRAY | "Item EUInfo"If item 7 "Item EU Type" is "Enumerated" - EUInfo will contain a SAFEARRAY of strings (VT_ARRAY \| VT_BSTR) which contains a list of strings (Example: "OPEN", "CLOSE", "IN TRANSIT", etc.) corresponding to sequential numeric values (0, 1, 2, etc.) |

**Table 7: Supported Item Properties**

# 7. System Compatibility

## 7.1. Operating System Compatibility

This toolkit supports the following operating systems:

- Windows XP SP1, SP2, SP3
- Windows Server 2003
- Windows 7
- Windows 8 and 8.1
- Windows 10
- Windows Server 2008
- Windows Server 2012

## 7.2. Visual Studio Compatibility

The supported visual studio versions are:

- Visual Studio 6.0
- Visual Studio 2005
- Visual Studio 2008
- Visual Studio 2010
- Visual Studio 2012

- Visual Studio 2013
- Visual Studio 2015
- Visual Studio 2017

# GETTING STARTED

## 1. Pre-installation Considerations

In order to properly run an OPC DA/DX/HDA server implemented using this toolkit, you should install these software components on the target system:

- Microsoft .NET Framework (Microsoft .NET Framework Version 2.0 or later Redistributable Package)
- OPC Core Components 3.00 that include all shared OPC modules such as the DCOM proxy/stub libraries, the OPC Server Enumerator, .NET wrappers, etc.

**All these components are mandatory to install, if you plan to develop OPC servers' applications supporting OPC Data Access, OPC Data eXchange, and OPC Historical Data Access.**

**If you plan to develop OPC DA Server or OPC HDA Server only using this toolkit and an unmanaged language (C++ or VB6), there is no need to install the Microsoft .NET Framework.**

## 2. Installation

To install the OPC Server Toolkit, proceed to the following steps:

1. Launch the setup package with a user account that has administrator privileges (run as administrator),
2. Click the "Next" button,

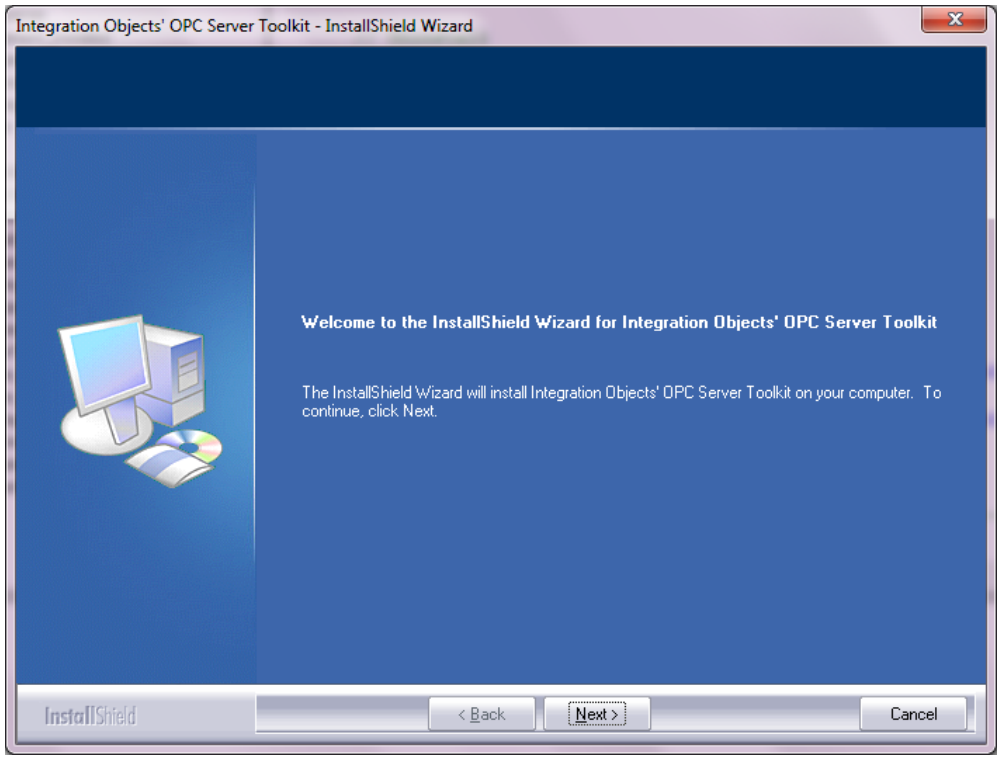


**Figure 5: Welcome Dialog Box**

3. Accept the license agreement terms, then click the "Next" button,

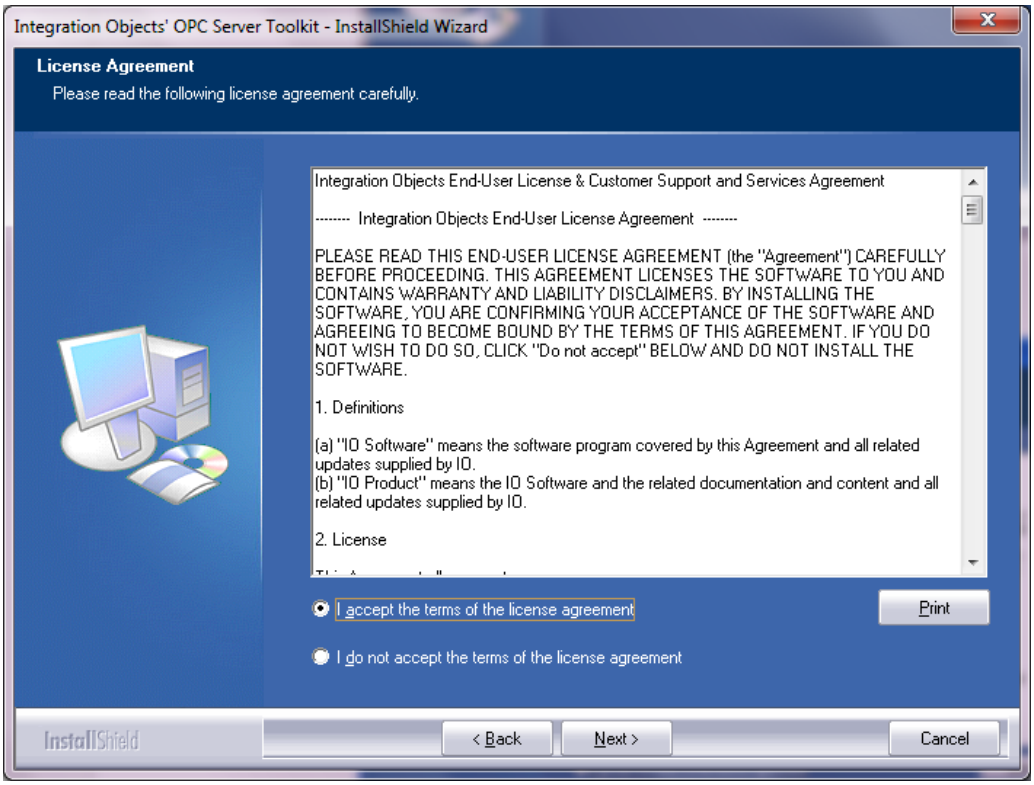


**Figure 6: License Agreement Dialog Box**

4. Enter the User Name and the Company Name,

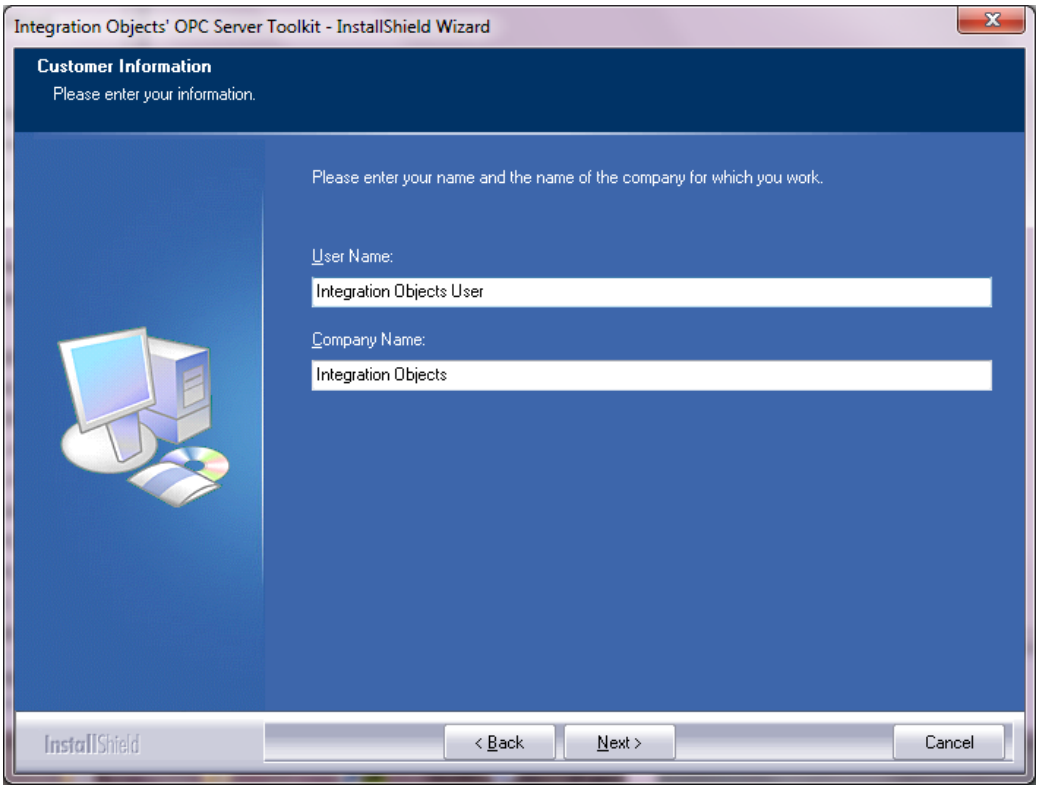


**Figure 7: Customer Information Dialog Box**

5. The setup contains two setup types. You need to select one of them:
    a. **Full Version**: This setup type is intended for a licensed development machine.
    b. **Demo Version:** The package is limited to run only 2 hours for a period of 16 days.

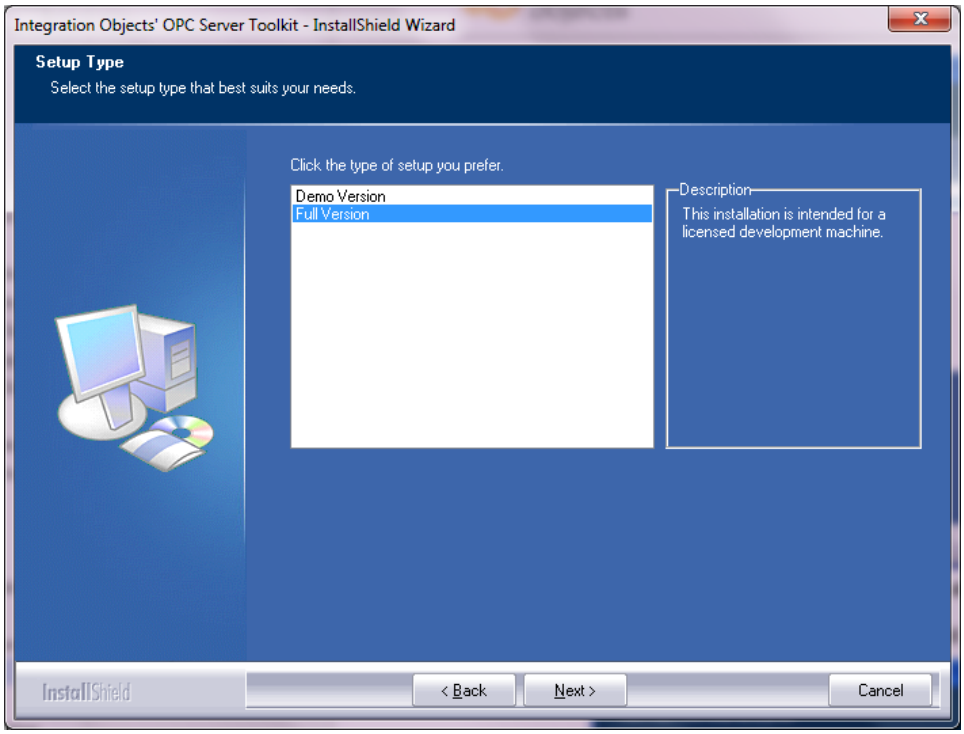


**Figure 8: Setup Type Dialog Box**

6. Select the path where the OPC Server Toolkit will be installed, then click "Next",

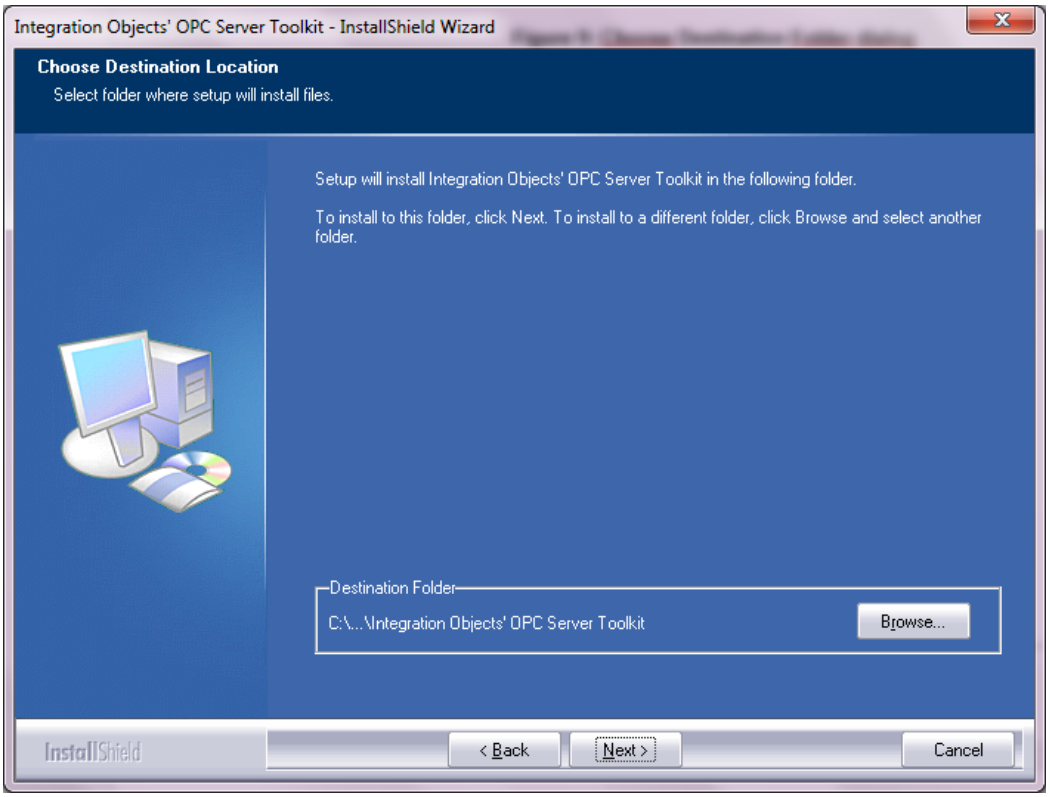


**Figure 9: Choose Destination Folder Dialog Box**

7. Click the "Install " button,

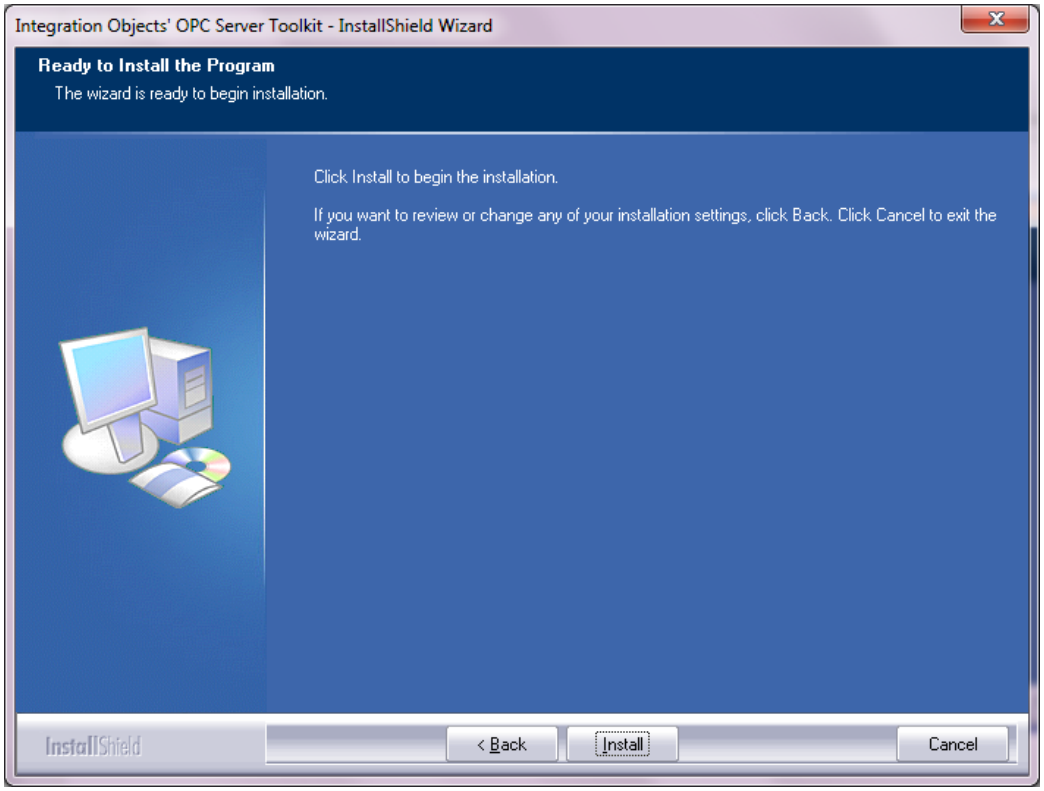


**Figure 10: The Install Dialog Box**

8. The setup is now copying all the files,

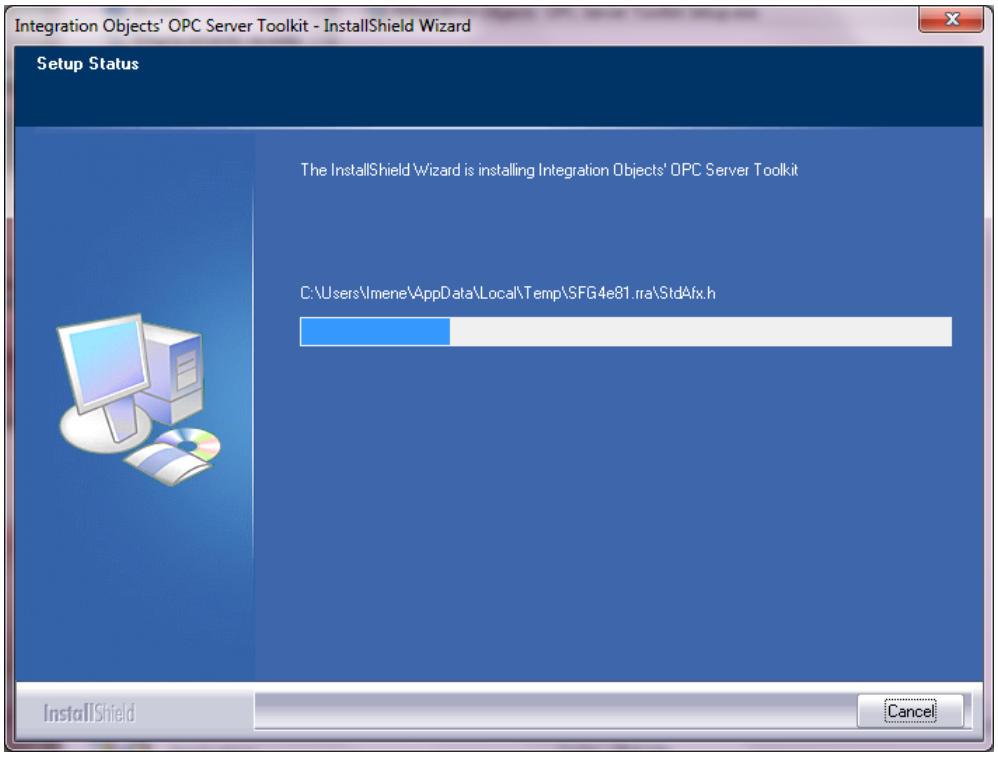


**Figure 11: The Installation Progress Dialog Box**

9. If you have selected the Full Version setup type, you will get the following dialog box for the license activation. Copy the "User ID" then send it to Integration Objects' sales team (sales@integrationobjects.com) and they will get back to you with the "Activation Code".

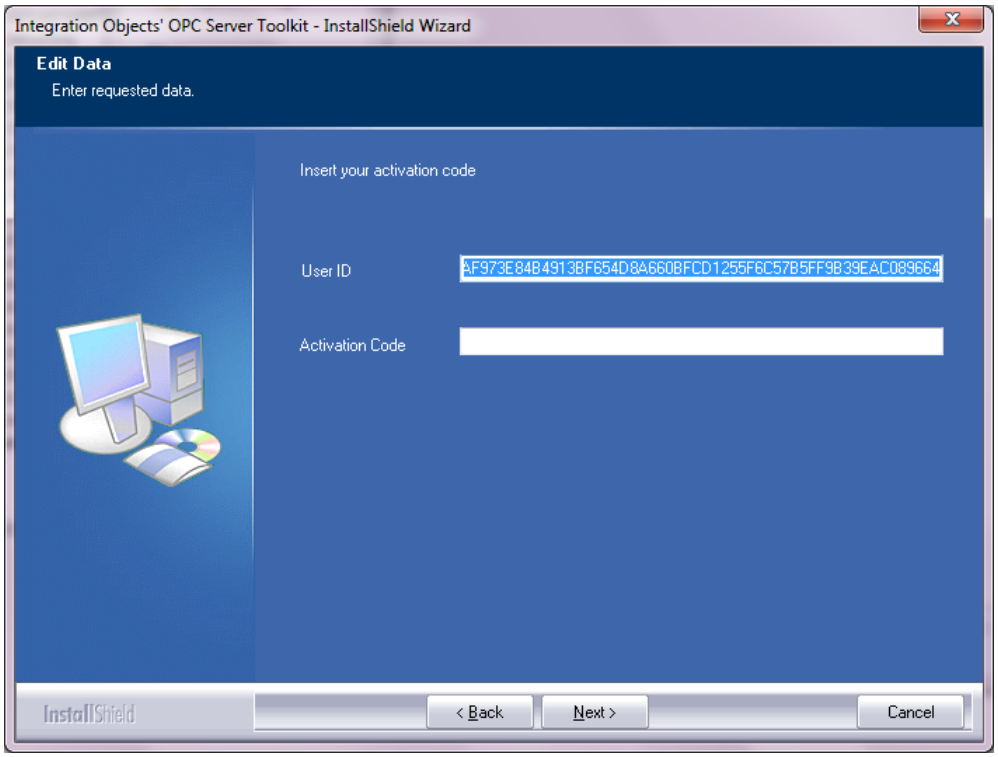


**Figure 12: License Activation Dialog Box**

10. Once you enter a valid activation code and click "Next", you will get the following confirmation message:

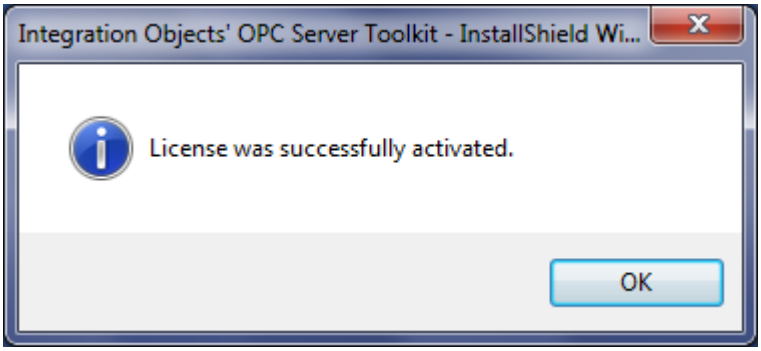


**Figure 13: License Activation Confirmation Message Box**

11. Click the "Finish" button. Now, the OPC Server Toolkit is properly installed in your machine.

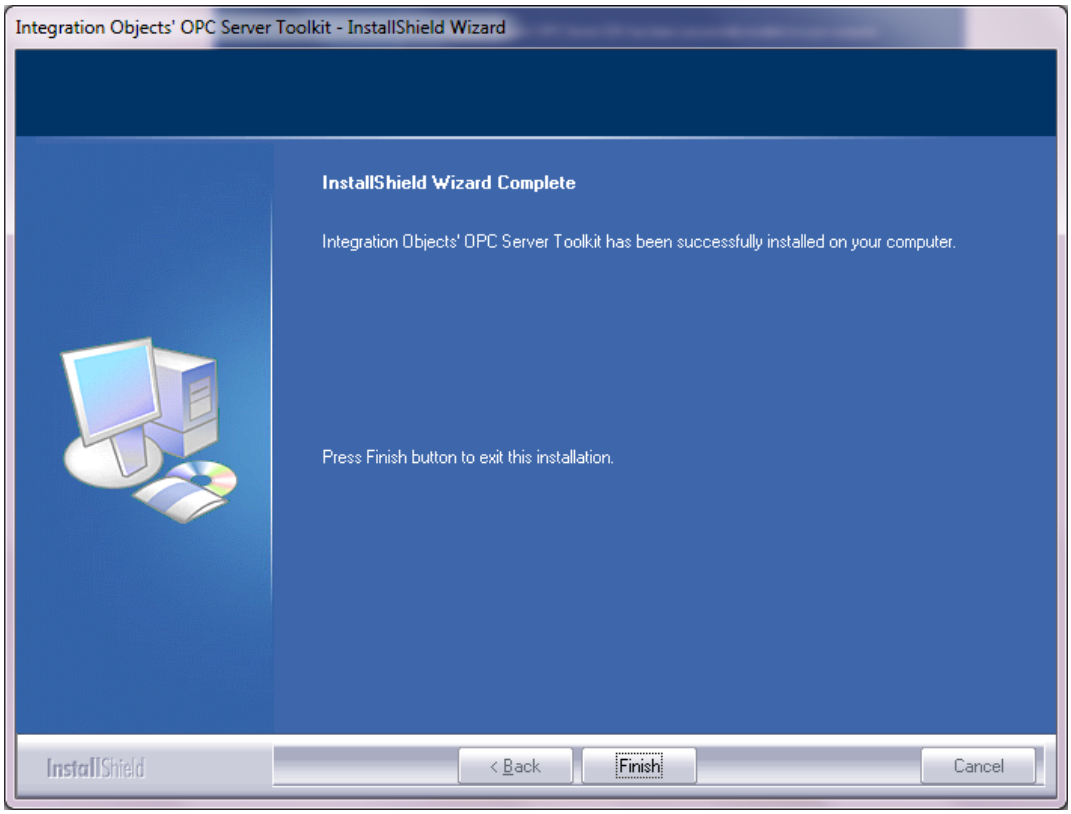


**Figure 14: Installation Completed Dialog Box**

# 3. Uninstallation

To uninstall the OPC Server Toolkit, you have two options:

1. From the Programs Menu, select the "Uninstaller" shortcut:

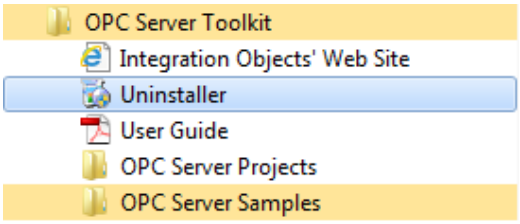


**Figure 15: Uninstaller Icon in the Start Menu**

The uninstall wizard will be automatically launched and will take you through the different steps of the software removal.

2. From Add/Remove Programs:
    a. Go to the start menu
    b. Select Settings and click on Control Panel.
    c. Choose Integration Objects' OPC Server Toolkit from the programs' list.

d. Click on Remove button

# 4. Files Included in the Distribution

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

| Folder | Files and Description |
|---|---|
| Lib | **DXServerDll.lib**: Contains the export definitions for this toolkit. |
| | |
| Include | **DXServerDll.h**: Includes all exported constants, structures, and signatures of methods and functions. It represents the interaction between the server application and the toolkit.<br>**OPC Include Files:**<br>**Opcerror.h**: Provided by the OPC Foundation, defines error codes for the Data Access specifications.<br>**OpcHDA_Error.h**: Provided by the OPC Foundation, defines error codes for the Historical Data Access specifications. |
| | |
| | |
| DLLs | **DXServerDll.dll**: The toolkit DLL used to build your application server in unmanaged language environment.<br>**OPCDADll.dll**: Used for Data eXchange functionalities.<br>**OPCNetServerSDK:** The .NET toolkit DLL used to build your application server in managed language environment.<br>**License.dll:** core component. |
| | |
| Configuration files | **OpcDxServer.Config.xml**: The configurations file for OPC Data eXchange services.<br>**SrvToolkit_CfgFile.ini**: contains configurable parameters for logging. |
| | |
| OPC Server Samples | This folder contains 9 OPC DA/DX/HDA sample servers:<br>VS6 C++ Sample Server built using Visual C++ 6.0. |

| | |
|---|---|
| | VS2008 C++ Sample Server built using VS 2008.<br>VB6 Sample Server implemented in Visual Basic 6.0.<br>OPCServerSimulationC#2008 built using VS 2008.<br>OPCServerSimulationVB.NET2008 built using VS 2008.<br>OPCServerSimulationC++.NET2008 built using VS 2008.<br>OPCServerSimulationC#2010 Service built using VS 2010<br>VS2008 C++ Sample Server Service built using VS 2010.<br>OPCServerSimulationCS2010WPF building using VS 2010 |
| | |
| EXE | **OPCServerSimulatorVS6.exe**: An executable of an OPC server for simulation.<br>**OPCServerSimulatorVS2008.exe**: An executable of an OPC server for simulation.<br>**OPCVB.Simulator.1.exe**: An executable of an OPC server for simulation.<br>**OPCServerSimulationC++.Net2008.exe**: An executable of an OPC server for simulation.<br>**OPCServerSimulationCS2008.exe**: An executable of an OPC server for simulation.<br>**OPCServerSimulationVB.NET2008.exe**: An executable of an OPC server for simulation.<br>**OPCServerSimulationCS2010Service.exe**: An executable of an OPC server for simulation service.<br>**OPCServerSimulatorVS2008Service.exe**: An executable of an OPC server for simulation service.<br>**OPCServerSimulationCS2010WPF.exe**: An executable of an OPC server for simulation. |
| | |
| Documentation | **OPC Server Toolkit User Guide.pdf**: This user guide. |

**Table 8: Folders in Distribution**

## 4.1. DX Configuration File

Any OPC DX configuration tool - which is an OPC DX client - is able to discover the OPC DX server on the network, configure, and manage its resources, whether it is online or offline. Once configured, the server will maintain a local persistent database that is loaded during server start-up to establish pre-configured connections with other servers.

This OPC server stores its configuration information in an XML file (OpcDxServer.config.xml by default) that has the following format:

```
<?xml version="1.0" encoding="utf-16" ?>
<Config xmlns="http://opcfoundation.org/webservices/OPCDX/10"
  xmlns:xsi="http://www.w3.org/2001-XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001-XMLSchema">

<!—Registration Information-->
<!—OPC DX Server Status  -->
<!—List of source servers  -->
<!—List of DXConnections  -->

</Config>
```

**Figure 16: DX Configuration File**

The configuration file must at least contain the text stated above and should only be accessed by the configuration tool. It contains descriptions of the OPC DX server status, DXConnections and source servers. Users can edit this file using Internet Explorer to see how it can be structured. This file is managed by this DLL.

# 5. Compiling and Linking Applications

## 5.1. Grant Permission Tool

It is mandatory to grant the permission to the current user account to use the OPC Server Toolkit in the development machine. To do so, the end user has to follow the steps below:

- Run the GrantPermissionTool.exe located under the installation folder as an administrator
- Enter the domain and the user name then click on the *Grant Permission* button

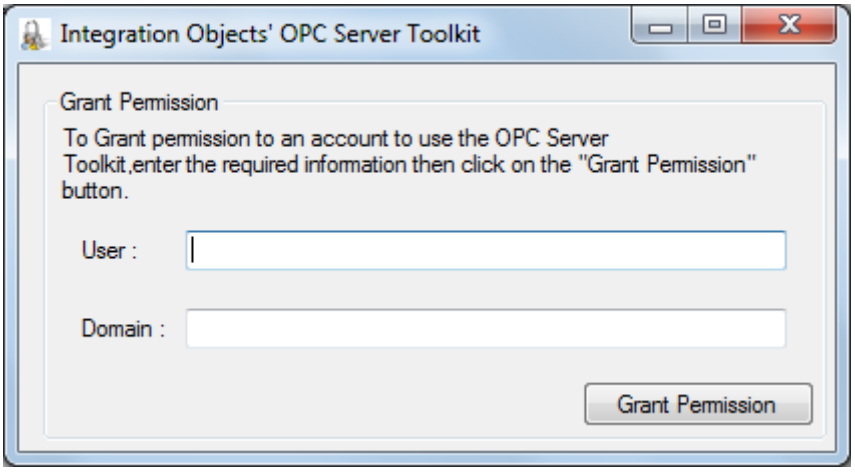


**Figure 17: Grant Permission Tool**

## 5.2. Visual Studio 6.0

To successfully create your server application using Visual Studio 6, you have to:

- Include the **DXServerDll.lib** file with the project files for the custom application.
- Include, in your source code project, the following files:
  - DXServerDll.h
  - Opcerror.h
  - OpcHDA_Error.h if your server application supports the HDA feature.
- Add the files: License.dll, OPCDADLL.dll, DXServerDll.dll, SrvToolkit_CfgFile.ini and OpcDxServer.Config.xml to your output project folder (where your executable server will be generated).

## 5.3. Visual Studio 2005, 2008, 2010, 2012, 2013, 2015 and 2017

To successfully create your server application using Visual Studio 2005 and higher, you have to:

- Add **OPCNetServerSDK.dll** as a reference to your project.
- Add the files: License.dll, OPCDADLL.dll, DXServerDll.dll, SrvToolkit_CfgFile.ini and OpcDxServer.Config.xml to your output project folder (where your executable server will be generated).

# 6. Deploying Your Server Application Using Full Development License

## 6.1. Pre-Deployment Considerations

To deploy the OPC Server developed using the OPC Server Toolkit, you need to make sure that the pre-requisites are available in the target machine. Please refer to the pre-installation considerations section of this chapter.

## 6.2. Deployment Procedure

After releasing your server application in the development machine, the following files, and any other custom depending assembly, should be copied under your OPC server application folder:

| Source Folder | Files |
|---|---|
| DLLs | • License.dll<br>• OPCDADLL.dll<br>• DXServerDll.dll<br>• OPCNetServerSDK.dll, in case the server application is a .NET application |
| | |
| Configuration files | • SrvToolkit_CfgFile.ini<br>• OpcDxServer.Config.xml |

**Table 9: Deployment Files**

**Make sure to not copy any other Integration Objects files in the deployment folder. Otherwise, the following message will be displayed.**

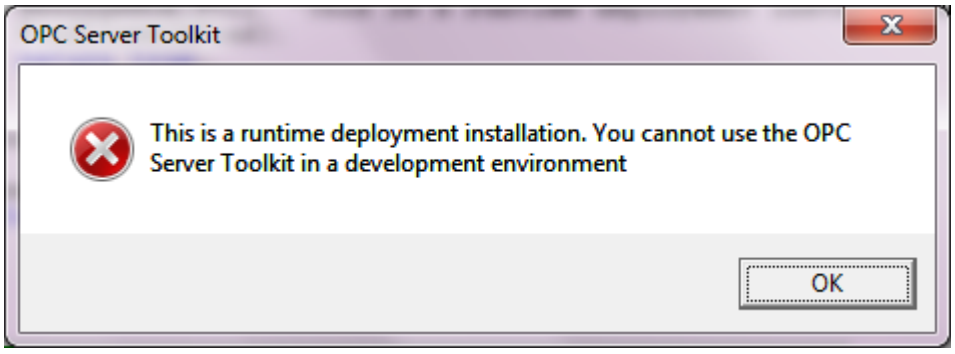


**Figure 18: Runtime Deployment Error**

**In case you are trying to deploy a full version, you need to check that no demo version of the OPC Server Toolkit is installed in the target machine. If it is the case, then you need to uninstall the demo version. Otherwise, the 2h demo version will be used.**

# 7. Migrating from Demo License to Full Development License

In order to migrate your development machine from the demo license to the full development license, follow the steps below:

1. Create a backup of your project and make sure that you project is not under the OPC Server toolkit folder
2. Uninstall the OPC Server Toolkit, previously installed with the demo license and make sure that the installation folder is empty.

3. Reinstall the OPC Server Toolkit and select the Full version type as illustrated in the figure below:

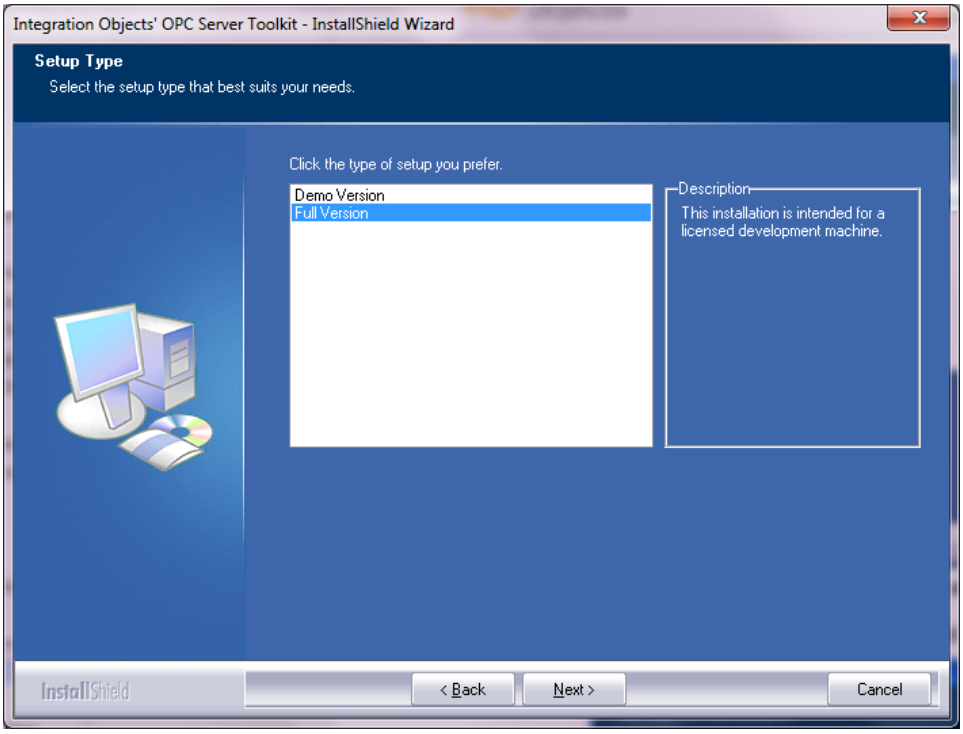


**Figure 19: Full Version Installation**

4. Click next and follow the installation instructions until you reach the user activation code dialog box.

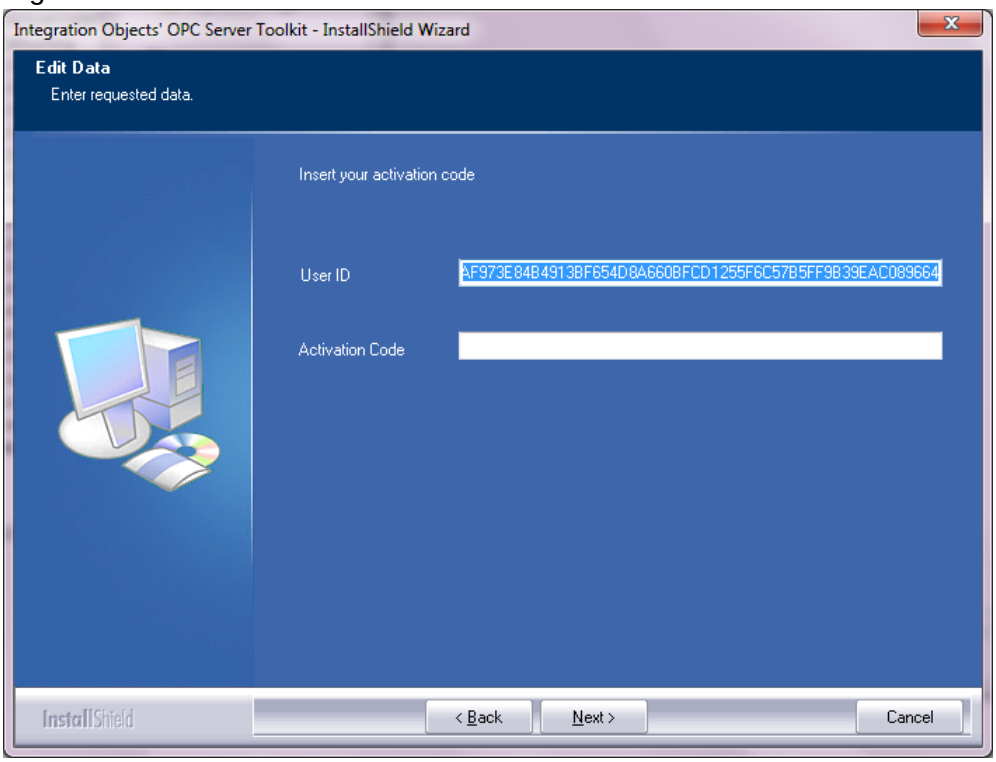

**Figure 20: Activation Dialog Box**

5. Copy the "User ID", then send it to Integration Objects' Sales team (sales@integrationobjects.com), and they will get back to you with the "Activation Code".
6. Enter the provided activation code, then click "Next" to get the following confirmation message:

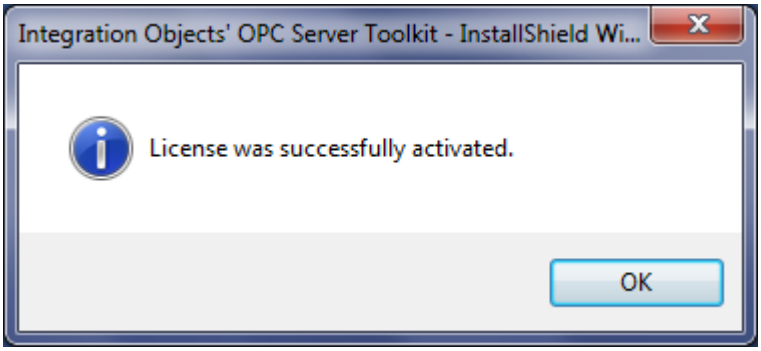


**Figure 21: Activation License Message Box**

7. Click the "Finish" button. Now, the **full version** of the OPC Server Toolkit is properly installed in your machine.
8. Open your custom server application and replace the old DxServer.dll in the project output folder with the new one that you can find under "**.:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\DLLs "**

## 8. Generate a new CLSID

Each OPC Server is identified by a unique CLSID. Microsoft provides a free CLSID generator utility called **GUIDGen.exe**.

To use this tool:

- Launch **GUIDGen.exe** from one of the paths below and that corresponds to the visual studio version installed in your machine.
  **VS 6.0:** C:\Program Files (x86)\Microsoft Visual Studio\Common\Tools
  **VS 2005:** C:\Program Files (x86)\Microsoft Visual Studio 8\Common7\Tools
  **VS 2008:** C:\Program Files (x86)\Microsoft Visual Studio 9.0\Common7\Tools
  **VS 2010:** C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Bin\NETFX 4.0 Tools
  **VS 2012:** C:\Program Files (x86)\Microsoft Visual Studio 11.0\Common7\Tools
  **VS 2013:** C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\Tools
  **VS 2015:** C:\Program Files (x86)\Microsoft Visual Studio 14.0\Common7\Tools
  **VS 2017:** C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\Common7\Tools
- Generate a new CLSID by clicking on the "New GUID" button as shown in the below figure.

- Copy it to the clipboard to be pasted later into your source code by clicking the "Copy" button.

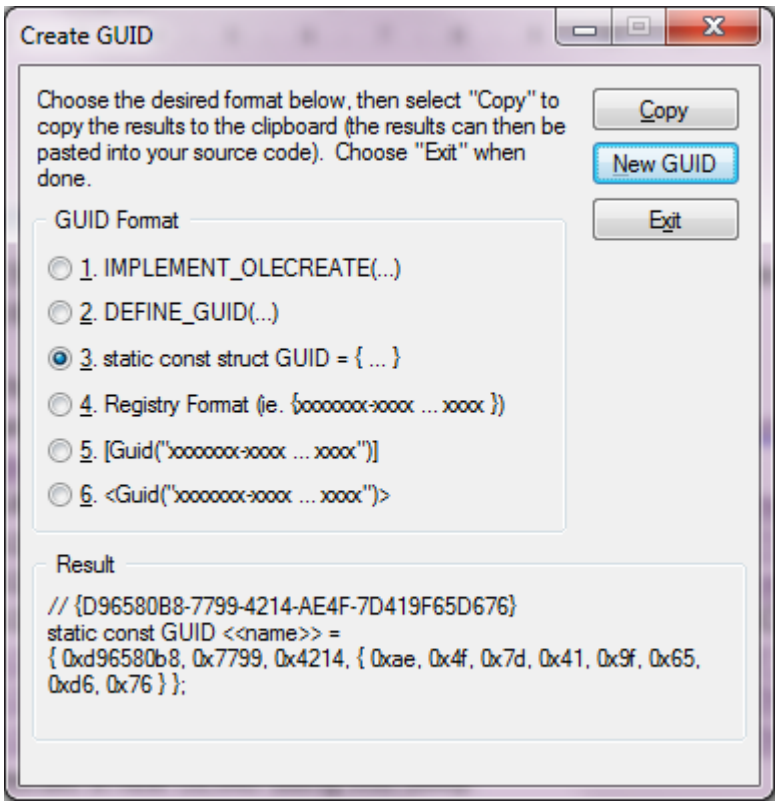


**Figure 22: GUID Generator**

**Sample CLSID**

DEFINE_GUID (CLSID_OPCSampleServer, 0x8f3ed890, 0xe0d3, 0x4d32, 0xa1, 0xf3, 0xec, 0xcb, 0x29, 0x76, 0xe6, 0x58);

# USING THE OPC SERVER TOOLKIT

## 1. Constants and Structures

### 1.1. Constants

**Server type**

The following constants are the possible OPC Server types supported by this toolkit.

- OPC_DA: OPC Data Access server
- OPC_DA_DX: OPC Data Access and Data eXchange server
- OPC_HDA: OPC Historical Data Access server

In .NET, you can use the class: **ServerType**.

- **ServerType .OPC_DA**: OPC Data Access server
- **ServerType .OPC_HDA**: OPC Historical Data Access server
- **ServerType .OPC_DA_DX**: OPC Data Access and Data eXchange server

**Source server type**

The following constants are reserved for Data eXchange use.

- SOURCE_SERVER_TYPE_COM_DA204: The source server supports OPC DA 2.04
- SOURCE_SERVER_TYPE_COM_DA205: The source server supports OPC DA 2.05

In .NET, you can use the class: **Data_eXchange**.

- Data_eXchange .SOURCE_SERVER_TYPE_COM_DA204: The source server supports OPC DA 2.04
- Data_eXchange .SOURCE_SERVER_TYPE_COM_DA205: The source server supports OPC DA 2.05

**Status**

The driver has to set the server status at least once.
In .NET, you can use the class: **ServerStatus**.

The possible values are described below.

| Status | Description |
|---|---|
| SERVER_RUNNING<br>**ServerStatus.SERVER_RUNNING** | The server is running normally. This is the usual state for a server. |
| SERVER_SUSPEND<br>**ServerStatus.SERVER_SUSPEND** | The server has been temporarily suspended via some vendor specific method and is not getting or sending data. Note that item ushoQuality will be returned as IO_USHOQUALITY_OUT_OF_SERVICE. |
| SERVER_FAILED<br>**ServerStatus.SERVER_FAILED** | A fatal error has occurred within the server. |
| SERVER_NOCONFIG<br>**ServerStatus.SERVER_NOCONFIG** | The server is running, but has no configuration information loaded, and thus does not transfer data. DX servers enter this state prior to their initial configuration and after having their configuration cleared. |
| SERVER_TEST<br>**ServerStatus.SERVER_TEST** | The server is in Test Mode. The outputs are disconnected from the real hardware but the server will otherwise behave normally. Inputs may be real or may be simulated depending on the vendor implementation. UshoQuality will generally be returned normally. |
| SERVER_COMMFAULT<br>**ServerStatus.SERVER_COMMFAULT** | The server is running properly but is having difficulty accessing data from its data sources. This may be due to communication problems, or some other problem preventing the underlying device, control system, etc. from returning valid data. It may be a complete failure, meaning that no data is available, or a partial failure, meaning that some data is still available. It is expected that items affected by the fault will individually return with a BAD ushoQuality indication for the items. |

**Table 10: Server State Values**

**Statistics**

The server can display some useful information like statistics about clients' connections and the count of created groups for all clients.

In .NET, you can use the class: **Statistics**

| Statistic ID | Description |
|---|---|
| NUM_CLIENTS<br>**Statistics.NUM_CLIENTS** | The total number of connected clients. |
| NUM_GROUPS<br>**Statistics.NUM_GROUPS** | The total number of created groups. |
| NUM_ITEMS<br>**Statistics. NUM_ITEMS** | The total number of added items. |
| NUM_DXCONNS<br>**Statistics.NUM_DXCONNS** | The total number of connections between the DX server and different source servers. |

**Table 11: Statistic ID Values**

**Log levels**

We defined seven levels for tracing. The last chapter describes this feature in detail.

In .NET, you can use the class: **LogLevel**

| Level | Description |
|---|---|
| LOG_CONTROLE (-1)<br>**LogLevel.LOG_CONTROLE** | It is the lowest level. This log file contains at least a description of succeeded methods. |
| LOG_FATAL_ERROR (0)<br>**LogLevel.LOG_FATAL_ERROR** | Only fatal error messages are logged. |
| LOG_CRITICAL_ERROR (1)<br>**LogLevel.LOG_CRITICAL_ERROR** | All critical error messages are logged. |
| LOG_ERROR (2)<br>**LogLevel.LOG_ERROR** | All errors are logged. |
| LOG_WARNING (3)<br>**LogLevel.LOG_WARNING** | All warnings are logged. |
| LOG_INFORMATION (4)<br>**LogLevel.LOG_INFORMATION** | All information is logged. |

| | |
|---|---|
| LOG_DEBUG (5)<br>**LogLevel.LOG_DEBUG** | This level is useful for debugging the DLL. |

**Table 12: Log Levels**

## 1.2. Structures

**TAGITEM**

The driver uses TAGITEM structure to update the list of items collected from the data source (device). This structure completely defines the information needed by the toolkit to manage data points.

```
typedef struct tagitem
{
        VARIANT         m_vValue;
        VARIANT         m_vEuInfo;
        double          m_dblMaxValue;
        double          m_dblMinValue;
        WORD            m_eEUType;
        FILETIME        m_ftTimestamp;
        WORD            m_wUshoQuality;
        DWORD           m_dwAccessRights;
        BSTR            tag;
        VARTYPE         m_vtDataType;

} TAGITEM;
```

**In C# .NET**

```
public struct TAGITEM
{
   public object m_vValue;
   public object m_vEuInfo;
   public double m_dblMaxValue;
   public double m_dblMinValue;
   public UInt16 m_eEUType;
   public FILETIME m_ftTimestamp;
   public UInt16 m_wUshoQuality;
   public UInt32 m_dwAccessRights;
   public string tag;
   public UInt16 m_vtDataType;
};
```

| Attribute Name | Description |
|---|---|
| tag | Item's identifier. It is the full path. |
| m_vValue | The value of the item. |
| m_vtDataType | The type of the item. It must be one of variant data types |

| | |
|---|---|
| | listed before (see section Supported Data Types). |
| m_ftTimestamp | The timestamp. |
| m_wUshoQuality | The ushoQuality of the item (valid values are presented in the section Appendix B: Item Qualities). |
| m_dwAccessRights | The access rights of the item. An item can be readable, writeable or readable and writeable.<br>Valid Values are:<br>IO_OPC_READABLE (read only)<br>IO_OPC_WRITEABLE (write only)<br>IO_OPC_READABLE \| IO_OPC_WRITEABLE (read and write) |
| m_eEUType | Defines the type of engineering unit this item.<br>Valid Values are:<br>IO_OPC_NOENUM (by default)<br>IO_OPC_ANALOG: for analog types<br>IO_OPC_ENUMERATED |
| m_dblMaxValue | Maximum value for analog type. |
| m_dblMinValue | Minimum value for analog type. |
| m_vEuInfo | If the m_eEUType is other than IO_OPC_ENUMERATED than this field is ignored.<br>Otherwise, this field holds an array of doubles or strings.<br>The developer has to fill in this field in case of string values (double values are filled by the toolkit). |

**Table 13: TAGITEM Attributes**

# 2. Functions

This section defines the main exported functions. These functions, their parameters, and behaviors are described in more details in the following sections.

We can distinguish three categories of exported functions. This section doesn't include callbacks routines.

| Category | Functions |
|---|---|
| Configuration routines | io_setservertype<br>io_intializeserver<br>io_setserverstatus<br>io_setstatus<br>io_requestdisconnect<br>io_closeall<br>io_registerserver<br>io_unregisterserver<br>io_forceshutdown<br>io_activategeneralcallbacks<br>io_activateclientconnectioncallbacks<br>io_getserverstatus |
| Exported functions for OPC DA/DX | **OPC DA**<br>io_getserverstatistic<br>io_getlastupdatetime<br>io_createtag<br>io_removetag<br>io_removetagfromcache<br>io_updatetag<br>io_activateitemiocallbacks (VC++)<br>io_activateitempropertiescallbacks (VC++)<br><br>VB_io_activateitemiocallbacks (VB[1])<br>VB_io_activateitempropertiescallbacks (VB)<br>VB_io_updatetag (VB)<br>**OPC DX**<br>io_startdadxmodules |

[1] Functions/callbacks that begin with "VB_" are designed for easy use in Visual Basic environment.

| | |
|---|---|
| Exported functions for OPC HDA | io_activatehistorycallbacks (VC++)<br>io_activateHDABrowsercallbacks (VC++)<br>io_hdasetitemaggregates (VC++)<br>io_hdasetitemattributes (VC++)<br>io_hdasetupdatemethods (VC++)<br>io_hdasethdaitems (VC++)<br>io_hdaAddhdaitems (VC++)<br>io_hdaRemovehdaitem (VC++)<br>io_hdaRemovehdaitems (VC++)<br>VB_io_activatehistorycallbacks (VB)<br>VB_io_hdasetitemaggregates (VB)<br>VB_io_hdasetitemattributes (VB)<br>VB_io_hdasetupdatemethods (VB)<br>VB_io_hdasethdaitems (VB) |

**Table 14: Functions by Categories**

## 2.1. io_setservertype

void io_setservertype ( [in] DWORD SvrType)

**Description**

Set the type of the OPC server. It should be called **only once**.

| Parameter Name | Description |
|---|---|
| SvrType | The server type. It can be one or a combination of the following:<br>OPC_DA<br>OPC_DA_DX<br>OPC_HDA<br>To combine 2 or more types you need to apply the bitwise-inclusive-OR operator (\|). |

**Table 15: Io_setservertype Parameters**

**Code example**

```
/*
It sets a server that supports OPC DA and HDA specifications.
*/
```

```
io_setservertype (OPC_DA | OPC_HDA);
```

**Return Codes**
None

**VB Wrapper**

```
Declare Sub io_setservertype Lib "DXServerDll.dll" (ByVal lngSvrType As Long)
```

**C# .NET**

```
void IO_SetServerType(UInt32 uintServerType)
```

**Code example**

```
//Create an instance  of the toolkit object
public static OPCNetServerSDKManager objOPCNetServerSDK = new OPCNetServerSDKManager();

//Set the OPC Server Type
uintSvrType = ServerType.OPC_DA_DX | ServerType.OPC_HDA;
objOPCNetServerSDK.IO_SetServerType(uintSvrType);
```

**VB .NET Wrapper**

```
Sub IO_SetServerType(ByVal uintServerType As UInt32)
```

**Code example**

```
//Create an instance  of the toolkit object
Public Shared ObjOPCNetServerSDK As New OPCNetServerSDKManager()

//Set the OPC Server Type
Public Shared uintSvrType As UInt32 = ServerType.OPC_DA_DX Or ServerType.OPC_HDA

ObjOPCNetServerSDK.IO_SetServerType(uintSvrType)
```

**C++ .NET**

```
void IO_SetServerType(UInt32 uintServerType)
```

**Code example**

```
//Create an instance  of the toolkit object
public: static OPCNetServerSDKManager^ ObjOPCNetServerSDK = gcnew
OPCNetServerSDKManager();

//Set the OPC Server Type
uintSvrType = ServerType.OPC_DA;

ObjOPCNetServerSDK->IO_SetServerType(uintSvrType);
```

## 2.2. io_initializeserver

```
int io_initializeserver (      [in] LPWSTR  VendorName
                             , [in] LPWSTR Descp
                             , [in] LPWSTR ClassNumber
                             , [in] LPWSTR VendorInfo
                             , [in] GUID    &ClsID
                             , [in] WORD   wMajorVersion
                             , [in] WORD   wMinorVersion
                             , [in] WORD   wBuildNumber
                             , [in] DWORD dwBandWidth
                             , [in] LPWSTR PathConfigFile
                             , [in] LPWSTR strCharSe
                             , [in] BOOL  bCheckTag )
```

**Description**

Set the most important server configuration parameters. It should be called **only once**. The core of this function encloses the call to the class factory.

**The driver can initialize the DLL at any point of its startup process, but it will only be able to handle effectively OPC client requests after calling the io_setstatus ( ) function.**

| Parameter Name | Description |
|---|---|
| VendorName | Vendor's name. |
| Descp | A description of the OPC server. |
| ClassNumber | The class number of the server application. |
| VendorInfo | Vendor's information. |
| ClsID | The class ID for this OPC server object (generated using the GUIDGen.exe utility). |
| wMajorVersion | The major version of the server application. |
| wMinorVersion | The minor version of the server application. |
| wBuildNumber | The build number of the server application. |
| dwBandWidth | The bandwidth of the server application. |

| | |
|---|---|
| PathConfigFile | The full path of the DX Configuration File (XML file). |
| strCharSe | Separator used for items.<br>The "/" separator is imposed by the OPC Foundation for DX. If there is any conflict with separator used by vendor to identify tags in the system, the DLL imposes a flat address space. |
| bCheckTag | Should be set to false in order to minimize the duration of creating an address space with more than 10000 tags. In this case, the toolkit will not handle the duplicated tags tests. |

**Table 16: Io_initializeserver Parameters**

| Code | Description |
|---|---|
| 0 | The function succeeded. |
| 1 | The function succeeded partially. That means, a configuration file problem has occurred, but the server can continue. The server has no configuration information loaded, and thus does not transfer data. |
| -1 | It is a fatal error. The given parameters are invalid to create a progID. The server object is not registered. In this case, the DLL couldn't be initialized. |
| -2 | It is a fatal error. The DLL couldn't initialize COM library. |
| -3 | It is a fatal error. The server is present in the registry but the class factory couldn't be initialized. In this case, the DLL couldn't be initialized. |
| -4 | It is a fatal error. Invalid separator. |
| -5 | It is a fatal error. It is the second call. |

**Table 17: Io_initializeserver Error Codes**

**Comments**

If the function succeeded, the ProgID is created as follows: **VendorName.Descp.ClassNumber**

**VB wrapper**

**Declare Function io_initializeserver Lib "DXServerDll.dll"**

```
(ByVal lngVendorName As Long, _
 ByVal lngDesc As Long, _
 ByVal lngClassNumber As Long, _
 ByVal lngVendorInfoString As Long,
```

```
ByRef clsID As GUID,
ByVal intMajorVersion As Integer,
ByVal intMinorVersion As Integer,
ByVal intBuildNumber As Integer,
ByVal lngBandWidth As Long,
ByVal lngPathConfigFile As Long,
ByVal lngCharSe As Long,
ByVal bCheckTag As Boolean,) As Integer
```

**With:**

**VB Code for wrapping the GUID Type**

```
Public Type GUID
    Data1 As Long
    Data2 As Integer
    Data3 As Integer
    Data4(7) As Byte
End Type
```

**VB Code example**

```
Dim strGUID As String
Dim udtCLSID As GUID
Dim i As Integer
Dim intResult As Integer
Dim param1 As Long
Dim param2 As Long
Dim param3 As Long
Dim param4 As Long
Dim param5 As Long
Dim param6 As Long
Dim param7 As Boolean

'Get CLSID
With udtCLSID
        .Data1 = 0
        .Data2 = 0
        .Data3 = 0
        For i = 0 To 7
          .Data4(i) = 0
        Next
End With

strGUID = "{FED10CD4-29C9-4b6b-9542-858F1818639D}"
' convert from string to a binary CLSID
```

```
CLSIDFromString StrPtr(strGUID), udtCLSID

param1 = StrPtr("IOOPC")
param2 = StrPtr("VB6.Simulation")
param3 = StrPtr("1")
param4 = StrPtr("Integration Objects, integrationobjects.com")
param5 = StrPtr("OpcDxServer.config.xml")
param6 = StrPtr("/")
param7 = False

intResult = io_initializeserver(param1, param2, param3, param4, udtCLSID, 2, 2, 0, 0, param5,
param6,param7)
```

**C# .NET**

```
int IO_InitializeServer(String strVendorName,
                        String strDescription,
                        String strClassNumber,
                        String strVendorInfoString,
                        ref Guid clsid,
                        UInt16 hMajorVersion,
                        UInt16 hMinorVersion,
                        UInt16 hBuildNumber,
                        UInt32 hBandWith,
                        String strPathConfigFile,
                        String strCharSe
                        Bool bCheckTag)
```

**Code example**

```
//Set the path of the DX File configuration
String PathConfFile = Application.StartupPath + "\\OpcDxServer.config.xml";
//The CLSID of the OPC Server
Guid clsid = new Guid("20260278-2008-1112-1982-111219829CAA");
//The separator caracter
String strCharSet = "/";
int intResult = objOPCNetServerSDK.IO_InitializeServer ("IntegrationObjects",
"DAHDASimulatorC#2008", "1", "Integration Objects, integrationobjects.com", clsid, 2, 2, 0, 0,
PathConfFile, strCharSet, bCheckTag);
```

**VB .NET Wrapper**

```
Function IO_InitializeServer(ByVal strVendorName As [String],
                             ByVal strDescription As [String],
                             ByVal strClassNumber As [String],
                             ByVal strVendorInfoString As [String],
                             ByRef clsid As Guid,
                             ByVal ushoMajorVersion As UInt16,
                             ByVal ushoMinorVersion As UInt16,
                             ByVal ushoBuildNumber As UInt16,
                             ByVal uintBandWith As UInt32,
                             ByVal strPathConfigFile As [String],
                             ByVal strCharSe As [String]
                             ByVal bCheckTag As [Boolean]) As Integer
```

**Code example**

```
'Set the path of the DX File configuration
Dim PathConfFile As [String] = Application.StartupPath &
"\OpcDxServer.config.xml"
 'The CLSID of the OPC Server
 Dim clsid As New Guid("20260278-2008-1112-1982-111219829B31")
 'The separator caracter
Dim strCharSet As [String] = "/"
Dim intResult As Integer = objOPCNetServerSDK.IO_initializeserver ("IntegrationObjects",
"DAHDASimulatorVB2008","1", "Integration Objects, integrationobjects.com", clsid, 2, 2, 0, 0,
PathConfFile, strCharSet, False)
```

**C++ .NET**

```
int IO_InitializeServer(String^ strVendorName,
                        String^ strDescription,
                        String^ strClassNumber,
                        String^ strVendorInfoString,
                        ref Guid^ clsid,
                        UInt16 ushoMajorVersion,
                        UInt16 ushoMinorVersion,
                        UInt16 ushoBuildNumber,
                        UInt32 uintBandWith,
                        String^ strPathConfigFile,
                        String^ strCharSe,
                        Bool bCheckTag)
```

**Code example**

```
//Set the path of the DX File configuration
String^ PathConfFile = Application::StartupPath + "\\OpcDxServer.config.xml";
//The CLSID of the OPC Server
Guid^ clsid = gcnew Guid("20260278-2008-1112-1982-111219829C08");
//The separator caracter
String^ strCharSet = "/";
Int intResult = objOPCNetServerSDK->IO_InitializeServer("IntegrationObjects" ,
"DAHDASimulatorC++.Net2008" , "1" , "Integration Objects, integrationobjects.com" , *clsid, 2, 2, 0, 0,
PathConfFile  //DX: you should pass the DX configuration file path , strCharSet, false);
```

## 2.3. io_setserverstatus

```
HRESULT    io_setserverstatus ([in] UINT MaxDxConnections
                              , [in] UINT uMaxQueSize
                              , [in] DWORD uintCount
                              , [in] LPCWSTR* SourceServersTypes
                              , [in] DWORD dwServerRate
                              , [in] bool  bDrivenMode,
                              , [in] DWORD dwMaxReturnValues
                              , [out] int**      pErrors)
```

**Description**

This sets the specific OPC Server DX/HDA configuration parameters. It also sets the server scan rate. It should be called **only once**.

| Parameter Name | Description |
|---|---|
| dwServerRate | The rate (in milliseconds) at which the DLL will update the cache and process the asynchronous client read and write requests.<br>Example:<br>A server rate set to 1000 milliseconds means that a timer set to 1 second will loop within the DLL to update the server cache and check the queued asynchronous requests posted by the clients.<br>The server rate is at least 100 ms.<br>The end user can implement a method, which loads the dwServerRate value from the SrvToolkit_CfgFile.ini file instead of having this parameter hard coded in the application server. |
| bDrivenMode | If true, the user has to implement a custom update of the OPC Server cache by calling the io_updatetag method.<br>If false, the cache will be automatically updated by the OPC Server Toolkit. |
| MaxDxConnections | It stores the maximum number of DXConnections that a DX server supports due to licensing or other limitations. If only system resources limit the server, then the value of this parameter is 0xFFFFFFFF.<br>At least, the server supports one DXConnection. |
| uintCount | It's the number of supported source servers. |
| SourceServersTypes | This parameter identifies supported Source Server Types that the DX server is capable of supporting for connections to source servers.<br>Possible values are:<br>SOURCE_SERVER_TYPE_COM_DA204<br>SOURCE_SERVER_TYPE_COM_DA205 |
| uMaxQueSize | This parameter stores the maximum number of source updates that a DX server can queue per DXConnection, when data is received faster than it can be written to the target. If the DX server does not support queuing, this value should be set to 1 indicating that only the latest source update will be cached for target updates. |

| | |
|---|---|
| dwMaxReturnValues | 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. |
| pErrors | It's an out parameter. This function sets three important configuration parameters; related fault codes are stored in pErrors at the same order as these parameters are presented.<br>The developer has to free resources allocated by the DLL for pErrors if the call succeeded. |

**Table 18: Io_setserverstatus Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. The configuration file is updated. |
| S_FALSE | Possible causes:<br>There is at least one error in pErrors; one or more parameters are invalid.<br>It could be an access failure to the configuration file. |
| E_INVALIDARG | pErrors = NULL. |
| E_FAIL | The function failed. |

**Table 19: Io_setserverstatus Error Codes**

| Index | Code | Description |
|---|---|---|
| 0 | 0<br>-1 | The MaxDxConnection is valid.<br>Invalid MaxDxConnection, it is less than 1. |
| 1 | 0<br>-1 | The MaxQueueSize is valid.<br>Invalid MaxQueSize, it is less than 1. |
| 2 | 0<br>-1 | The source server types are valid.<br>UintCount = 0 or invalid source server types. |

**Table 20: [pErrors] Error Codes**

If all parameters are valid, all pErrors fault codes are equal to 0 and the XML configuration file is updated with the following configuration parameters:

- MaxDxConnections.
- MaxQueSize.
- SourceServerTypes.

**Comments**

If fault code = S_OK, the function completed successfully and thus pErrors are ignored.
If fault code = S_FALSE, the driver should check pErrors. If pErrors[i] succeeded then there is a configuration file problem.
If fault code = E_FAIL, a fatal error has occurred in two cases:

- The server scan rate is invalid. The server should be stopped because this parameter is critical for the DLL processing.
- It is the second call to this function.

If fault code = E_INVALIDARG, you passed a null pointer for pErrors.

**VB Wrapper**

```
Declare Function io_setserverstatus Lib "DXServerDll.dll" _
                    (ByVal intMaxDxConnection As Integer, _
                     ByVal intMaxQueSize As Integer, _
                     ByVal lngCount As Long, _
                     lngSourceServersTypes As Long, _
                     ByVal lngServer_Rate As Long, _
                    ByVal bDrivenMode As Boolean,_
                     ByVal lngMaxReturnValues As Long, _
                     intErrors() As Integer) As Long
```

**VB Code example**

```
Dim intResult As Long
Dim pErrors(0 To 2) As Integer
Dim SourceServersTypes(0 To 1) As Long
SourceServersTypes(0) = StrPtr("COM-DA2.04")
SourceServersTypes(1) = StrPtr("COM-DA2.05")
‘ Support 5 DXConnections, 100 the maximum number of returned hda ‘ values
‘ And a scan rate of 1 second.
intResult = io_setserverstatus(5, 1, 2, SourceServersTypes(0), 1000, false,100, pErrors)
```

**C# .NET**

```
int IO_SetServerStatus(UInt32 uintMaxDxConnection,
                       UInt32 uintMaxQueSize,
                       UInt32 uintCount,
                       string[] strSourceServersTypes,
                       UInt32 uintServerRate,
                       bool blnDrivenMode,
                       UInt32 uintMaxReturnValues,
```

```
                out int[] intErrors)
```

**Code example**

```
public static UInt32 dwServer_Rate = 500;
string[] SourceServersTypes = { "", "" };//Source Server Types
SourceServersTypes[0] = Data_eXchange.SOURCE_SERVER_TYPE_COM_DA204;
SourceServersTypes[1] = Data_eXchange.SOURCE_SERVER_TYPE_COM_DA205;
UInt32 MaxDxConnection = 5;
UInt32 MaxQueSize = 1;
UInt32 dwCount = 2;
UInt32 dwMaxReturnValues = 100;
 int[] pErrors;
int intResult = objOPCNetServerSDK.IO_SetServerStatus(uintMaxDxConnection, uintMaxQueSize,
uintCount, strSourceServersTypes, uintServerRate, blnDrivenMode, uintMaxReturnValues, intErrors);
```

**VB .NET Wrapper**

```
Function IO_SetServerStatus(ByVal uintMaxDxConnection As UInt32,
                            ByVal uintMaxQueSize As UInt32,
                            ByVal uintCount As UInt32,
                            ByVal strSourceServersTypes As String(),
                            ByVal uintServerRate As UInt32,
                            ByVal blnDrivenMode As Boolean,
                            ByVal uintMaxReturnValues As UInt32,
                            ByRef intErrors As Integer()) As Integer
```

**Code example**

```
'The server Rate
 Public Shared dwServer_Rate As UInt32 = 500
'Source Server Types
 SourceServersTypes(0) = Data_eXchange.SOURCE_SERVER_TYPE_COM_DA204
 SourceServersTypes(1) = Data_eXchange.SOURCE_SERVER_TYPE_COM_DA205

 Dim MaxDxConnection As UInt32 = 5
 Dim MaxQueSize As UInt32 = 1
 Dim dwCount As UInt32 = 2
 Dim dwMaxReturnValues As UInt32 = 100
 Dim pErrors As Integer()
 'Set the server configuration parameter and the scan rate.
Dim intResult As Integer = objOPCNetServerSDK.IO_SetServerStatus (uintMaxDxConnection,
uintMaxQueSize, uintCount, strSourceServersTypes, uintServerRate, blnDrivenMode,
uintMaxReturnValues, intErrors);
```

**C++ .NET**

```
int IO_SetServerStatus(UInt32 uintMaxDxConnection,
                       UInt32 uintMaxQueSize,
                       UInt32 uintCount,
                       array<String^>^ strSourceServersTypes,
                       UInt32 uintServerRate,
                       bool blnDrivenMode,
```

```
UInt32 uintMaxReturnValues,
array<int^>^ %intErrors)
```

**Code example**

```
//The server Rate
public: static UInt32 dwServer_Rate = 1000;
array<String^,1>^ SourceServersTypes = { "", "" };//Source Server Types

SourceServersTypes[0] = Data_eXchange::SOURCE_SERVER_TYPE_COM_DA204;
SourceServersTypes[1] = Data_eXchange::SOURCE_SERVER_TYPE_COM_DA205;

UInt32 MaxDxConnection = 5;
UInt32 MaxQueSize = 1;
UInt32 dwCount = 2;
UInt32 dwServer_Rate = 1000;
UInt32 dwMaxReturnValues = 100;
array<int,1>^ pErrors;

//Set the server configuration parameter and the scan rate.
int intResult = objOPCNetServerSDK->IO_SetServerStatus(uintMaxDxConnection, uintMaxQueSize,
uintCount, strSourceServersTypes, uintServerRate, blnDrivenMode, uintMaxReturnValues, intErrors);
```

## 2.4. io_activategeneralcallbacks

**HRESULT IO_ActivateGeneralCallbacks(**
**Del_IOSVRCANUNLOADNOWPROC Del_IOSVRCANUNLOADNOWPROC1,**
**Del_IOGETERRORSTRINGPROC Del_IOGETERRORSTRINGPROC1)**

**Description**

The developer should implement the callbacks cores and pass a pointer of these functions to the DLL. The DLL invokes these callbacks whenever it needs specific information from the device supervised by the OPC server. You will find below a reserved section that explains these callbacks.

| Parameter Name | Description |
|---|---|
| Del_IOSVRCANUNLOADNOWPROC1 | Pointer to SvrCanUnloadNow function |
| Del_IOGETERRORSTRINGPROC1 | Pointer to GetErrorString function |

**Table 21: Io_activategeneralcallbacks Parameters**

**VB Wrapper**

```
Declare Function io_activategeneralcallbacks Lib "DXServerDll.dll" _
                (ByVal fnc1 As Long, _
                 ByVal fnc2 As Long) As Long
```

**VB Code example**

```
Dim lngResult As Long
```

```
lngResult = io_activategeneralcallbacks(AddressOf SvrCanUnloadNow, AddressOf
GetErrorString)
'SvrCanUnloadNow and GetErrorString are callbacks
```

**C# .NET**

```
void IO_ActivateGeneralCallbacks(
                        OPCNetServerSDKManager.Del_IOSVRCANUNLOADNOWPROC
                        del_IOSVRCANUNLOADNOWPROC,
                        OPCNetServerSDKManager.Del_IOGETERRORSTRINGPROC
                        del_IOGETERRORSTRINGPROC);
```

**Code example**

```
private OPCNetServerSDKManager.Del_IOSVRCANUNLOADNOWPROC
delIOSVRCANUNLOADNOWPROC = new
OPCNetServerSDKManager.Del_IOSVRCANUNLOADNOWPROC(SvrCanUnloadNow);
private OPCNetServerSDKManager.Del_IOGETERRORSTRINGPROC delIOGETERRORSTRINGPROC
= new OPCNetServerSDKManager.Del_IOGETERRORSTRINGPROC(GetErrorString);

objOPCNetServerSDK.IO_ActivateGeneralCallbacks(delIOSVRCANUNLOADNOWPROC1,delIOGETER
RORSTRINGPROC1);
```

**VB .NET Wrapper**

```
Sub IO_ActivateGeneralCallbacks(
  ByVal del_IOSVRCANUNLOADNOWPROC As Del_IOSVRCANUNLOADNOWPROC,
  ByVal del_IOGETERRORSTRINGPROC As Del_IOGETERRORSTRINGPROC)
```

**Code example**

```
//io_activategeneralcallbacks SvrCanUnloadNow/GetErrorString
public del_IOSVRCANUNLOADNOWPROC As New
OPCNetServerSDKManager.Del_IOSVRCANUNLOADNOWPROC(AddressOf SvrCanUnloadNow)
public del_IOGETERRORSTRINGPROC As New
OPCNetServerSDKManager.Del_IOGETERRORSTRINGPROC(AddressOf GetErrorString)

objOPCNetServerSDK.IO_activategeneralcallbacks(del_IOSVRCANUNLOADNOWPROC,
del_IOGETERRORSTRINGPROC)
```

**C++ .NET**

```
void IO_ActivateGeneralCallbacks(
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOSVRCANUNLOADNOWPROC^
del_IOSVRCANUNLOADNOWPROC,
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOGETERRORSTRINGPROC^
del_IOGETERRORSTRINGPROC);
```

**Code example**

```
public:  OPCNetServerSDK::OPCNetServerSDKManager::Del_IOSVRCANUNLOADNOWPROC^
del_IOSVRCANUNLOADNOWPROC;
public:  OPCNetServerSDK::OPCNetServerSDKManager::Del_IOGETERRORSTRINGPROC^
del_IOGETERRORSTRINGPROC;

objOPCNetServerSDK->IO_ActivateGeneralCallbacks (del_IOSVRCANUNLOADNOWPROC,
del_IOGETERRORSTRINGPROC);
```

## 2.5. io_activateitemiocallbacks

HRESULT io_activateitemiocallbacks( [in] IOWRITEITEMPROC lpWriteItem
,[in] IOREADITEMPROC lpReadItem
,[in] IOUPDATEITEMPROC lpUpdateItem
,[in] IOWRITEVQTPROC lpWriteVQTItem
,[in] IOVALIDATEITEMPROC lpValidItem
,[in] IOVALIDATEANDADDPROC lpValidateAndInsert)

**Description**

This sets callbacks to add/validate tags and read/write their values.

| Parameter Name | Description |
|---|---|
| lpWriteItem | Pointer to WriteItem callback |
| lpReadItem | Pointer to ReadItem callback |
| lpUpdateItem | Pointer to UpdateItem callback |
| lpWriteVQTItem | Pointer to WriteVQT callback |
| lpValidItem | Pointer to ValidItem callback |
| lpValidateAndInsert | Pointer to ValidateAndInsert callback |

**Table 22: Io_activateitemiocallbacks Parameters**

**VB Wrapper**

```
Declare Function VB_io_activateitemiocallbacks Lib "DXServerDll.dll" _
                (ByVal fnc1 As Long, _
                 ByVal fnc2 As Long, _
                 ByVal fnc4 As Long, _
                 ByVal fnc5 As Long, _
                 ByVal fnc6 As Long) As Long
```

**When programming with Visual Basic 6, the UpdateItem callback is not required since it has been replaced by the io_updateitem function.**

**VB Code example**

```
Dim lngResult As Long
lngResult = VB_io_activateitemiocallbacks(AddressOf WriteItem, AddressOf ReadItem, AddressOf WriteVQT, AddressOf ValidateItem, AddressOf ValidateAndAdd)
```

**C# .NET**

```
void IO_ActivateItemioCallbacks(
OPCNetServerSDKManager.Del_IOWRITEITEMPROC del_IOWRITEITEMPROC,
OPCNetServerSDKManager.Del_IOREADITEMPROC del_IOREADITEMPROC,
OPCNetServerSDKManager.Del_IOUPDATEITEMPROC del_IOUPDATEITEMPROC,
OPCNetServerSDKManager.Del_IOWRITEVQTPROC del_IOWRITEVQTPROC,
OPCNetServerSDKManager.Del_IOVALIDATEITEMPROC del_IOVALIDATEITEMPROC,
OPCNetServerSDKManager.Del_IOVALIDATEANDADDPROC
del_IOVALIDATEANDADDPROC)
```

**Code example**

```
//Activate the Update Item callbacks
ObjOPCNetServerSDK.IO_ActivateItemioCallbacks(delIOWRITEITEMPROC
                                              , delIOREADITEMPROC
                                    , delIOUPDATEITEMPROC
                                  , delIOWRITEVQTPROC
                                  , delIOVALIDATEITEMPROC
                                  , delIOVALIDATEANDADDPROC);
```

**VB .NET Wrapper**

```
Sub IO_ActivateItemioCallbacks(
ByVal del_IOWRITEITEMPROC As OPCNetServerSDKManager.Del_IOWRITEITEMPROC , ByVal del_IOREADITEMPROC As OPCNetServerSDKManager.Del_IOREADITEMPROC , ByVal del_IOUPDATEITEMPROC As OPCNetServerSDKManager.Del_IOUPDATEITEMPROC
ByVal del_IOWRITEVQTPROC As OPCNetServerSDKManager.Del_IOWRITEVQTPROC, ByVal del_IOVALIDATEITEMPROC As OPCNetServerSDKManager.Del_IOVALIDATEITEMPROC,
ByVal del_IOVALIDATEANDADDPROC As OPCNetServerSDKManager.Del_IOVALIDATEANDADDPROC)
```

**Code example**

```
objOPCNetServerSDK.IO_activateitemiocallbacks(delIOWRITEITEMPROC, delIOREADITEMPROC, delIOUPDATEITEMPROC, delIOWRITEVQTPROC, delIOVALIDATEITEMPROC, delIOVALIDATEANDADDPROC)
```

**C++ .NET**

```
void IO_ActivateItemioCallbacks (
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOWRITEITEMPROC ^
del_IOWRITEITEMPROC,
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOREADITEMPROC ^
del_IOREADITEMPROC,
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOUPDATEITEMPROC ^
```

del_IOUPDATEITEMPROC,
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOWRITEVQTPROC ^
del_IOWRITEVQTPROC,
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOVALIDATEITEMPROC ^
del_IOVALIDATEITEMPROC,
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOVALIDATEITEMPROC ^
del_IOVALIDATEANDADDPROC);

**Code example**

```
objOPCNetServerSDK->IO_ActivateItemioCallbacks(delIOWRITEITEMPROC, del_IOREADITEMPROC,
del_IOUPDATEITEMPROC, del_IOWRITEVQTPROC, del_IOVALIDATEITEMPROC,
del_IOVALIDATEANDADDPROC);
```

## 2.6. io_activateitempropertiescallbacks

```
HRESULT  io_activateitempropertiescallbacks (
              [in] IOQUERYAVAILABLEPROPERTIESPROC lpQeryAProp
             ,[in] IOGETITEMPROPERTIESPROC lpGetItmProp
             ,[in]IOLOOKUPITEMIDSPROC lpLookItmProp)
```

**Description**

It sets callbacks for item properties.

| Parameter Name | Description |
|---|---|
| lpQeryAProp | Pointer to QueryAvailableProperties function |
| lpGetItmProp | Pointer to GetItemProperties function |
| lpLookItmProp | Pointer to LookUpItemProperties function |

**Table 23: Io_activateitempropertiescallbacks Parameters**

**VB Wrapper**

```
Declare Function VB_io_activateitempropertiescallbacks Lib "DXServerDll.dll"
                    (ByVal fnc0 As Long, _
                     ByVal fnc1 As Long, _
                     ByVal fnc2 As Long, _
                     ByVal fnc3 As Long) As Long
```

**VB Code example**

```
Dim lngResult As Long
intResult = VB_io_activateitempropertiescallbacks(AddressOf NumberOfItemProperties,
AddressOf QueryAvailableProps, AddressOf GetItemProps, AddressOf LookItemProps)
```

**C# .NET**

```
VoidI O_activateitempropertiescallbacks
(OPCNetServerSDK.OPCNetServerSDKManager.Del_IOQUERYAVAILABLEPROPERTIESPRO
CBSTR del_IOQUERYAVAILABLEPROPERTIESPROCBSTR,
OPCNetServerSDK.OPCNetServerSDKManager.Del_IOGETITEMPROPERTIESPROC
del_IOGETITEMPROPERTIESPROC,
OPCNetServerSDK.OPCNetServerSDKManager.Del_IOLOOKUPITEMIDSPROC
del_IOLOOKUPITEMIDSPROC)
```

**Code example**

```
//Activate the Item properties callbacks
objOPCNetServerSDK.IO_ActivateItemPropertiesCallbacks(del_IOQUERYAVAILABLEPROPERTIESPR
OCBSTR,del_IOGETITEMPROPERTIESPROC, del_IOLOOKUPITEMIDSPROC);
```

**VB .NET Wrapper**

```
 Sub IO_activateitempropertiescallbacks (
ByVal del_IOQUERYAVAILABLEPROPERTIESPROCBSTR As
OPCNetServerSDK.OPCNetServerSDKManager.Del_IOQUERYAVAILABLEPROPERTIESPROC
BSTR,
ByVal del_IOGETITEMPROPERTIESPROC As
OPCNetServerSDK.OPCNetServerSDKManager.Del_IOGETITEMPROPERTIESPROC,
ByVal del_IOLOOKUPITEMIDSPROC As
OPCNetServerSDK.OPCNetServerSDKManager.Del_IOLOOKUPITEMIDSPROC)
```

**Code example**

```
objOPCNetServerSDK.IO_ActivateItemPropertiesCallbacks(

del_IOQUERYAVAILABLEPROPERTIESPROCBSTR,del_IOGETITEMPROPERTIESPROC,

del_IOLOOKUPITEMIDSPROC);
```

**C++ .NET**

```
void IO_activateitempropertiescallbacks(
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOQUERYAVAILABLEPROPERTIESPRO
CBSTR^ del_IOQUERYAVAILABLEPROPERTIESPROCBSTR,
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOGETITEMPROPERTIESPROC^
del_IOGETITEMPROPERTIESPROC,
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOLOOKUPITEMIDSPROC^
del_IOLOOKUPITEMIDSPROC)
```

**Code example**

```
//Activate the Item properties callbacks
objOPCNetServerSDK->IO_ActivateItemPropertiesCallbacks
(del_IOQUERYAVAILABLEPROPERTIESPROCBSTR
,del_IOGETITEMPROPERTIESPROC                    , del_IOLOOKUPITEMIDSPROC);
```

## 2.7. io_activateclientconnectioncallbacks

HRESULT io_activateclientconnectioncallbacks (
[in] IOONCLIENTCONNECTPROC lpConn,
[in] IOONCLIENTDISCONNECTPROC lpDisConn)

**Description**

It sets callbacks for client connection/disconnection.

| Parameter Name | Description |
|---|---|
| lpConn | Pointer to OnClientConnect function |
| lpDisConn | Pointer to OnClientDisConnect function |

**Table 24: Io_activateclientconnectionscallbacks Parameters**

**VB Wrapper**

```
Declare Function io_activateclientconnectioncallbacks Lib "DXServerDll.dll" _
                (ByVal fnc1 As Long _
                , ByVal fnc2 As Long _
                ) As Long
```

**VB Code example**

```
Dim lngResult As Long
intResult = io_activateclientconnectioncallbacks(AddressOf OnClientConnect, AddressOf
OnClientDisconnect)
```

**C# .NET**

```
void IO_activateclientconnectioncallbacks(
OPCNetServerSDK.OPCNetServerSDKManager.Del_IOONCLIENTCONNECTPROC
del_IOONCLIENTCONNECTPROC,
OPCNetServerSDK.OPCNetServerSDKManager.Del_IOONCLIENTDISCONNECTPROC
del_IOONCLIENTDISCONNECTPROC)
```

**Code example**

```
//Activate the connection callbacks

objOPCNetServerSDK.IO_ActivateClientConnectionCallbacks(del_IOONCLIENTCONNECTPROC,

del_IOONCLIENTDISCONNECTPROC);
```

**VB .NET Wrapper**

```
Sub IO_activateclientconnectioncallbacks(
```

```
ByVal Del_IOONCLIENTCONNECTPROC1 As
OPCNetServerSDK.OPCNetServerSDKManager.Del_IOONCLIENTCONNECTPROC,
ByVal Del_IOONCLIENTDISCONNECTPROC1 As
OPCNetServerSDK.OPCNetServerSDKManager.Del_IOONCLIENTDISCONNECTPROC)
```

**Code example**

```
objOPCNetServerSDK.IO_activateclientconnectioncallbacks(del_IOONCLIENTCONNECTPROC,

del_IOONCLIENTDISCONNECTPROC)
```

**C++ .NET**

```
void IO_ActivateClientConnectionCallbacks (
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOONCLIENTCONNECTPROC^
del_IOONCLIENTCONNECTPROC,
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOONCLIENTDISCONNECTPROC^
del_IOONCLIENTDISCONNECTPROC)
```

**Code example**

```
//Activate the connection callbacks
objOPCNetServerSDK->IO_ActivateClientConnectionCallbacks(        delIOONCLIENTCONNECTPROC,
delIOONCLIENTDISCONNECTPROC);
```

## 2.8. io_createtag

HRESULT io_createtag (TAGITEM*  pItem)

**Description**

Create a tag in the server address space. This function is used to add only Data Access items (collected from device). It returns an integer value that indicates whether the item has been successfully created (S_OK) or it already exists in the server address space (S_FALSE). "pItem" structure should be filled entirely.

**Please note that the number of created tags is limited to 100000 tags.**

**VB Wrapper**

```
Declare Function io_createtag Lib "DXServerDll.dll" (param As TAGITEM) As Integer
```

**With**

**VB Code for wrapping the TAGITEM structure**

```
Public Type TAGITEM
    Value As Variant
```

```
    EuInfo As Variant
    MaxValue As Double
    MinValue As Double
    EUType As Integer
    TimeStamp As FILETIME
    UshoQuality As Integer
    AccessRights As Long
    StrTagName As String
    DataType As Integer
End Type
```

**And**

**VB Code for wrapping the FILETIME Type**

```
' Note that FILETIME is a user defined structure in VB.
Public Type FILETIME
    dwLowDateTime As Long
    dwHighDateTime As Long
End Type
```

**VB Code example**

```
Dim intResult As Integer
Dim tag1 As TAGITEM

tag1.Value = CVar(CLng(10))
tag1.StrTagName = "DA/Static/ReadWrite/TAGVT_I4"
tag1.AccessRights = 3   'readable/writeable
tag1.UshoQuality = 192      'good
tag1.DataType = 3       'VT_I4
CoFileTimeNow tag1.TimeStamp
intResult = io_createtag(tag1)
```

**C# .NET**

```
int IO_CreateTag(OPCNetServerSDK.TAGITEM Item)
```

**Code example**

```
int intResult = 0;

OPCNetServerSDK.TAGITEM TempTag;

intResult = objOPCNetServerSDK.IO_CreateTag(TempTag);
```

**VB .NET Wrapper**

```
Function IO_CreateTag(ByVal Item As OPCNetServerSDK.TAGITEM) As Integer
```

**Code example**

```
Dim intResult As Integer
Dim tempTag As OPCNetServerSDK.TAGITEM
bResult = ObjOPCNetServerSDK.IO_CreateTag(TempTag)
```

**C++ .NET**

```
int IO_CreateTag (OPCNetServerSDK.TAGITEM* Item)
```

**Code example**

```
int iResult;

TAGITEM^ tempTag;

TempTag = gcnew OPCNetServerSDK::TAGITEM();

TempTag->m_vValue = 100;

TempTag->m_vEuInfo = gcnew Object();

TempTag->m_dblMinValue = 0.0;

TempTag->m_dblMaxValue = 0.0;

TempTag->m_eEUType = 0;

TempTag->m_ftTimestamp =

OPCNetServerSDK::OPCNetServerSDKManager::GetFileTimeFromDateTime(DateTime::Now);

TempTag->m_wUshoQuality = OPC_UshoQuality::IO_OPC_USHOQUALITY_GOOD;

TempTag->m_dwAccessRights = OPC_AccessRights.IO_OPC_READABLE_WRITEABLE;

TempTag->tag = "Tag.VT_I1";

TempTag->m_vtDataType = (int)VarEnum::VT_I1;

intResult = OPCServer::ObjOPCNetServerSDK->IO_CreateTag(*TempTag);
```

## 2.9. io_removetag

HRESULT io_removetag (BSTR  bstrFullStrTagName)

**Description**

Remove a tag from the server address space (and thus from its cache) in runtime mode. This function is used to remove only Data Access items (collected from device). It returns HRESULT value that indicates whether the item has been successfully removed (S_OK) or it does not figure in the server address space (S_FALSE) to be removed. "bstrFullStrTagName" is the full tag name.

**VB Wrapper**

Declare Function io_removetag Lib "DXServerDll.dll" (ByVal strStrTagName As String) As Integer

**C# .NET**

```
int IO_RemoveTag(string strStrTagName);
```

**Code example**

```
string strStrTagName;

int intResult = objOPCNetServerSDK.IO_RemoveTag(strStrTagName);
```

**VB .NET Wrapper**

```
Function IO_RemoveTag(ByVal strStrTagName As String) As Integer
```

**Code example**

```
Dim strStrTagName As String

int intResult As Integer = objOPCNetServerSDK.IO_RemoveTag(strStrTagName)
```

**C++ .NET**

```
int IO_RemoveTag (String^ strStrTagName)
```

**Code example**

```
String^ strStrTagName;

int intResult = OPCServer::ObjOPCNetServerSDK->IO_RemoveTag (strStrTagName);
```

## 2.10. io_removetagfromcache

HRESULT io_removetagfromcache (BSTR bstrFullStrTagName)

**Description**

Remove a tag from the server cache in runtime mode. The removed tag will no longer be updated, but it will remain in the server address space.

This function is used to remove only Data Access items (collected from device). It returns a HRESULT value that indicates whether the item has been successfully removed (S_OK) or it does not figure in the server address space (S_FALSE) to be removed. "bstrFullStrTagName" is the full tag name.

**VB Wrapper**

Declare Function io_removetagfromcache Lib "DXServerDll.dll" (ByVal StrTagName As String) As Integer

**C# .NET**

```
int IO_RemoveTagFromCache(string strStrTagName)
```

**Code example**

```
string strStrTagName;
int intResult = objOPCNetServerSDK.IO_RemoveTagFromCache(strStrTagName);
```

**VB .NET Wrapper**

```
Function IO_ RemoveTagFromCache (ByVal strStrTagName As String) As Integer
```

**Code example**

```
Dim strStrTagName As String

int intResult As Integer = objOPCNetServerSDK.IO_RemoveTagFromCache (strStrTagName)
```

**C++ .NET**

```
int IO_RemoveTagFromCache (String^ strStrTagName)
```

**Code example**

```
String^ strStrTagName;



int intResult = objOPCNetServerSDK->IO_RemoveTagFromCache (strStrTagName);
```

## 2.11. io_removeAlltags

HRESULT io_removeAlltags();

**Description**

Removes all the tags from the server address space.

It returns an integer value that indicates whether the items have been successfully removed (S_OK) or any error occurred (S_FALSE) in removing them.

## 2.12. io_startdadxmodules

int io_startdadxmodules ( )

**Description**

This function is only available for DX server development. There is no need to use it if you will just build a DA/HDA-only server. Once the DLL is correctly initialized, the server can start DX services. These services include server configuration and the transfer of data from other OPC servers (DA and DX servers).

This function encapsulates:

- Loading of the server configuration information from the configuration file.
- Building of the DX hierarchy of the server address space.

Note that the DLL can update the server status. In fact, the manipulation of the configuration file is performed by the DLL. For example, if the file's path or the file itself is invalid, the status of the server will be "OPC_STATUS_NOCONFIG".

| Code | Description |
|---|---|
| 0 | The function succeeded. |
| -1 | Loading Data eXchange configuration failed (Invalid file path). |
| -2 | The DA hierarchy of the server address space is empty. At least one tag should be created. |
| -3 | A second call was rejected. |

**Table 25: Io_startdxmodules Error Codes**

**Comments**

This function is called after at least one call to **io_createtag** to allow configuration of DXConnections (we need at least one tag to be a target item).

**VB Wrapper**

```
Declare Function io_startdadxmodules Lib "DXServerDll.dll" () As Integer
```

**C# .NET**

```
int IO_StartDADXModules()
```

**Code example**

```
int intReturn = objOPCNetServerSDK.IO_StartDADXModules();
```

**VB .NET Wrapper**

```
Function IO_StartDADXModules () As Integer
```

**Code example**

```
Dim intResult As Integer = objOPCNetServerSDK.IO_StartDADXModules ()
```

**C++ .NET**

```
int IO_StartDADXModules ()
```

**Code example**

```
int intResult = objOPCNetServerSDK->IO_StartDADXModules()
```

## 2.13. io_setstatus

HRESULT io_setstatus ( [in] int State
, [in] LPWSTR StatusString)

**Description**

Set the current OPC server status.

| Parameter Name | Description |
|---|---|
| State | It should be one of these values:<br>SERVER_RUNNING<br>SERVER_SUSPEND<br>SERVER_FAILED<br>SERVER_NOCONFIG<br>SERVER_TEST<br>SERVER_COMMFAULT |
| StatusString | A string describing the server status. |

**Table 26: Io_setstatus Parameters**

| Code | Description |
|---|---|
| TRUE | The function succeeded. |
| FALSE | The function failed. The state is not valid or the callback object is not yet set (see the log file to get the exact error). |

**Table 27: Io_setstatus Error Codes**

**VB Wrapper**

```
Declare Function io_setstatus Lib "DXServerDll.dll" _
                    (ByVal state As Integer, _
                     ByVal hStatusString As Long) As Long
```

**VB Code example**

```
Dim state As Long
Dim intResult As Long
state = StrPtr("The server is running")
‘ 0: the server is running
intResult = io_setstatus(0, state)
```

**C# .NET**

```
int IO_SetStatus(int intState, string strStatusString)
intState=    ServerStatus.SERVER_COMMFAULT//6
                    |ServerStatus.SERVER_FAILED //2
                    |ServerStatus.SERVER_NOCONFIG     //3
                    |ServerStatus.SERVER_RUNNING      //0
                    |ServerStatus.SERVER_SUSPEND      //1
                    |ServerStatus.SERVER_TEST    //5
```

**Code example**

```
long lngResultSt = objOPCNetServerSDK.IO_SetStatus (ServerStatus.SERVER_RUNNING , "The server

is running.");
```

**VB .NET Wrapper**

```
Function IO_SetStatus(ByVal intState As Integer, ByVal strStatusString As String) As Integer

intState = ServerStatus.SERVER_COMMFAULT
Or ServerStatus.SERVER_FAILED
Or ServerStatus.SERVER_NOCONFIG
Or ServerStatus.SERVER_RUNNING
Or ServerStatus.SERVER_SUSPEND
Or ServerStatus.SERVER_TEST
```

**Code example**

```
Dim intResultSt As Integer = ObjOPCNetServerSDK.IO_setstatus(ServerStatus.SERVER_RUNNING,

"The server is running.")
```

**C++ .NET**

```
int IO_SetStatus(int intState, string^ strStatusString)
```

**Code example**

```
//Set Server Status
int intResultSt = objOPCNetServerSDK->IO_SetStatus(ServerStatus.SERVER_RUNNING, "The server is
running.");
```

## 2.14. io_updatetag

HRESULT io_updatetag ([in] BSTR StrTagName
, [in] VARIANT Value
, [in] WORD UshoQuality
, [in] FILETIME Timestamp)

**Description**

This function allows the DLL to update the value, ushoQuality and timestamp information of the specified item.

| Parameter Name | Description |
|---|---|
| StrTagName | The tag name to update its attributes. |
| Value | New value. |
| UshoQuality | New ushoQuality. |
| Timestamp | New timestamp. |

**Table 28: Io_updatetag Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| OPC_E_INVALIDITEMID | The ItemID is not syntactically valid. |
| OPC_E_UNKNOWNITEMID | The ItemID is not in the server address space. |
| E_FAIL | The function failed. The function was unsuccessful for this item. |
| S_xxx (success code)<br>E_xxx (error code) | Vendor specific information. |

**Table 29: Io_updatetag Error Codes**

**Please note that the « io_updateTag » method will succeed only if the tags you want to update were requested and added by an OPC client application.**
**This method should be called only when the driven mode is enabled, meaning the bDrivenMode parameter is set to true when calling io_setserverstatus function.**

### VB Wrapper

```
Declare Function VB_io_updatetag Lib "DXServerDll.dll" _
                    (ByVal StrTagName As String, _
                     ByVal Value As Variant, _
                     ByVal UshoQuality As Integer, _
                     Time As FILETIME) As Integer
```

**These exported functions were presented in the same order in which they should be called in the driver source code of a DA/DX Server.**

### C# .NET

```
int IO_UpdateTag(string strStrTagName, object objValue, UInt16 ushoUshoQuality, System.Runtime.InteropServices.ComTypes.FILETIME timestamp)
```

**Code example**

```
int intResult = objOPCNetServerSDK.IO_UpdateTag ("Tag.VT_I1",

objValue,OPC_UshoQuality.IO_OPC_USHOQUALITY_GOOD,

OPCNetServerSDK.OPCNetServerSDKManager.GetFileTimeFromDateTime(DateTime.Now));
```

### VB .NET Wrapper

```
Function IO_UpdateTag(ByVal strStrTagName As String, ByVal objValue As Object, ByVal ushoUshoQuality As UInt16, ByVal timestamp As System.Runtime.InteropServices.ComTypes.FILETIME) As Integer
```

**Code example**

```
Dim hResultSt As Integer = objOPCNetServerSDK.IO_UpdateTag ("Tag.VT_I1",

objValue,OPC_UshoQuality.IO_OPC_USHOQUALITY_GOOD,

OPCNetServerSDK.OPCNetServerSDKManager.GetFileTimeFromDateTime(DateTime.Now));
```

**C++ .NET**

int IO_UpdateTag (String^ strStrTagName, Object^ objValue, UInt16 ushoUshoQuality, System::Runtime::InteropServices::ComTypes::FILETIME timestamp)

**Code example**

```
int intResult = objOPCNetServerSDK::IO_UpdateTag("Tag.VT_I1",

count,OPC_UshoQuality::IO_OPC_USHOQUALITY_GOOD,

OPCNetServerSDK::OPCNetServerSDKManager.GetFileTimeFromDateTime(DateTime::Now));
```

## 2.15. io_getserverstatistic

HRESULT io_getserverstatistic ( [in] int StatId
, [out] int* Res)

**Description**

Some server statistics are imported to be accessible by the driver (server application based on this DLL).

Giving the statistics' identifier, the server can identify:

- The number of clients now connected to it.
- The number of groups created by OPC clients.
- The total number of added items to groups.
- The number of configured DX connections.

| Parameter Name | Description |
|---|---|
| StatId | The statistic ID. It must be one of these values:<br>NUM_CLIENTS<br>NUM_GROUPS<br>NUM_ITEMS<br>NUM_DXCONNS |
| Res | It is an out parameter that contains the requested value. |

**Table 30: Io_getserverstatistic Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. The value is stored in Res. |
| S_FALSE | The function failed. The StatId was invalid. |

**Table 31: Io_getserverstatistic Error Codes**

**Code example**

```
/*
  Get the current number of OPC connected clients.
*/

int hResult   = 1;
DWORD dwResult = 0;
bResult = io_getserverstatistic( NUM_CLIENTS, &dwResult );
```

**VB Wrapper**

```
Declare Function io_getserverstatistic Lib "DXServerDll.dll" _
                    (ByVal StatId As Integer, _
                     ByRef res As Long) As Integer
```

**C# .NET**

```
int IO_GetServerStatistic(int StatId,out UInt32 Res)
```

**Code example**

```
Int intResult = objOPCNetServerSDK.IO_GetServerStatistic (Statistics.NUM_CLIENTS,out intRes);
```

**VB .NET Wrapper**

```
Function IO_GetServerStatistic(ByVal StatId As Integer, ByRef Res As UInt32) As Integer
```

**Code example**

```
Dim intResult As Int = objOPCNetServerSDK.IO_GetServerStatistic (Statistics.NUM_CLIENTS, intRes)
```

**C++ .NET**

```
int IO_GetServerStatistic (int intStatId, UInt32 %intResult)
```

**Code example**

```
int res = objOPCNetServerSDK->IO_GetServerStatistic (Statistics::NUM_CLIENTS,  result);
```

## 2.16. io_getlastupdatetime

FILETIME  io_getlastupdatetime ( )

**Description**

Return the last update time of the server. This time is set at each server loop.

**VB Wrapper**

Declare Function io_getlastupdatetime Lib "DXServerDll.dll" () As FILETIME

**C# .NET**

System.Runtime.InteropServices.ComTypes.FILETIME IO_GetLastUpdateTime()

**Code example**

```
System.Runtime.InteropServices.ComTypes.FILETIME result =

objOPCNetServerSDK.IO_GetLastUpdateTime();
```

**VB .NET Wrapper**

Function IO_GetLastUpdateTime() As System.Runtime.InteropServices.ComTypes.FILETIME

**Code example**

```
Dim result As System.Runtime.InteropServices.ComTypes.FILETIME =

objOPCNetServerSDK.IO_GetLastUpdateTime()
```

**C++ .NET**

System::Runtime::InteropServices::ComTypes::FILETIME IO_GetLastUpdateTime ()

**Code example**

```
System::Runtime::InteropServices::ComTypes::FILETIME  result = objOPCNetServerSDK-

>IO_GetLastUpdateTime();
```

## 2.17. io_getserverstatus

LPCTSTR io_getserverstatus ([out] int *Status)

**Description**

Return the server status as a string and an integer.

**VB Wrapper**

Declare Function io_getserverstatus Lib "DXServerDll.dll" (ByRef Status As Integer) As String

**VB Code example**

```
Dim stat As String
stat = StrConv(io_getserverstatus(), vbFromUnicode)
MsgBox (stat)
```

**C# .NET**

```
String IO_GetServerStatus (out int iStatus)
```

**Code example**

```
int iStatus = 0;

String strResult = objOPCNetServerSDK.IO_GetServerStatus(out iStatus);
```

**VB .NET Wrapper**

```
Function IO_GetServerStatus(ByRef iStatus As Integer) As String
```

**Code example**

```
Dim iStatus As Integer = 0

Dim strResult As String = objOPCNetServerSDK.IO_getserverstatus(iStatus)
```

**C++ .NET**

```
String^ IO_GetServerStatus(UInt32 % iStatus)
```

**Code example**

```
int iStatus=0;

String ^strResult = objOPCNetServerSDK->IO_GetServerStatus(iStatus);
```

## 2.18. io_closeall

void io_closeall ( )

**Description**

Free all the DLL allocated resources and close the COM library. This function should be called only once at the end of server processing.

**Return Codes**
None

**VB Wrapper**
Declare Sub io_closeall Lib "DXServerDll.dll" ()

**C# .NET**
void IO_CloseAll()

**Code example**

objOPCNetServerSDK.IO_CloseAll();

**VB .NET Wrapper**
Sub IO_CloseAll()

**Code example**

objOPCNetServerSDK.IO_CloseAll()

**C++ .NET**
void IO_CloseAll();

**Code example**

objOPCNetServerSDK::IO_CloseAll();

## 2.19. io_registerserver

HRESULT io_registerserver ([in] LPWSTR ExePath, [in] LPWSTR strExeName, [in] LPWSTR strProductVendorInfo ,[in] bool IsService, [in] LPWSTR ServiceName, [in] LPWSTR ServiceParameters)

**Description**
Add the necessary server entries in the Windows registry.

| Parameter Name | Description |
|---|---|
| ExePath | The path of server executable. |
| strExeName | The name of the server executable. |
| strProductVendorInfo | The server application information. |
| IsService | Specifies whether the server will be registered as a service or |

| | |
|---|---|
| | not |
| ServiceName | The service name |
| ServiceParameters | The service parameters |

**Table 32: Io_registerserver Parameters**

**Comments**

The io_registerserver should be called after calling the io_initializeServer and the io_SetServerType methods. Otherwhise, the server registration will not succeed

**VB Wrapper**

Declare Function io_registerserver Lib ".\DXServerDll.dll" (ByVal wExePath As Long, ByVal ExeName As String, ByVal ProductVendorInfo As String, ByVal IsService As Boolean, ByVal ServiceName As String, ByVal ServiceParameters As String) As Long

**VB Code example**

```
Dim ExePath As Long
Dim result As Long
ExePath = StrPtr(App.Path & "\" & App.EXEName)
'register the server
result = io_registerserver( ExePath, "OPCVB.Simulator.1.exe", "Integration Objects' OPC DA HDA Server Simulator VB6; http://www.integrationobjects.com",False, "", "")
```

**C# .NET**

```
int IO_RegisterServer(string strServerPath, string strExeName , string strProductVendorInfo ,bool IsService, string ServiceName, string ServiceParameters)
```

**Code example**

```
String strServerPath= System.Reflection.Assembly.GetExecutingAssembly().Location;

int intResult = objOPCNetServerSDK.IO_RegisterServer(strServerPath,

"OPCServerSimulationCS2008.exe", "Integration Objects' OPC DA DX HDA Server Simulator Csharp;

http://www.integrationobjects.com",false,"","");
```

**VB .NET Wrapper**

```
Function IO_RegisterServer(ByVal strServerPath As String , ByVal strExeName As String , ByVal strProductVendorInfo As String ,ByVal IsService As Boolean, ByVal ServiceName As String, ByVal ServiceParameters As String) As Integer
```

**Code example**

```
Dim strServerPath As String = System.Reflection.Assembly.GetExecutingAssembly().Location
```

```
Dim intResult As Integer objOPCNetServerSDK.IO_RegisterServer(strServerPath, "
OPCServerSimulationVB.NET2008.exe", "Integration Objects' OPC DA DX HDA Server Simulator VB.Net;
http://www.integrationobjects.com", False, "", "")
```

**C++ .NET**

```
Int IO_RegisterServer(String^ strServerPath, String^ strExeName, String^
strProductVendorInfo,bool IsService, String^ ServiceName, String^ ServiceParameters)
```

**Code example**

```
String^ strServerPath = System::Reflection::Assembly::GetExecutingAssembly()->Location;
int intResult = objOPCNetServerSDK->IO_RegisterServer(strServerPath,
"OPCServerSimulationC++.Net2008.exe","Integration Objects' OPC DA DX HDA Server Simulator .Net
C++; http://www.integrationobjects.com",false,"","");
```

## 2.20. io_unregisterserver

HRESULT io_unregisterserver ([in] LPWSTR strExeName )

**Description**

Remove all server entries from the system registry.

| Parameter Name | Description |
|---|---|
| strExeName | The name of the server executable. |

**Table 33: Io_unregisterserver Parameters**

**VB Wrapper**

```
Declare Function io_unregisterserver Lib ".\DXServerDll.dll" (ByVal ExeName As String)
As Long
```

**C# .NET**

```
void IO_UnregisterServer(string strExeName);
```

**Code example**

```
objOPCNetServerSDK.IO_UnregisterServer("OPCServerSimulationCS2008.exe");
```

**VB .NET Wrapper**

```
Sub IO_UnregisterServer(ByVal strExeName As String) As Integer
```

**Code example**

```
objOPCNetServerSDK.IO_UnregisterServer(" OPCServerSimulationVB.NET2008.exe")
```

**C++ .NET**

```
int IO_UnregisterServer(String^ strExeName);
```

**Code example**

```
objOPCNetServerSDK::IO_UnregisterServer("OPCServerSimulationC++.Net2008.exe");
```

## 2.21. io_registerserverwithparameters

HRESULT io_registerserverwithparameters ([in] LPWSTR ExePath, [in] bool IsService, [in] GUID clsid, [in] DWORD dwServerType, [in] LPWSTR strprogid, [in] LPWSTR strProductVendorInfo, [in] LPWSTR strExeName, [in] LPWSTR ServiceName, [in] LPWSTR ServiceParameters)

**Description**

Add the necessary server entries in the Windows registry.

| Parameter Name | Description |
|---|---|
| ExePath | The path of server executable. |
| IsService | Specifies whether the server will be registered as a service or not |
| clsid | The server CLSID |
| dwServerType | The server type (DA, HDA…) |
| strprogid | The server ProgID |
| strProductVendorInfo | The server application information. |
| strExeName | The name of the server executable. |
| ServiceName | The service name |
| ServiceParameters | The service parameters |

**Table 34: Io_registerserverwithparameters Parameters**

**Comments**

The io_registerserverwithparameters is called without the need to call the io_initializeServer and the io_SetServerType methods.

**VB Wrapper**

Declare Function io_registerserverwithparameters Lib ".\DXServerDll.dll" (ByVal wExePath As Long, ByVal IsService As Boolean, ByVal clsid As GUID, ByVal dwServerType As Long, ByVal strprogid As String, ByVal ProductVendorInfo As String, ByVal ExeName As String, ,ByVal ServiceName As String, ByVal ServiceParameters As String) As Long

**VB Code example**

```
Dim ExePath As Long
Dim result As Long
ExePath = StrPtr(App.Path & "\" & App.EXEName)
'register the server
result = io_registerserverwithparameters ( ExePath, False, udtCLSID, 6,"
IOOPC.VB6.Simulation.1", "Integration Objects' OPC DA HDA Server Simulator
VB6;http://www.integrationobjects.com","OPCVB.Simulator.1.exe", "", "")
```

**C# .NET**

```
int IO_RegisterServerWithParameters(string ServerPath, bool IsService, Guid clsid, uint ServerType, string strprogid, string strVendorInfo, string strExeName, string ServiceName, string ServiceParameters)
```

**Code example**

```
Guid clsid = new Guid("20260278-2008-1112-1982-111219829CAA");

UInt32 dwSvrType =ServerType.OPC_DA_DX | ServerType.OPC_HDA;

String strServerPath= System.Reflection.Assembly.GetExecutingAssembly().Location;

int intResult = objOPCNetServerSDK.IO_RegisterServerWithParameters(strServerPath, false, clsid,

dwSvrType,"IntegrationObjects.DAHDASimulatorC#2008.1","Integration Objects' OPC DA DX HDA

Server Simulator Csharp; http://www.integrationobjects.com","OPCServerSimulationCS2008.exe" ,"","");
```

**VB .NET Wrapper**

```
Function IO_RegisterServerWithParameters(ByVal strServerPath As String, ByVal IsService As Boolean,ByVal clsid As Guid,ByVal  ServerType as UInteger , ByVal strprogid As String,ByVal strVendorInfo As String,ByVal strExeName As String,ByVal ServiceName As String, ByVal ServiceParameters As String) As Integer
```

**Code example**

```
Dim strServerPath As String = System.Reflection.Assembly.GetExecutingAssembly().Location
```

```
Dim clsid As New Guid("20260278-2008-1112-1982-111219829B31")
Dim dwSvrType As UInt32 = ServerType.OPC_DA_DX Or ServerType.OPC_HDA Or ServerType.OPC_DA
Dim intResult As Integer objOPCNetServerSDK.IO_RegisterServerWithParameters(strServerPath, False,clsid,dwSvrType,"IntegrationObjects.DAHDASimulatorVB2008.1 ,"Integration Objects' OPC DA DX HDA Server Simulator VB.Net; http://www.integrationobjects.com","OPCServerSimulationVB.NET2008.exe","", "")
```

**C++ .NET**

Int IO_ RegisterServerWithParameters(String^ strServerPath, bool IsService, Guid^ clsid, UInt32 ServerType, String^ strprogid, String^ strVendorInfo, String^ strExeName, String^ ServiceName, String^ ServiceParameters)

**Code example**

```
String^ strServerPath = System::Reflection::Assembly::GetExecutingAssembly()->Location;
Guid^ clsid = gcnew Guid("20260278-2008-1112-1982-111219829C08");
UInt32 dwSvrType = ServerType().OPC_DA_DX | ServerType().OPC_HDA | ServerType().OPC_DA;
int intResult = objOPCNetServerSDK->IO_RegisterServerWithParameters(strServerPath, false, clsid, dwSvrType,"IntegrationObjects.DAHDASimulatorC++.Net2008.1","Integration Objects' OPC DA DX HDA Server Simulator .Net C++; http://www.integrationobjects.com","OPCServerSimulationC++.Net2008.exe","","");
```

## 2.22. io_unregisterserverwithparameters

HRESULT io_unregisterserverwithparameters([in] GUID clsid, [in] DWORD dwServerType, [in] LPWSTR strprogid, [in] LPWSTR strExeName)

**Description**

Remove all server entries from the system registry.

| Parameter Name | Description |
|---|---|
| clsid | The server CLSID |
| dwServerType | The server type (DA, HDA…) |

| | |
|---|---|
| strprogid | The server ProgID |
| strExeName | The name of the server executable. |

**Table 35: Io_unregisterserverwithparameters Parameters**

**VB Wrapper**

Declare Function io_unregisterserverwithparameters Lib ".\DXServerDll.dll" (ByVal clsid As GUID, ByVal dwServerType As Long, ByVal strprogid As String ,ByVal ExeName As String) As Long

**C# .NET**

```
void IO_UnregisterServerWithParameters (Guid clsid, uint ServerType, string strprogid ,string strExeName);
```

**Code example**

```
Guid clsid = new Guid("20260278-2008-1112-1982-111219829CAA");

UInt32 dwSvrType =ServerType.OPC_DA_DX | ServerType.OPC_HDA;

objOPCNetServerSDK.IO_UnregisterServerWithParameters(clsid,dwSvrType,"IntegrationObjects.DAHDA

SimulatorC#2008.1",""OPCServerSimulationCS2008.exe");
```

**VB .NET Wrapper**

```
Sub IO_UnregisterServerWithParameters(ByVal clsid As Guid,ByVal ServerType as UInteger , ByVal strprogid As String ,ByVal strExeName As String) As Integer
```

**Code example**

```
Dim clsid As New Guid("20260278-2008-1112-1982-111219829B31")

Dim dwSvrType As UInt32 = ServerType.OPC_DA_DX Or ServerType.OPC_HDA Or

ServerType.OPC_DA

objOPCNetServerSDK.IO_UnregisterServerWithParameters(clsid,dwSvrType,"IntegrationObjects.DAHDA

SimulatorVB2008.1","OPCServerSimulationVB.NET2008.exe")
```

**C++ .NET**

```
void IO_ UnregisterServerWithParameters(Guid^ clsid, UInt32 ServerType, String^ strprogid ,String^ strExeName);
```

**Code example**

```
Guid^ clsid = gcnew Guid("20260278-2008-1112-1982-111219829C08");
```

```
UInt32 dwSvrType = ServerType().OPC_DA_DX | ServerType().OPC_HDA | ServerType().OPC_DA;
objOPCNetServerSDK::IO_UnregisterServerWithParameters(clsid,dwSvrType,"IntegrationObjects.DAHDASimulatorC++.Net2008.1","OPCServerSimulationC++.Net2008.exe");
```

## 2.23. io_requestdisconnect

void io_requestdisconnect ( [in] LPCSTR szReason )

**Description**

This function is called by the server to ask clients for a graceful shutdown. The server can mention the reason of such a request.

| Parameter Name | Description |
|---|---|
| szReason | The cause of shutdown. The server specifies why it has to shut down. |

**Table 36: Io_requestdisconnect Parameters**

**Return Codes**
None

**VB Wrapper**
Declare Sub io_requestdisconnect Lib "DXServerDll.dll" (ByVal Reason As String)

**C# .NET**
```
void IO_RequestDisconnect(string strReason)
```

**Code example**

```
objOPCNetServerSDK.IO_RequestDisconnect("Server request to shutdown");
```

**VB .NET Wrapper**
```
Sub IO_RequestDisconnect(ByVal strReason As String)
```

**Code example**

```
objOPCNetServerSDK.IO_RequestDisconnect("Server request to shutdown")
```

**C++ .NET**
```
void IO_RequestDisconnect(string^ strReason)
```

**Code example**

```
objOPCNetServerSDK::IO_RequestDisconnect("Server request to shutdown");
```

## 2.24. io_forceshutdown

void io_forceshutdown( )

**Description**
This function is called by the server to force clients to disconnect from it. It releases all created OPC server objects.

**Return Codes**
None

**VB Wrapper**
Declare Sub io_forceshutdown Lib "DXServerDll.dll" ()

**C# .NET**
```
void IO_ForceShutdown()
```

**Code example**

```
objOPCNetServerSDK.IO_ForceShutdown();
```

**VB .NET Wrapper**

```
Sub IO_ForceShutdown()
```

**Code example**

```
objOPCNetServerSDK.IO_ForceShutdown()
```

**C++ .NET**

```
void IO_ForceShutdown()
```

**Code example**

```
objOPCNetServerSDK::IO_ForceShutdown():
```

**The following exported functions are designed for HDA:**

## 2.25. io_activatehistorycallbacks

```
HRESULT   io_activatehistorycallbacks (
                    [in] IOREADCURRENTATTRIBUTEPROC ReadAttCallBack
                   ,[in] IOREADPROCESSEDPROC  ReadProcessedCallBack
                   ,[in] IOREADATTIMEPROC ReadAtTimeCallBack
                   ,[in] IOVALIDAGGREGATEPROC ValidAggCallBack
                   ,[in]IOINSERTINTOHISTORIANPROC InsertCallBack
                   ,[in]IOREMOVEFROMHISTORIANPROC   RemoveCallBack
                   ,[in] IOREADRAWDATUMPROC ReadRawCallBack)
```

**Description**

Set callbacks for history transactions (read, validate, insert, and remove).

| Parameter Name | Description |
|---|---|
| ReadAttCallBack | Pointer to ReadAttribute callback |
| ReadProcessedCallBack | Pointer to ReadProcessed callback |
| ReadAtTimeCallBack | Pointer to ReadAtTime callback |
| ValidAggCallBack | Pointer to ValidAggregate callback |

| | |
|---|---|
| InsertCallBack | Pointer to InsertIntoHistorian callback |
| RemoveCallBack | Pointer to RemoveFromHistorian callback |
| ReadRawCallBack | Pointer to ReadRaw callback |

**Table 37: Io_activatehistorycallbacks Parameters**

**VB Wrapper**

```
Declare Function VB_io_activatehistorycallbacks Lib "DXServerDll.dll" _
                (ByVal fnc1 As Long, _
                 ByVal fnc2 As Long, _
                 ByVal fnc3 As Long, _
                 ByVal fnc4 As Long, _
                 ByVal fnc5 As Long, _
                 ByVal fnc6 As Long, _
                 ByVal fnc7 As Long) As Integer
```

**VB Code example**

```
Dim result As Integer
result = VB_io_activatehistorycallbacks(
            AddressOf ReadCurrentAttribute
          , AddressOf ReadProcessed
          , AddressOf ReadAtTime
          , AddressOf ValidateAggregate
          , AddressOf InsertIntoHistorian
          , AddressOf RemoveFromHistorian
          , AddressOf ReadRawDatum)
```

**C# .NET**

```
void IO_activatehistorycallbacks(
OPCNetServerSDKManager.Del_IOREADCURRENTATTRIBUTEPROC
del_IOREADCURRENTATTRIBUTEPROC,
OPCNetServerSDKManager.Del_IOREADPROCESSEDPROC del_IOREADPROCESSEDPROC,
OPCNetServerSDKManager.Del_IOREADATTIMEPROC del_IOREADATTIMEPROC,
OPCNetServerSDKManager.Del_IOVALIDAGGREGATEPROC
del_IOVALIDAGGREGATEPROC,
OPCNetServerSDKManager.Del_IOINSERTINTOHISTORIANPROC
del_IOINSERTINTOHISTORIANPROC,
OPCNetServerSDKManager.Del_IOREMOVEFROMHISTORIANPROC
del_IOREMOVEFROMHISTORIANPROC,
OPCNetServerSDKManager.Del_IOREADRAWDATUMPROC del_IOREADRAWDATUMPROC)
```

**Code example**

```
objOPCNetServerSDK.IO_activatehistorycallbacks(del_IOREADCURRENTATTRIBUTEPROC,

del_IOREADPROCESSEDPROC, del_IOREADATTIMEPROC, del_IOVALIDAGGREGATEPROC,

del_IOINSERTINTOHISTORIANPROC, del_IOREMOVEFROMHISTORIANPROC,

Del_IOREADRAWDATUMPROC) ;
```

**VB .NET Wrapper**

```
Private Sub IO_activatehistorycallbacks(
ByVal del_IOREADCURRENTATTRIBUTEPROC As
OPCNetServerSDKManager.Del_IOREADCURRENTATTRIBUTEPROC,
ByVal del_IOREADPROCESSEDPROC As
OPCNetServerSDKManager.Del_IOREADPROCESSEDPROC,
ByVal del_IOREADATTIMEPROC As OPCNetServerSDKManager.Del_IOREADATTIMEPROC,
ByVal del_IOVALIDAGGREGATEPROC As
OPCNetServerSDKManager.Del_IOVALIDAGGREGATEPROC,
ByVal del_IOINSERTINTOHISTORIANPROC As
OPCNetServerSDKManager.Del_IOINSERTINTOHISTORIANPROC,
ByVal del_IOREMOVEFROMHISTORIANPROC As
OPCNetServerSDKManager.Del_IOREMOVEFROMHISTORIANPROC, _
ByVal del_IOREADRAWDATUMPROC As
OPCNetServerSDKManager.Del_IOREADRAWDATUMPROC)
```

**Code example**

```
objOPCNetServerSDK.IO_activatehistorycallbacks(del_IOREADCURRENTATTRIBUTEPROC,
del_IOREADPROCESSEDPROC, del_IOREADATTIMEPROC, del_IOVALIDAGGREGATEPROC,
del_IOINSERTINTOHISTORIANPROC, del_IOREMOVEFROMHISTORIANPROC, _
del_IOREADRAWDATUMPROC)
```

**C++ .NET**

```
void IO_ActivateHistoryCallbacks(
        OPCNetServerSDKManager.Del_IOREADCURRENTATTRIBUTEPROC
        del_IOREADCURRENTATTRIBUTEPROC,
        OPCNetServerSDKManager.Del_IOREADPROCESSEDPROC
        del_IOREADPROCESSEDPROC,
        OPCNetServerSDKManager.Del_IOREADATTIMEPROC
        del_IOREADATTIMEPROC,
        OPCNetServerSDKManager.Del_IOVALIDAGGREGATEPROC
        del_IOVALIDAGGREGATEPROC,
        OPCNetServerSDKManager.Del_IOINSERTINTOHISTORIANPROC
        del_IOINSERTINTOHISTORIANPROC,
        OPCNetServerSDKManager.Del_IOREMOVEFROMHISTORIANPROC
        del_IOREMOVEFROMHISTORIANPROC,
        OPCNetServerSDKManager.Del_IOREADRAWDATUMPROC
        del_IOREADRAWDATUMPROC)
```

**Code example**

```
objOPCNetServerSDK->IO_activatehistorycallbacks( del_IOREADCURRENTATTRIBUTEPROC,
del_IOREADPROCESSEDPROC, del_IOREADATTIMEPROC, Del_IOVALIDAGGREGATEPROC,
Del_IOINSERTINTOHISTORIANPROC, Del_IOREMOVEFROMHISTORIANPROC,
Del_IOREADRAWDATUMPROC) ;
```

## 2.26. io_hdasetitemaggregates

```
HRESULT io_hdasetitemaggregates ( [in] DWORD NumAgg
                                , [in] DWORD AggregateIDs [ ]
                                , [in] LPCSTR AggregateNames [ ]
                                , [in] LPCSTR AggregateDescs [ ]  )
```

**Description**

Set aggregates supported by the server. It should be called only once. The OPC Server Toolkit computes some OPC standard aggregates internally, which are: IO_OPCHDA_AVERAGE, IO_OPCHDA_TOTAL, IO_OPCHDA_COUNT, IO_OPCHDA_INTERPOLATIVE, IO_OPCHDA_START and IO_OPCHDA_END.

Although calculating these aggregates is made within the DLL, the developer has to specify if the server will support them (for more details see the Appendix B: Aggregates section).

| Parameter Name | Description |
|---|---|
| NumAgg | The number of server supported aggregates. |
| AggregateIDs | Array of the aggregate IDs. |
| AggregateNames | Array of the aggregate names. |
| AggregateDescs | Array of the aggregate descriptions. |

**Table 38: Io_hdasetitemaggregates Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| OPC_E_INVALIDITEMID | One or more invalid parameters were passed. |
| OPC_E_OUTOFMEMORY | Not enough memory. |
| E_FAIL | The function failed. |

**Table 39: Io_hdasetitemaggregates Error Codes**

**VB Wrapper**

```
Declare Function VB_io_hdasetitemaggregates Lib "DXServerDll.dll" _
                 (ByVal NumAgg As Integer, _
                  AggregateIDs() As Integer, _
                  AggregateNames() As String, _
                  AggregateDescs() As String) As Long
```

**C# .NET**

```
int IO_hdasetitemaggregates(UInt32 uintNumAgg,
   UInt32 []uintAggregateIDs,
   string[]strAggregateNames,
   string[]strAggregateDescs)
```

**Code example**

```
int intResult = objOPCNetServerSDK.IO_hdasetitemaggregates (uintNumAgg, uintAggregateIDs,
strAggregateNames, strAggregateDescs);
```

**VB .NET Wrapper**

```
Function IO_hdasetitemaggregates(ByVal uintNumAgg As UInteger,
                                 ByVal uintAggregateIDs As UInteger(),
                                 ByVal strAggregateNames As String(),
```

```
                                        ByVal strAggregateDescs As String()) As Integer
```

**Code example**

```
Dim intResult As Integer = objOPCNetServerSDK.IO_hdasetitemaggregates (uintNumAgg,
uintAggregateIDs, strAggregateNames, strAggregateDescs)
```

**C++ .NET**

```
int IO_hdasetitemaggregates(UInt32 uintNumAgg, array<UInt32 > ^uintAggregateIDs,
array<String ^ > ^strAggregateNames, array<String ^> ^strAggregateDescs)
```

**Code example**

```
int intResult = objOPCNetServerSDK->IO_hdasetitemaggregates (uintNumAgg, uintAggregateIDs,
strAggregateNames, strAggregateDescs);
```

## 2.27. io_hdasetitemattributes

```
HRESULT io_hdasetitemattributes( [in] DWORD   NumAtt,
                                 [in] DWORD   UintAttributeIDs[ ],
                                 [in] LPCSTR   AttributeNames[ ],
                                 [in] LPCSTR   AttributeDescs[ ],
                                 [in] VARTYPE AttributeTypes[ ],
                                 [in] BOOL       AttributesArchived[ ])
```

**Description**

Set the server supported attributes. It should be called **only once**.

| Parameter Name | Description |
|---|---|
| NumAtt | The number of attributes supported by the server. |
| UintAttributeIDs | Array of the attribute IDs. |
| AttributeNames | Array of the attribute names. |
| AttributeDescs | Array of the attribute descriptions. |
| AttributeTypes | Array of the attribute types (VARTYPE). |
| AttributesArchived | Array of Boolean values, specifying for each attribute if it is archived by the historian or only available to the client as a "current value".<br>All these arrays have a size of NumAttr. |

**Table 40: Io_hdasetitemattributes Parameters**

For more details about UintAttributeIDs, AttributeNames, AttributeDescs and AttributeTypes values, see Appendix B: Attributes section.

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| OPC_E_INVALIDITEMID | One or more invalid parameters were passed. |
| OPC_E_OUTOFMEMORY | Not enough memory. |
| E_FAIL | The function failed. |

**Table 41: Io_hdasetitemattributes Error Codes**

**VB Wrapper**

```
Declare Function VB_io_hdasetitemattributes Lib "DXServerDll.dll" _
                 (ByVal NumAtt As Integer, _
                  UintAttributeIDs() As Integer,  _
                  AttributeNames() As String, _
                  AttributeDescs() As String, _
                  AttributeTypes() As Integer, _
                  AttributesArchived() As Integer) As Long
```

**C# .NET**

```
int IO_hdasetitemattributes(UInt32 uintNumAttr,
    UInt32[]uintTagAttrID,
    string[] strAttributeNames,
    string[] strAttributeDescs,
    UInt16[] ushoAttributeType,
    Int32[] uintIsAttributeArchived)
```

**Code example**

```
Int intResult = objOPCNetServerSDK.IO_hdasetitemattributes(uintNumAttr , uintTagAttrID,
strAttributeNames, strAttributeDescs, ushoAttributeType, uintIsAttributeArchived)
```

**VB .NET Wrapper**

```
Function IO_hdasetitemattributes(ByVal uintNumAttr As UInt32,
                                 ByVal uintTagAttrID As UInt32(),
                                 ByVal strAttributeNames As String(),
                                 ByVal strAttributeDescs As String(),
                                 ByVal ushoAttributeType As UInt16(),
                                 ByVal uintIsAttributeArchived As Int32()) As Integer
```

**Code example**

```
Dim intResult As Integer = objOPCNetServerSDK->IO_hdasetitemattributes(uintNumAttr , uintTagAttrID,
strAttributeNames, strAttributeDescs, ushoAttributeType, uintIsAttributeArchived)
```

**C++ .NET**

```
int IO_ hdasetitemattributes (UInt32 uintNumAttr,
                              array<UInt32 > ^ uintTagAttrID,
                              array<String ^ > ^ strAttributeNames,
                              array<String ^> ^ strAttributeDescs,
                              array<UInt16 > ^ushoAttributeType,
                              array<UInt32 > ^ uintIsAttributeArchived)
```

**Code example**

```
int intResult = objOPCNetServerSDK.IO_hdasetitemattributes(uintNumAttr , uintTagAttrID,
strAttributeNames, strAttributeDescs, ushoAttributeType, uintIsAttributeArchived)
```

## 2.28. io_hdasethdaitems

```
HRESULT io_hdasethdaitems ([in] DWORD       numItem,
                           , [in] LPCSTR    mItemIDs[ ]
                           , [in] VARTYPE   mDataTypes[ ]
                           , [in] LPCSTR    mDescs[ ]
                           , [in] LPCSTR    mEUs[ ]
                           , [in] BOOL      mUsables[ ]
                           , [out] HANDLE **hHandleItems )
```

**Description**

Set the HDA server tags. It should be called **only once**.

| Parameter Name | Description |
|---|---|
| numItem | The number of tags handled by the OPC server. |
| mItemIDs | Array of the tags IDs. |
| mDataTypes | Array of tags data types. |
| mDescs | Array of tags descriptions. |
| mEUs | Array of tags engineering units (EU). |
| mUsables | Array of Boolean values specifying whether each tag is active or not. The developer may need to introduce some tag names just for browsing purposes.<br>If mUsables[x] == TRUE, the DLL assigns a valid handle (HANDLE) to mItemIDs[x]. |

| | |
|---|---|
| | Else it assigns an INVALID_HANDLE_VALUE. |
| handleItems | Array of tag handles assigned by the DLL.<br>A given valid handle is unique for each 'usable' tag. Once returned, these handles should be kept by the server since they uniquely reference the tags and they will be used to communicate with the server. |

**Table 42: Io_hdasethdaitems Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| OPC_E_INVALIDITEMID | One or more invalid parameters were passed. |
| OPC_E_OUTOFMEMORY | Not enough memory. |
| E_FAIL | The function failed. |

**Table 43: Io_hdasethdaitems Error Codes**

**VB Wrapper**

```
Declare Function VB_io_hdasethdaitems Lib "DXServerDll.dll" _
                 (ByVal NumItem As Integer, _
                  ItemIDs() As String, _
                  DataTypes() As Integer, _
                  Descs() As String, _
                  EUs() As String, _
                  Usables() As Integer, _
                  handleItems() As Long) As Long
```

**C# .NET**

```
int IO_hdasethdaitems(UInt32 NumItem,
                      [In]string[] mItemID,
                      UInt16[] mDataType,
                      [In]string[] mDesc,
                      [In]string[] mEU,
                      Int32[] mUsable,
                      out IntPtr handleItems)
```

**Code example**

```
objOPCNetServerSDK.IO_hdasethdaitems(TagsNumber, TagIDs, TagVarTypes, TagDescriptors, TagEngUnits, TagsUsable, out TagsHandles);
```

**VB .NET Wrapper**

```
Function IO_hdasethdaitems(ByVal uintNumItem As UInt32,
                           ByVal strItemIDs As String(),
                           ByVal ushoDataTypes As UInt16(), <[In]()>
                           ByVal strDescs As String(), <[In]()>
                           ByVal strEUs As String(),
                           ByVal uintUsables As Int32(),
                           ByRef handleItems As IntPtr) As Integer
```

**Code example**

```
Dim intResult As Integer      objOPCNetServerSDK.IO_hdasethdaitems(uintNumItem, TagIDs,
TagVarTypes, TagDescriptors, TagEngUnits, TagsUsable, out handleItems);
```

**C++ .NET**

```
int IO_hdasethdaitems (UInt32 uintNumAttr,
                       array<String ^ > ^ strItemIDs,
                       array<UInt16> ushoDataTypes,
                       array<String ^> ^strDescs,
                       array<String ^ > ^strEUs,
                       array<UInt32 > uintUsables,
                       IntPtr %handleItems)
```

**Code example**

```
int intResult = objOPCNetServerSDK.IO_hdasethdaitems(uintNumAttr, strItemIDs, ushoDataTypes,
strDescs, strEUs, uintUsables, handleItems);
```

## 2.29. io_hdaAddhdaitems

```
HRESULT  io_hdaAddhdaitems (  [in] DWORD       numItem,
                            , [in] LPCSTR      mItemIDs[ ]
                            , [in] VARTYPE  mDataTypes[ ]
                            , [in] LPCSTR      mDescs[ ]
                            , [in] LPCSTR      mEUs[ ]
                            , [in] BOOL        mUsables[ ]
                            , [out] HANDLE **hHandleItems )
```

**Description**

Set the HDA server tags after the first initialization. It can be called more than once. If the tag already exists in the address space, it will be updated with the given data types, description and engineering units.

| Parameter Name | Description |
|---|---|
| numItem | The number of tags handled by the OPC server. |
| mItemIDs | Array of the tags IDs. |

| | |
|---|---|
| mDataTypes | Array of tags data types. |
| mDescs | Array of tags descriptions. |
| mEUs | Array of tags engineering units (EU). |
| mUsables | Array of Boolean values specifying whether each tag is active or not. The developer may need to introduce some tag names just for browsing purposes.<br>If mUsables[x] == TRUE, the DLL assigns a valid handle (HANDLE) to mItemIDs[x].<br>Else it assigns an INVALID_HANDLE_VALUE. |
| handleItems | Array of tag handles assigned by the DLL.<br>A given valid handle is unique for each 'usable' tag. Once returned, these handles should be kept by the server since they uniquely reference the tags and they will be used to communicate with the server. |

**Table 44: Io_hdaAddhdaitems Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| OPC_E_INVALIDITEMID | One or more invalid parameters were passed. |
| OPC_E_OUTOFMEMORY | Not enough memory. |
| E_FAIL | The function failed. |

**Table 45: Io_ hdaAddhdaitems Error Codes**

**C# .NET**

```
int IO_HDAAAddHDAItems(UInt32 uintNumAttr,
                       [In]string[] strItemIDs,
                       UInt16[] ushoDataTypes,
                       [In]string[] strDescs,
                       [In]string[] strEUs,
                       Int32[] uintUsable,
                       out IntPtr handleItems)
```

**Code example**

```
Int intResult = objOPCNetServerSDK.IO_HDAAAddHDAItems(uintNumAttr, strItemIDs, ushoDataTypes,
strDescs, strEUs, uintUsable, out handleItems);
```

**VB .NET Wrapper**

```
Function IO_HDAAAddHDAItems(ByVal uintNumAttr As UInt32,
                            <[In]()> ByVal mItemID As String(),
                            ByVal mDataType As UInt16(),
                            <[In]()> ByVal mDesc As String(),
                            <[In]()> ByVal mEU As String(),
                            ByVal mUsable As Int32(),
                            ByRef HandleItems As IntPtr) As Integer
```

**Code example**

```
Dim intResult As Integer      = objOPCNetServerSDK.IO_HDAAAddHDAItems(uintNumItem, TagIDs,
TagVarTypes, TagDescriptors, TagEngUnits, TagsUsable, out handleItems);
```

**C++ .NET**

```
int IO_HDAAAddHDAItems(UInt32 uintNumAttr,
                       array<String ^ > ^ strItemIDs,
                       array<UInt16> ushoDataTypes,
                       array<String ^> ^strDescs,
                          array<String ^ > ^strEUs,
                       array<UInt32 > uintUsables,
                       IntPtr %handleItems)
```

**Code example**

```
Int intResult = objOPCNetServerSDK->IO_HDAAAddHDAItems(uintNumAttr, strItemIDs, ushoDataTypes,
strDescs, strEUs, uintUsables, handleItems);
```

## 2.30. io_hdaRemoveAllhdaitems

HRESULT  io_ hdaRemoveAllhdaItems ()

**Description**

Remove the all the tags from the address space of the HDA OPC Server.

**C# .NET**

```
int IO_HDARemoveAllHDAItems()
```

**Code example**

```
int intResult = objOPCNetServerSDK.IO_ HDARemoveAllHDAItems();
```

**VB .NET Wrapper**

```
Function IO_ HDARemoveAllHDAItems () As Integer
```

**Code example**

```
Dim intResult As Integer = objOPCNetServerSDK.IO_ HDARemoveAllHDAItems()
```

**C++ .NET**

```
int IO_ HDARemoveAllHDAItems()
```

**Code example**

```
int intResult = objOPCNetServerSDK->IO_ HDARemoveAllHDAItems();
```

## 2.31. io_hdaRemovehdaitems

```
HRESULT  io_hdaRemovehdaitems (DWORD    NumItem,
                               LPCSTR   mItemIDs[]))
```

**Description**

Remove the given tags from the address space of the HDA OPC Server.

| Parameter Name | Description |
|---|---|
| NumItem | The number of tags that will be removed. |
| mItemIDs | Array of the tags IDs. |

**Table 46: Io_hdaRemovehdaitems Parameters**

**C# .NET**

```
int IO_HDARemoveHDAItems(uint uintNumItem, string[] strItemIDs)
```

**Code example**

```
string[] strItemIDs ={"MyTag"};

int intResult = objOPCNetServerSDK.IO_HDARemoveHDAItems(1, strItemIDs);
```

**VB .NET Wrapper**

```
Function IO_HDARemoveHDAItems(ByVal uintNumItem As UInteger,
                              ByVal strItemIDs As String()) As Integer
```

**Code example**

```
Dim strStrTagNames As String() = {"MyTag"}
Dim intResult As Integer = objOPCNetServerSDK.IO_HDARemoveHDAItems(1, strStrTagNames)
```

**C++ .NET**

```
int IO_HDARemoveHDAItems(UInt32 uintNumAttr,
                         array<String ^ > ^ strItemIDs)
```

**Code example**

```
array<String ^ > ^ strItemIDs;
int intResult = objOPCNetServerSDK->IO_HDAAAddHDAItems(uintNumAttr, strItemIDs);
```

## 2.32. io_hdasetupdatemethods

HRESULT io_hdasetupdatemethods ( [in] DWORD NumUpdate,
, [in] WORD Capabilities [ ] )

**Description**

Set the server update capabilities. It should be called **only once**.

| Parameter Name | Description |
|---|---|
| NumUpdate | Number of update methods supported by the server. The range is [0...5]. |
| Capabilities | List of the supported methods:<br>IO_OPCHDA_INSERTCAP<br>IO_OPCHDA_REPLACECAP<br>IO_OPCHDA_INSERTREPLACECAP<br>IO_OPCHDA_DELETERAWCAP<br>IO_OPCHDA_DELETEATTIMECAP |

**Table 47: Io_hdasetupdatemethods Parameters**

(See Appendix B: **Capabilities section** for more details about supported methods)

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| OPC_E_INVALIDITEMID | One or more invalid parameters were passed. |
| OPC_E_OUTOFMEMORY | Not enough memory. |
| E_FAIL | The function failed. |

**Table 48: Io_hdasetupdatemethods Error Codes**

**Code example**

```
/*
  We pass five update methods to io_hdasetupdatemethods.
*/

WORD m_capabilities[5] = { IO_OPCHDA_INSERTCAP,
                           IO_OPCHDA_REPLACECAP,
```

```
                                    IO_OPCHDA_INSERTREPLACECAP,
                                    IO_OPCHDA_DELETERAWCAP,
                                    IO_OPCHDA_DELETEATTIMECAP};
HRESULT Dllhr = io_hdasetupdatemethods(5,m_capabilities);
```

**VB Wrapper**

```
Declare Function VB_io_hdasetupdatemethods Lib "DXServerDll.dll" _
                    (ByVal NumUpdate As Integer, _
                     capabilities () As Integer) As Long
```

**C# .NET**

```
int IO_hdasetupdatemethods(UInt32 uintNumUpdate,
                                    UInt16[] ushohdacapabilities)
```

**Code example**

```
int intResult = objOPCNetServerSDK. IO_hdasetupdatemethods(uintNumUpdate, ushohdacapabilities);
```

**VB .NET Wrapper**

```
Function IO_hdasetupdatemethods(ByVal uintNumUpdate As UInt32,
                                                ByVal ushoHDAcapabilities As UInt16())
                                                As Integer
```

**Code example**

```
Dim intResult As Integer = objOPCNetServerSDK. IO_HDARemoveHDAItems.IO_hdasetupdatemethods
(uintNumUpdate, ushoHDAcapabilities)
```

**C++ .NET**

```
int IO_hdasetupdatemethods(UInt32 uintNumUpdate,
                                    array<UInt16 > ^ ushoHDAcapabilities)
```

**Code example**

```
int intResult = objOPCNetServerSDK->IO_HDAAAddHDAItems(uintNumUpdate, ushoHDAcapabilities);
```

## 2.33. io_hdaaggregatesComputedAtServerLevel

VOID
io_hdaaggregatesComputedAtServerLevel(BOOL bShouldbecomputedAtServerLevel)

**Description**

By default, the basic aggregates are calculated inside the toolkit. However, if the OPC Server needs to support more aggregates, developers can call this method and the Readprocessed callback will be activated. It should be called **only once**.

| Parameter Name | Description |
|---|---|
| bShouldbecomputedAtServerLevel | True: if the aggregates are computed in the server level.<br>False: if the aggregates are computed in the OPC Server Toolkit |

**Table 49 : Io_hdaaggregatesComputedAtServerLevel Parameters**

**C# .NET**

```
void IO_hdaaggregatesComputedAtServerLevel(bool bShouldbecomputedAtServerLevel)
```

**Code example**

```
objOPCNetServerSDK.IO_hdaaggregatesComputedAtServerLevel(false);
```

## 2.34. io_isBuffer

VOID
io_isBuffer(BOOL bIsBuffer)

**Description**

By default, the UpdateItem callback is fired in order to update the OPC tags values at server level. However, if the OPC Server needs to behave as a buffer, developers can call this method and set the bIsBuffer parameter to true. It should be called **only once**.
An OPC Server acting like a buffer means that the OPC tags values cannot be updated at server level, it can be changed only by the OPC Write Item operation via an OPC Client.

| Parameter Name | Description |
|---|---|
| bIsBuffer | True: if the OPC Server Tags values are static.<br>False: if OPC Server Tags values are updated. |

**Table 50 : Io_IsBuffer Parameters**

**C# .NET**

```
void IO_isBuffer(bool bisBuffer)
```

**Code example**

```
objOPCNetServerSDK.IO_isBuffer(true);
```

# 4. Callbacks Specification

## 4.1. Overview

This section describes all callbacks needed to allow communication between this toolkit and the device. The developer's role is to focus on this part.

### 4.1.1. Definitions

| OPC DA/DX | |
|---|---|
| **Common to VB & VC++** | |
| VOID | ValidateItem(BSTR, BSTR, VARTYPE, DWORD *, VARTYPE *, HRESULT *) |
| VOID | GetErrorString( DWORD, DWORD, BSTR*) |
| VOID | ReadItem( BSTR, VARIANT*, FILETIME *, WORD*, HRESULT* ) |
| VOID | WriteItem(BSTR, VARIANT, FILETIME *, WORD*, HRESULT* ) |
| VOID | WriteVQT( BSTR, VARIANT&, BOOL, FILETIME*, BOOL, WORD*, HRESULT* ) |
| VOID | ValidateAndInsert(BSTR, HRESULT* ) |
| VOID | OnClientConnect(HRESULT*) |
| VOID | OnClientDisconnect(HRESULT* ) |
| VOID | SvrCanUnloadNow(HRESULT* ) |
| **VC++** | |
| VOID | UpdateItem(BSTR, VARIANT*, FILETIME *, WORD*, HRESULT* ) |
| VOID | QueryAvailableProperties(BSTR, DWORD *, DWORD **, LPWSTR **, VARTYPE **, HRESULT *) |
| VOID | GetItemProperties(BSTR, DWORD, DWORD *, VARIANT **, HRESULT **, HRESULT *) |

| | |
|---|---|
| VOID | LookupItemProperties(BSTR, DWORD, DWORD **, HRESULT *) |
| **VB** | |
| VOID | VB_NumberOfItemProperties(BSTR, DWORD*) |
| VOID | VB_QueryAvailableProperties(BSTR, DWORD, SAFEARRAY **, SAFEARRAY **, SAFEARRAY **, HRESULT *) |
| VOID | VB_GetItemProperties(BSTR, DWORD, SAFEARRAY **, SAFEARRAY **, SAFEARRAY **, HRESULT *) |
| VOID | VB_LookupItemProperties(BSTR, DWORD, SAFEARRAY **, HRESULT *) |
| **C#** | |
| VOID | ValidateItem(string,string,UInt16,out UInt32,out UInt16,out IntPtr) |
| VOID | GetErrorString(UInt32,UInt32,out IntPtr ) |
| VOID | ReadItem(String, out object, out FILETIME, out UInt16,out IntPtr) |
| VOID | WriteItem(String, object ,out FILETIME ,out UInt16 , out IntPtr) |
| VOID | WriteVQT (String , ref object , bool , out FILETIME , bool , out UInt16 ,out IntPtr) |
| VOID | ValidateAndInsert(String, out IntPtr) |
| VOID | OnClientConnect(out IntPtr) |
| VOID | OnClientDisconnect(out IntPtr) |
| VOID | SvrCanUnloadNow(out IntPtr ) |
| VOID | UpdateItem(String , out object , out FILETIME , out UInt16 ,out IntPtr) |
| VOID | QueryAvailableProperties(String, Out UInt32,out UInt32[], out IntPtr[],out UInt16[], out IntPtr hr) |
| VOID | GetItemProperties(String,UInt32,IntPtr,out object[], out IntPtr[], out IntPtr hr) |
| VOID | LookupItemIDs(String,UInt32,out UInt32[],out IntPtr) |
| **VB.Net** | |
| VOID | ValidateItem(ByVal As String, ByVal As String, ByVal As UInt16, ByRef As UInt32, ByRef As UInt16, ByRef As IntPtr) |
| VOID | GetErrorString(ByVal As UInt32, ByVal As UInt32, ByRef As IntPtr) |
| VOID | ReadItem(ByVal As [String], ByRef As Object, ByRef As FILETIME, ByRef As |

| | |
|---|---|
| | UInt16, ByRef As IntPtr) |
| VOID | WriteItem(ByVal As [String], ByVal As Object, ByRef As FILETIME, ByRef As UInt16, ByRef As IntPtr) |
| VOID | WriteVQT (ByVal As [String], ByRef As Object, ByVal As Boolean, ByRef As FILETIME, ByVal As Boolean, ByRef As UInt16, ByRef As IntPtr) |
| VOID | ValidateAndInsert(ByVal As [String], ByRef As IntPtr) |
| VOID | OnClientConnect(ByRef As IntPtr) |
| VOID | OnClientDisconnect(ByRef As IntPtr) |
| VOID | SvrCanUnloadNow(ByRef As IntPtr) |
| VOID | UpdateItem(ByVal As [String], ByRef As Object, ByRef As FILETIME, ByRef As UInt16, ByRef As IntPtr) |
| VOID | QueryAvailableProperties(ByVal As [String], ByRef As UInt32, ByRef As UInt32(), ByRef As IntPtr(), ByRef As UInt16(), ByRef As IntPtr) |
| VOID | GetItemProperties(ByVal As [String], ByVal As UInt32, ByVal As IntPtr, ByRef As Object(), ByRef As IntPtr(), ByRef As IntPtr) |
| VOID | LookupItemIDs(ByVal As [String], ByVal As UInt32, ByRef As UInt32(), ByRef As IntPtr) |
| **C++.Net** | |
| VOID | ValidateItem(String^,String^,UInt16,UInt32 %,UInt16 %, IntPtr %) |
| VOID | GetErrorString(UInt32,UInt32,IntPtr %) |
| VOID | ReadItem(String^ , Object^ % , FILETIME % , UInt16 % , IntPtr %) |
| VOID | WriteItem(String^ ,Object^ ,FILETIME % ,UInt16 % , IntPtr %) |
| VOID | WriteVQT (String^ ,Object^ % , bool ,FILETIME % , bool , UInt16 % , IntPtr %) |
| VOID | ValidateAndInsert(String^, IntPtr %) |
| VOID | OnClientConnect( IntPtr %) |
| VOID | OnClientDisconnect( IntPtr %) |
| VOID | SvrCanUnloadNow( IntPtr %) |
| VOID | UpdateItem(String^ , Object^ % , FILETIME % , UInt16 % ) |
| VOID | QueryAvailableProperties(String^, UInt32 %, array<UInt32,1>^ %, array<IntPtr,1>^ %, array<UInt16,1>^ %, IntPtr %) |
| VOID | GetItemProperties( String^, UInt32, IntPtr, array<Object^,1>^ % ,array<IntPtr,1>^ %, IntPtr %) |

**Table 51: OPC DA/DX Callbacks Definition**

| | |
|---|---|
| **OPC HDA** | |
| **Common to VB & VC++** | |
| VOID | ReadCurrentAttribute (HANDLE, DWORD, VARIANT*, HRESULT*) |
| VOID | ReadProcessed (HANDLE , DWORD , FILETIME * , FILETIME * , VARIANT * ,DWORD* , FILETIME *, FILETIME *, HRESULT*) |
| VOID | ReadAtTime (HANDLE, FILETIME *, VARIANT *, DWORD *, HRESULT*) |
| VOID | ValidAggregate (HANDLE, DWORD, HRESULT*) |
| VOID | InsertIntoHistorian (HANDLE, FILETIME, double, DWORD, long, HRESULT*) |
| VOID | RemoveFromHistorian (HANDLE, FILETIME*, FILETIME*, int, HRESULT*) |
| **VC++** | |
| VOID | ReadRaw (HANDLE , FILETIME * , FILETIME * , BOOL, long*, VARIANT **, DWORD** , FILETIME **, HRESULT*) |
| **VB** | |
| VOID | VB_ReadRaw( HANDLE , FILETIME * , FILETIME * , BOOL, long*, SAFEARRAY **, SAFEARRAY ** , SAFEARRAY **, HRESULT*) |
| **C#** | |
| VOID | ReadCurrentAttribute (int, UInt32, out object, out IntPtr) |
| VOID | ReadProcessed ( int hItem, UInt32 , ref FILETIME , ref FILETIME , out object , out UInt32 , out FILETIME , out FILETIME, out IntPtr) |
| VOID | ReadAtTime ( int, ref FILETIME, out object, out UInt16, out IntPtr) |
| VOID | ValidAggregate (int,UInt32,out Int32) |
| VOID | InsertIntoHistorian (int,FILETIME,object,UInt16,int,out IntPtr) |
| VOID | RemoveFromHistorian ( int, ref FILETIME, ref FILETIME, int,out IntPtr) |
| VOID | ReadRawDatum ( int , ref FILETIME, ref FILETIME , Int32 , ref Int32 , out object[] , out UInt32[] , out FILETIME[] , out IntPtr) |
| **VB.Net** | |

| | |
|---|---|
| VOID | ReadCurrentAttribute (ByVal As Integer, ByVal As UInt32, ByRef As Object, ByRef As IntPtr) |
| VOID | ReadProcessed( ByVal As Integer, ByVal As UInt32, ByRef As FILETIME, ByRef As FILETIME, ByRef As Object, ByRef qual As UInt32,ByRef As FILETIME, ByRef As FILETIME, ByRef As IntPtr) |
| VOID | ReadAtTime (ByVal As Integer, ByRef As FILETIME, ByRef As Object, ByRef As UInt16, ByRef As IntPtr) |
| VOID | ValidAggregate (ByVal As Integer, ByVal As UInt32, ByRef As Int32) |
| VOID | InsertIntoHistorian (ByVal As Integer, ByVal As FILETIME, ByVal As Object, ByVal As UInt16, ByVal As Integer, ByRef As IntPtr) |
| VOID | RemoveFromHistorian (ByVal As Integer, ByRef As FILETIME, ByRef As FILETIME, ByVal As Integer, ByRef As IntPtr) |
| VOID | ReadRawDatum( ByVal As Integer, ByRef As FILETIME, ByRef As FILETIME, ByVal As Int32, ByRef As Int32, ByRef As Object(), ByRef As UInt32(), ByRef As FILETIME(), ByRef As IntPtr) |
| **C++.Net** | |
| VOID | ReadCurrentAttribute (int, UInt32, Object^ %, IntPtr %) |
| VOID | ReadProcessed ( int , UInt32,FILETIME % , FILETIME %, Object^ % , UInt32 %, FILETIME %, FILETIME %, IntPtr %) |
| VOID | ReadAtTime (int, FILETIME %, Object^ %, UInt16 %, IntPtr %) |
| VOID | ValidAggregate (int, UInt32, Int32 %) |
| VOID | InsertIntoHistorian (int, FILETIME, Object^, UInt16, int, IntPtr %) |
| VOID | RemoveFromHistorian (int, FILETIME %, FILETIME %, int, IntPtr %) |
| VOID | ReadRawDatum ( int , FILETIME % , FILETIME % , Int32 , Int32 % , array<Object^,1>^ % ,array<UInt32,1>^ % ,FILETIME,1>^ % , IntPtr %) |

**Table 52: OPC HDA Callbacks Definition**

The following table matches these callbacks with the corresponding pointers described below.

| **OPC DA/DX & general callbacks functions** | |
|---|---|
| **Pointer** | **Referenced Object (callback function)** |
| IOSVRCANUNLOADNOWPROC | SvrCanUnloadNow |
| IOGETERRORSTRINGPROC | GetErrorString |

| | |
|---|---|
| IOWRITEITEMPROC | WriteItem |
| IOREADITEMPROC | ReadItem |
| IOUPDATEITEMPROC | UpdateItem |
| IOWRITEVQTPROC | WriteVQT |
| IOVALIDATEITEMPROC | ValidItem |
| IOVALIDATEANDADDPROC | ValidateAndInsert |
| IOQUERYAVAILABLEPROPERTIESPROC | QueryAvailableProperties |
| IOGETITEMPROPERTIESPROC | GetItemProperties |
| IOLOOKUPITEMIDSPROC | LookupItemProperties |
| IOONCLIENTCONNECTPROC | OnClientConnect |
| IOONCLIENTDISCONNECTPROC | OnClientDisconnect |
| (For VB) | |
| VB_NUMBEROFITEMPROPERTIESPROC | VB_NumberOfProperties |
| VB_IOQUERYAVAILABLEPROPERTIESPROC | VB_QueryAvailableProperties |
| VB_IOGETITEMPROPERTIESPROC | VB_GetItemProperties |
| VB_IOLOOKUPITEMIDSPROC | VB_LookupItemProperties |
| **OPC HDA** | |
| IOREADCURRENTATTRIBUTEPROC | ReadAttribute |
| IOREADPROCESSEDPROC | ReadProcessed |
| IOREADATTIMEPROC | ReadAtTime |
| IOVALIDAGGREGATEPROC | ValidAggregate |
| IOINSERTINTOHISTORIANPROC | InsertIntoHistorian |
| IOREMOVEFROMHISTORIANPROC | RemoveFromHistorian |
| IOREADRAWDATUMPROC | ReadRaw |
| (For VB) | |
| VB_IOREADRAWDATUMPROC | VB_ReadRaw |

**Table 53: Callbacks Mapping**

## 4.2. Description

These callbacks' implementations are vendor specific. Thus, the server application developer should respect the specification of each function. It should return the appropriate error code to the DLL whenever one of these functions is invoked.

## 4.3. OnClientConnect

VOID OnClientConnect ([out] HRESULT* hr )

**Description**

This function will be called whenever a client is connecting to the OPC server. If the driver returns something other than S_OK, the client connection will be rejected.

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| E_ACCESSDENIED | The access for this client is denied. |
| E_UNEXPECTED | An unexpected error. |
| E_OUTOFMEMORY | Enough memory. |
| E_FAIL | The operation failed. |

**Table 54: [hr] Possible Values for OnClienConnect**

Default behavior returns S_OK.

**VB Wrapper**

Sub OnClientConnect (ByRef error As Long)

**C# .NET**

```
delegate void Del_IOONCLIENTCONNECTPROC(out IntPtr hr)
```

**Code example**

```
// static void OnClientConnect(out IntPtr hr)

OPCNetServerSDKManager.Del_IOONCLIENTCONNECTPROC del_IOONCLIENTCONNECTPROC =
new OPCNetServerSDKManager.Del_IOONCLIENTCONNECTPROC(OnClientConnect);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOONCLIENTCONNECTPROC(ByRef hr As IntPtr)
```

**Code example**

```
'Public Shared Sub OnClientConnect (ByRef hr As IntPtr)

Dim del_IOONCLIENTCONNECTPROC As New
OPCNetServerSDKManager.Del_IOONCLIENTCONNECTPROC (OnClientConnect)
```

**C++ .NET**

```
delegate void Del_IOONCLIENTCONNECTPROC (IntPtr %hr);
```

**Code example**

```
//public static void OnClientConnect (IntPtr %hr)

OPCNetServerSDKManager.Del_IOONCLIENTCONNECTPROC del_IOONCLIENTCONNECTPROC =
gcnew OPCNetServerSDKManager.Del_IOONCLIENTCONNECTPROC(OnClientConnect);
```

## 4.4. OnClientDisconnect

VOID OnClientDisconnect ([out] HRESULT* hr)

**Description**

This function will be called whenever a client has disconnected. The server can then perform memory release operations.

(Default behavior is returning S_OK)

**VB Wrapper**

Sub OnClientDisconnect (ByRef error As Long)

**C# .NET**

```
delegate void Del_IOONCLIENTDISCONNECTPROC(out IntPtr hr)
```

**Code example**

```
// static void OnClientDisonnect(out IntPtr hr);

OPCNetServerSDKManager.Del_IOONCLIENTDISCONNECTPROC
del_IOONCLIENTDISCONNECTPROC = new
OPCNetServerSDKManager.Del_IOONCLIENTDISCONNECTPROC(OnClientDisconnect);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOONCLIENTCONNECTPROC(ByRef hr As IntPtr)
```

**Code example**

```
'Public Shared Sub OnClientDisonnect (ByRef hr As IntPtr)
```

```
Dim del_IOONCLIENTDISCONNECTPROC As New OPCNetServerSDKManager.Del_IOONCLIENTDISCONNECTPROC (AddressOf OnClientDisconnect)
```

**C++ .NET**

```
delegate void Del_IOONCLIENTDISCONNECTPROC (IntPtr %hr);
```

**Code example**

```
//public static void OnClientDisconnect (IntPtr %hr)

OPCNetServerSDKManager.Del_IOONCLIENTDISCONNECTPROC del_IOONCLIENTDISCONNECTPROC = gcnew OPCNetServerSDKManager.Del_IOONCLIENTDISCONNECTPROC (OnClientDisconnect);
```

## 4.5. SvrCanUnloadNow

VOID SvrCanUnloadNow ( [out] HRESULT* hr )

**Description**

This method allows the DLL to notify the driver that it can unload itself from memory since no client is connected.

**Out parameters**

hr: returned error code (by default S_OK).

**VB Wrapper**

```
Sub SvrCanUnloadNow(ByRef error As Long)
```

**C# .NET**

```
delegate void Del_IOSVRCANUNLOADNOWPROC(out IntPtr hr)
```

**Code example**

```
OPCNetServerSDKManager.Del_IOSVRCANUNLOADNOWPROC del_IOSVRCANUNLOADNOWPROC = new OPCNetServerSDKManager.Del_IOSVRCANUNLOADNOWPROC(SvrCanUnloadNow);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOSVRCANUNLOADNOWPROC(ByRef hr As IntPtr)
```

**Code example**

```
Private Del_IOSVRCANUNLOADNOWPROC1 As New OPCNetServerSDKManager.Del_IOSVRCANUNLOADNOWPROC(AddressOf SvrCanUnloadNow)
```

**C++ .NET**

```
delegate void Del_IOSVRCANUNLOADNOWPROC(IntPtr %hr)
```

**Code example**

```
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOSVRCANUNLOADNOWPROC^
del_IOSVRCANUNLOADNOWPROC = gcnew
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOSVRCANUNLOADNOWPROC(SvrCanUnloadN
ow);
```

## 4.6. ValidateItem

```
VOID ValidateItem (     [in] BSTR          AccessPath
                      , [in] BSTR          ItemID
                      , [in] VARTYPE       vtRequestedDataType
                      , [out] DWORD*       pdwAccessRights
                      , [out] VARTYPE*     pvtType
                      , [out] HRESULT*     hr)
```

**Description**

This method allows the DLL to determine if the given item is valid (it should exist in the driver namespace).

| Parameter Name | Description |
|---|---|
| AccessPath | Identifies the access path of the tag to validate. A NUL string ("") may be passed, if this parameter needs to be ignored. Note that Access Paths are optional in the OPC Data Access specification. |
| ItemID | Identifies the tag name. |
| vtRequestedDataType | Identifies the requested data type of the tag. |
| pdwAccessRights | Item access rights. |
| pvType | The canonical data type. |
| Hr | Returned error code (see possible values in the next table). |

**Table 55: ValidateItem Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. Validation information is stored in (pdwAccessRights, pvType) out parameters. |
| OPC_E_INVALIDITEMID | The ItemID is not syntactically valid. |
| 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. |
| OPC_E_BADTYPE | The requested data type cannot be returned for this item. |
| E_FAIL | The function failed. The function was unsuccessful for this item. |

**Table 56: [hr] Possible Values for ValidateItem**

**VB Wrapper**

```
Sub ValidateItem (ByVal AccessPath As String _
                , ByVal ItemID As String _
                , ByVal RequestedDataType As Long _
                , ByRef AccessRight As Long _
                , ByRef DataType As Integer _
                , ByRef error As Long)
```

**C# .NET**

```
delegate void Del_IOVALIDATEITEMPROC(string strAccessPath,
                                     string strStrTagName,
                                     UInt16 ushoRequestedDataType,
                                     out UInt32 uintAccessRights,
                                     out UInt16 ushoType,
                                     out IntPtr hr)
```

**Code example**

```
/*public static void ValidateItem(string strAccessPath,
                                 string strStrTagName,
                                 UInt16 ushoRequestedDataType,
                                 out UInt32 uintAccessRights,
                                 out UInt16 ushoType,
                                 out IntPtr hr)*/
OPCNetServerSDKManager.Del_IOVALIDATEITEMPROC del_IOVALIDATEITEMPROC = new
OPCNetServerSDKManager.Del_IOVALIDATEITEMPROC(ValidateItem);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOVALIDATEITEMPROC(
                 ByVal bstrAccessPath As String,
                ByVal strStrTagName As String,
                ByVal ushoRequestedDataType As UInt16,
                ByRef uintAccessRights As UInt32,
                ByRef ushoType As UInt16,
                ByRef hr As IntPtr)
```

**Code example**

```
'public Sub ValidateItem(ByVal bstrAccessPath As string,
```

```
                    'ByVal StrTagName As string,
                    'ByVal vtRequestedDataType As UInt16,
                    'ByRef pdwAccessRights As UInt32,
                    'ByRef pvtType As UInt16,
                    'ByRef hr As IntPtr)
Dim del_IOVALIDATEITEMPROC As New
OPCNetServerSDKManager.Del_IOVALIDATEITEMPROC(AddressOf ValidateItem)
```

**C++ .NET**

```
delegate void Del_IOVALIDATEITEMPROC(
            string^ bstrAccessPath,
            string^ StrTagName,
            UInt16 vtRequestedDataType,
            UInt32 pdwAccessRights,
            UInt16 pvtType,
            IntPtr %hr)
```

**Code example**

```
/*static void ValidateItem(String^ bstrAccessPath, String^ StrTagName, UInt16 vtRequestedDataType,
UInt32 %pdwAccessRights, UInt16 %pvtType, IntPtr %hr)*/
OPCNetServerSDKManager::Del_IOVALIDATEITEMPROC^ del_IOVALIDATEITEMPROC = gcnew
OPCNetServerSDKManager::Del_IOVALIDATEITEMPROC(ValidateItem);
```

## 4.7. ValidateAndInsert

VOID ValidateAndInsert ( [in] BSTR StrTagName
, [out] HRESULT * hr)

**Description**

Check if a given itemID is a valid tag name. First, the developer searches for this tag locally in the server. If it does not exist, he should ask the equipment. If this item figure is in the real system, the developer should call the io_createtag function to add it to the server address space.

| Parameter Name | Description |
|---|---|
| StrTagName | The tag name to validate. |
| Hr | Returned error code (see possible values in the next table). |

**Table 57: ValidateAndInsert Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| OPC_E_UNKNOWNITEMID | The given NameTag does not correspond to any |

| | |
|---|---|
| | monitored data point. |
| E_FAIL | The function failed. The function was unsuccessful for this item. |
| S_xxx (success code)<br>E_xxx (error code) | Vendor specific information. |

**Table 58: [hr] Possible Values for ValidateAndInsert**

**VB Wrapper**

```
Sub ValidateAndAdd(ByVal StrTagName As String, _
                   ByRef error As Long)
```

**C# .NET**

```
delegate void Del_IOVALIDATEANDADDPROC(string strTagName,
                out IntPtr hr)
```

**Code example**

```
//static void ValidateAndInsert (String strTagName, out IntPtr hr);

OPCNetServerSDKManager.Del_IOVALIDATEANDADDPROC del_IOWRITEITEMPROC = new
OPCNetServerSDKManager.Del_IOVALIDATEANDADDPROC(ValidateAndInsert);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOUPDATEITEMPROC(ByVal StrTagName As String,
                                  ByRef hr As IntPtr)
```

**Code example**

```
'Shared Sub ValidateAndInsert(ByVal strTagName As String,
                'ByRef hr As IntPtr)

Private del_IOVALIDATEANDADDPROC As New
OPCNetServerSDKManager.Del_IOVALIDATEANDADDPROC(AddressOf ValidateAndInsert)
```

**C++ .NET**

```
delegate void Del_IOVALIDATEANDADDPROC (String^ strTagName,  IntPtr %hr )
```

**Code example**

```
//void ValidateAndInsert(String^ strTagName,  IntPtr %hr);

OPCNetServerSDK::OPCNetServerSDKManager::Del_IOVALIDATEANDADDPROC^
del_IOVALIDATEANDADDPROC = gcnew
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOVALIDATEANDADDPROC (ValidateAndInsert);
```

## 4.8. QueryAvailableProperties

```
VOID    QueryAvailableProperties (    [in] LPCTSTR     stItemID
                                    , [in] DWORD       *uintCount
                                    , [out] DWORD      **phPropertyIDs
                                    , [out] LPWSTR     **pawszDescriptions
                                    , [out] VARTYPE **pavtDataTypes
                                    , [out] HRESULT  * hr)
```

**Description**

This function allows the DLL to ask for the IDs, descriptions and data types of the properties supported by the specified ItemID (see the section **Item properties**).

| Parameter Name | Description |
|---|---|
| szItemID | Identifies the tag for which the DLL wants to know the available properties. |
| uintCount | The number of returned properties. |
| phPropertyIDs | Property identifier of the returned properties. |
| pawszDescriptions | Description of the returned properties. |
| pavtDataTypes | Data types of the returned properties. |
| Hr | Returned error code (see possible values in the next table). |

**Table 59: QueryAvailableProperties Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| OPC_E_INVALIDITEMID | The ItemID is not syntactically valid. |
| OPC_E_UNKNOWNITEMID | The ItemID is not in the server address space. |
| OPC_E_OUTOFMEMORY | Not enough memory. |
| E_FAIL | The function failed. |

**Table 60: [hr] Possible Values for QueryAvailableProperties**

**Comments**

For developers that are using C++ environment, phPropertyIDs, pawszDescriptions, and pavtDataTypes arrays should be allocated by the driver using the **AllocMemory** function (see the section Helper functions ). Then they are freed by the DLL.

**VB Wrapper**

For VB environment, we added the "VB_NumberOfItemProperties" callback. This callback retrieves the number of properties of a given tag and logically precedes any call to VB_QueryAvailablePropertiesCallBack.

**C++ Prototype:**

```
VB_NumberOfItemProperties (BSTR ItemID
 , DWORD* Count)
```

**VB:**

```
Sub NumberOfItemProperties(ByVal ItemID As String _
, ByRef Count As Long)
```

**C++ Prototype:**

```
VB_QueryAvailableProperties  (BSTR ItemID
                              , DWORD Count
                              , SAFEARRAY ** PropertyID
                              , SAFEARRAY ** Description
                              , SAFEARRAY ** Datatype
                              , HRESULT * error)
```

**VB:**

```
Sub QueryAvailableProperties(ByVal ItemID As String _
                              , ByVal Count As Long _
                              , PropertyID() As Long _
                              , Description() As String _
                              , Datatype() As Integer _
                              , ByRef error As Long)
```

C# .NET

```
delegate void Del_IOQUERYAVAILABLEPROPERTIESPROCBSTR(
                string strTagName,
                out UInt32 uintCount,
                out UInt32[] uintPropertyIDs,
                out IntPtr[] hDescriptions,
                out UInt16[] ushoDataTypes,
                out IntPtr hr)
```

**Code example**

```
/*static void QueryAvailableProperties(String strTagName,
                        out UInt32 uintCount,
                        out UInt32[] uintPropertyIDs,
                        out IntPtr[] hDescriptions,
                        out UInt16[] ushoDataTypes,
                        out IntPtr hr)*/

OPCNetServerSDKManager.Del_IOQUERYAVAILABLEPROPERTIESPROCBSTR
del_IOQUERYAVAILABLEPROPERTIESPROC = new
OPCNetServerSDKManager.Del_IOQUERYAVAILABLEPROPERTIESPROCBSTR(WriteVQT);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOQUERYAVAILABLEPROPERTIESPROCBSTR(
                        ByVal strTagName As String,
                        ByRef uintCount As UInt32,
                        ByRef uintPropertyIDs As UInt32(),
                        ByRef hDescriptions As IntPtr(),
                        ByRef ushoDataTypes As UInt16(),
                        ByRef hr As IntPtr)
```

**Code example**

```
'Shared Sub QueryAvailableProperties(ByVal strTagName As String,
                        'ByRef uintCount As UInt32,
                        'ByRef uintPropertyIDs As UInt32(),
                        'ByRef hDescriptions As IntPtr(),
                        'ByRef ushoDataTypes As UInt16(),
                        'ByRef hr As IntPtr)

Private del_IOQUERYAVAILABLEPROPERTIESPROCBSTR As New
OPCNetServerSDKManager.Del_IOQUERYAVAILABLEPROPERTIESPROCBSTR(AddressOf
QueryAvailableProperties)
```

**C++ .NET**

```
delegate void Del_IOQUERYAVAILABLEPROPERTIESPROCBSTR(
                                        String^ strTagName,
                                        UInt32 %uintCount,
                                        array < UInt32 ^> %uintPropertyIDs,
                                        array < IntPtr ^> %hDescriptions,
                                        array < UInt16 ^> %ushoDataTypes,
                                        IntPtr %hr)
```

**Code example**

```
/*public void QueryAvailableProperties (String^ strTagName,
                        UInt32 %uintCount,
                        array < UInt32 ^> %uintPropertyIDs,
                        array < IntPtr ^> %hDescriptions,
                        array < UInt16 ^> %ushoDataTypes,
                        IntPtr %hr)*/
```

```
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOQUERYAVAILABLEPROPERTIESPROCBSTR
^ del_IOWRITEITEMPROC = gcnew
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOQUERYAVAILABLEPROPERTIESPROCBSTR
(QueryAvailableProperties);
```

## 4.9. GetItemProperties

```
VOID    GetItemProperties ( [in] BSTR         ItemID
                          , [in] DWORD        uintCount
                          , [in] DWORD        hPropertyIDs
                          , [out] VARIANT*    objData
                          , [out] HRESULT*    pErrors
                          , [out] HRESULT*    hr)
```

**Description**

This function allows the DLL to ask for the current data values for the ID codes that have been passed.

| Parameter Name | Description |
|---|---|
| ItemID | Identifies the tag for which the DLL wants to know the current values of some of its properties. |
| uintCount | The number of properties to read. |
| hPropertyIDs | The list of properties IDs for which we need to read current values. |
| objData | Array of VARIANTs containing the current values of the requested properties. |
| pErrors | Returned errors for each item property. |
| Hr | Returned error code (see possible values in the next table). |

**Table 61: GetItemProperties Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| S_FALSE | The function succeeded partially. Check error codes in pErrors. |
| OPC_E_INVALIDITEMID | The ItemID is not syntactically valid. |
| OPC_E_UNKNOWNITEMID | The ItemID is not in the server address space. |

| | |
|---|---|
| E_FAIL | The function failed. |

**Table 62: [hr] Possible Values for GetItemProperties**

**Comments**

objData and pErrors arrays are allocated by the DLL and need just to be filled by the driver.

**VB Wrapper**

**C++ prototype:**

```
VB_GetItemProperties (BSTR ItemID
                      , DWORD Count
                      , SAFEARRAY ** PropertyIDs
                      , SAFEARRAY ** data
                      , SAFEARRAY ** Errors
                      , HRESULT * error)
```

**VB:**

```
Sub GetItemProperties (ByVal ItemID As String _
                       , ByVal Count As Long _
                       , PropertyIDs() As Long _
                       , data() As Variant _
                       , Errors() As Long _
                       , ByRef error As Long)
```

**C# .NET**

```
delegate void Del_IOGETITEMPROPERTIESPROC(string strTagName,
                                          UInt32 uintCount,
                                          IntPtr hPropertyIDs,
                                          out object[] objData,
                                          out IntPtr[] hErrors,
                                          out IntPtr hr)
```

**Code example**

```
/*static void GetItemProperties(String strTagName,
                        UInt32 uintCount,
                        IntPtr hPropertyIDs ,
                        out object[] objData,
                        out IntPtr[] hErrors,
                        out IntPtr hr)*/

OPCNetServerSDKManager.Del_IOGETITEMPROPERTIESPROC
del_IOQUERYAVAILABLEPROPERTIESPROC = new
OPCNetServerSDKManager.Del_IOGETITEMPROPERTIESPROC(GetItemProperties);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOGETITEMPROPERTIESPROC(ByVal strTagName As String,
                                         ByVal uintCount As UInt32,
                                         ByVal hPropertyIDs  As IntPtr,
                                         ByRef objData As Object(),
                                         ByRef hErrors As IntPtr(),
                                         ByRef hr As IntPtr)
```

**Code example**

```
'Shared Sub GetItemProperties (ByVal strTagName As String,
                    'ByVal uintCount  As UInt32,
                    'ByVal phPropertyIDs  As IntPtr,
                    'ByRef objData As Object(),
                    'ByRef hErrors As IntPtr(),
                    'ByRef hr As IntPtr)

Private del_IOGETITEMPROPERTIESPROC As New
OPCNetServerSDKManager.Del_IOGETITEMPROPERTIESPROC (AddressOf GetItemProperties)
```

**C++ .NET**

```
delegate void Del_IOGETITEMPROPERTIESPROC(String^ strTagName,
                                          UInt32 uintCount ,
                                          IntPtr hPropertyIDs ,
                                          array < Object ^> %objData,
                                          array < IntPtr ^> %hErrors,
                                          IntPtr %hr)
```

**Code example**

```
/*public void QueryAvailableProperties (String^ StrTagName,
                        UInt32 uintCount ,
                        IntPtr hPropertyIDs ,
                        array < Object ^> %objData,
                        array < IntPtr ^> %hErrors,
                        IntPtr %hr) */

OPCNetServerSDK::OPCNetServerSDKManager::Del_IOGETITEMPROPERTIESPROC ^
del_IOWRITEITEMPROC = gcnew
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOGETITEMPROPERTIESPROC
(GetItemProperties);
```

## 4.10. LookupItemIDs

```
VOID   LookupItemIDs (  [in] BSTR          ItemID
                      , [in] DWORD         uintCount
                      , [out] DWORD*       hPropertyIDs
                      , [out] HRESULT*     hr )
```

**Description**

This function allows the DLL to ask for the list of property IDs (if available) for the Item identifier that has been passed. The DLL will generate the new item IDs for returned property IDs.

Example

**DA/ReadWrite/Float: 2**

| Parameter Name | Description |
|---|---|
| ItemID | Identifies the tag for which the DLL wants to look up the list of properties. |
| uintCount | The number of properties to look up. |
| hPropertyIDs | IDs available for the requested Item ID. |
| hr | Returned error code (see possible values in the next table). |

**Table 63: LookUpItemIDs Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| OPC_E_INVALIDITEMID | The ItemID is not syntactically valid. |
| OPC_E_UNKNOWNITEMID | The ItemID is not in the server address space. |
| E_FAIL | The function failed. |

**Table 64: [hr] Possible Values for LookUpItemIDs**

**Comments**

The OPC_INVALID_PID error code is handled in the DLL.

For developers using C++ environment, phPropertyIDs array should be allocated by the driver using the **AllocMemory** function (see section Helper functions ). Then they are freed by the DLL.

**VB Wrapper**

**C++ prototype:**

```
VB_LookupItemProperties ( BSTR ItemID
                        , DWORD Count
                        , SAFEARRAY ** PropertyIDs
                        , HRESULT * error)
```

**VB:**

```
Sub LookupItemID (ByVal ItemID As String _
                , ByVal Count As Long _
                , PropertyIDs() As Long _
                , ByRef error As Long)
```

**C# .Net**

```
delegate void Del_IOLOOKUPITEMIDSPROC(String strItemID,
                                      UInt32 uintCount,
                                      out UInt32[] hPropertyIDs ,
                                      out IntPtr hr)
```

**Code example**

```
/*public static void LookupItemIDs(String strTagName,
                                  UInt32 uintCount,
                                  out UInt32[] hPropertyIDs ,
                                  out IntPtr hr)*/

OPCNetServerSDKManager.Del_IOLOOKUPITEMIDSPROC del_IOLOOKUPITEMIDSPROC = new
OPCNetServerSDKManager.Del_IOLOOKUPITEMIDSPROC(LookupItemIDs);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOLOOKUPITEMIDSPROC(ByVal strItemID  As [String],
                                     ByVal uintCount As UInt32,
                                     ByRef hPropertyIDs  As UInt32(),
                                     ByRef hr As IntPtr)
```

**Code example**

```
'Public Shared Sub LookupItemIDs(ByVal strItemID  As String,
                                 ByVal uintCount As UInt32,
                                 ByRef hPropertyIDs  As UInt32(),
                                 ByRef hr As IntPtr)

Dim del_IOLOOKUPITEMIDSPROC As New
OPCNetServerSDKManager.Del_IOLOOKUPITEMIDSPROC(LookupItemIDs)
```

**C++ .NET**

```
delegate void Del_IOLOOKUPITEMIDSPROC(String^ strItemID ,
                                      UInt32 uintCount,
                                      array<UInt32[]> %hPropertyIDs ,
                                      IntPtr %hr)
```

**Code example**

```
/*public static void LookupItemIDs(String^ strTagName,
                                  UInt32 uintCount,
                                  array<UInt32[]> %hPropertyIDs ,
                                  IntPtr %hr)*/
```

```
OPCNetServerSDKManager.Del_IOLOOKUPITEMIDSPROC del_IOLOOKUPITEMIDSPROC = gcnew
OPCNetServerSDKManager.Del_IOLOOKUPITEMIDSPROC(LookupItemIDs);
```

## 4.11. UpdateItem

HRESULT UpdateItem ( ([in] BSTR StrTagName
, [out] VARIANT* Value
, [out] WORD* UshoQuality
, [out] FILETIME* Timestamp
, [out] HRESULT* hr)

**Description**

This function allows the DLL to update the value, ushoQuality, and timestamp information of the specified item. It will be activated once an OPC client connects to the OPC Server, add a group then add items. Only the added tags will be updated.

| Parameter Name | Description |
|---|---|
| StrTagName | The tag name to update its attributes. |
| Value | New value. |
| UshoQuality | New ushoQuality. |
| Timestamp | New timestamp. |
| hr | Returned error code (see possible values in the next table). |

**Table 65: UpdateItem Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| OPC_E_INVALIDITEMID | The ItemID is not syntactically valid. |
| OPC_E_UNKNOWNITEMID | The ItemID is not in the server address space. |
| E_FAIL | The function failed. The function was unsuccessful for this item. |
| S_xxx (success code)<br>E_xxx (error code) | Vendor specific information. |

**Table 66: [hr] Possible Values for UpdateItem**

**C# .NET**

```
delegate void Del_IOUPDATEITEMPROC(string strTagName,
                out object objValue,
                out System.Runtime.InteropServices.ComTypes.FILETIME tS,
                out UInt16 ushoQuality,
                out IntPtr hr)
```

**Code example**

```
/*static void UpdateItem(String strTagName,
          out object objValue,
          out System.Runtime.InteropServices.ComTypes.FILETIME TS,
          out UInt16 ushoQuality,
          out IntPtr hr)*/

OPCNetServerSDKManager.Del_IOUPDATEITEMPROC del_IOWRITEITEMPROC = new
OPCNetServerSDKManager.Del_IOUPDATEITEMPROC(WriteVQT);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOUPDATEITEMPROC(ByVal strTagName As String,
            ByRef objValue As Object,
            ByRef tS As System.Runtime.InteropServices.ComTypes.FILETIME,
            ByRef ushoUshoQuality As UInt16,
            ByRef hr As IntPtr)
```

**Code example**

```
'Shared Sub UpdateItem(ByVal strTagName As String,
        'ByRef objValue As Object,
       'ByRef tS As System.Runtime.InteropServices.ComTypes.FILETIME,
        'ByRef ushoQuality As UInt16,
        'ByRef hr As IntPtr)
Private del_IOUPDATEITEMPROC As New
OPCNetServerSDKManager.Del_IOUPDATEITEMPROC(AddressOf UpdateItem)
```

**C++ .NET**

```
delegate void Del_IOUPDATEITEMPROC (String^ strTagName,
                Object^ %objValue,
                System::Runtime::InteropServices::ComTypes::FILETIME %tS,
                UInt16 %ushoUshoQuality,
                IntPtr %hr )
```

**Code example**

```
/*UpdateItem(String^ strTagName,
          Object^ %objValue,
          System::Runtime::InteropServices::ComTypes::FILETIME %TS,
          UInt16 %ushoUshoQuality,
          IntPtr %hr)
```

```
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOUPDATEITEMPROC^
del_IOWRITEITEMPROC = gcnew
OPCNetServerSDK::OPCNetServerSDKManager::Del_IOUPDATEITEMPROC(WriteVQT);
```

## 4.12. ReadItem

HRESULT ReadItem ( [in] BSTR strTagName
, [out] VARIANT* Value
, [out] FILETIME * Timestamp
, [out] WORD* UshoQuality
, [out] HRESULT* hr)

**Description**

This function allows the DLL to update the value, ushoQuality and timestamp information of the specified item.

It will be activated once an OPC client connects to the OPC Server, adds a group (read mode is from device) then adds the OPC items. Only the added items will be updated.

As this toolkit supports the OPC Data Access specifications version 3.0, the given item is not necessarily an added item, or a cached one, but it is simply a tag in the server address space. The client can send a read request both at the server or group level (see section Supported Interfaces).

| Parameter Name | Description |
|---|---|
| StrTagName | The tag name to update its attributes. |
| Value | The current value. |
| UshoQuality | The current ushoQuality. |
| Timestamp | The current timestamp. |
| hr | Returned error code (see possible values in the next table). |

**Table 67: ReadItem Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| OPC_E_INVALIDITEMID | The ItemID is not syntactically valid. |
| OPC_E_UNKNOWNITEMID | The ItemID is not in the server address space. |
| E_FAIL | The function failed. The function was unsuccessful for |

| | |
|---|---|
| | this item. |
| S_xxx (success code)<br>E_xxx (error code) | Vendor specific information can be provided if this item ushoQuality is other than GOOD. |

**Table 68: [hr] Possible Values for ReadItem**

**Comments**

Some error codes can be treated internally. The driver does not have to check if the given item to be read is readable. This is treated within the DLL, as it already has the access rights information for all items available in server address space.

**VB Wrapper**

```
Sub ReadItem (ByVal StrTagName As String _
                , ByRef ptValue As Variant _
                , ByRef Time As FILETIME _
                , ByRef UshoQuality As Integer _
                , ByRef error As Long)
```

**C# .NET**

```
public delegate void Del_IOREADITEMPROC(string strStrTagName, out object objValue, out System.Runtime.InteropServices.ComTypes.FILETIME fTS, out UInt16 ushoUshoQuality, out IntPtr hr)
```

**Code example**

```
/*static void ReadItem(String strStrTagName,
            out object objValue,
            out System::Runtime::InteropServices::ComTypes::FILETIME fTS,
            out UInt16 ushoUshoQuality,
            out IntPtr hr )*/
OPCNetServerSDK.OPCNetServerSDKManager.Del_IOREADITEMPROC del_IOREADITEMPROC =
new OPCNetServerSDK.OPCNetServerSDKManager.Del_IOREADITEMPROC(ReadItem);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOREADITEMPROC(ByVal strStrTagName As String, ByRef objValue As Object, ByRef fTS As System.Runtime.InteropServices.ComTypes.FILETIME, ByRef ushoUshoQuality As UInt16, ByRef hr As IntPtr)
```

**Code example**

```
'public Shared Sub ReadItem(ByVal strStrTagName As string,
    'ByRef objValue As object,
    'ByRef fTS As  System.Runtime.InteropServices.ComTypes.FILETIME,
    'ByRef ushoUshoQuality As UInt16,
    'ByRef hr As IntPtr)
Dim del_IOREADITEMPROC As New
OPCNetServerSDK.OPCNetServerSDKManager.Del_IOREADITEMPROC(AddressOf ReadItem)
```

**C++ .NET**

```
delegate void Del_IOREADITEMPROC(String^ strStrTagName,
                Object^ %objValue,
                System::Runtime::InteropServices::ComTypes::FILETIME %TS,
                UInt16 %ushoUshoQuality,
                IntPtr %hr )
```

**Code example**

```
/*static void ReadItem(String^ strStrTagName,
            Object^ %objValue,
            System::Runtime::InteropServices::ComTypes::FILETIME %TS,
            UInt16 %ushoUshoQuality,
            IntPtr %hr )*/

OPCNetServerSDK::OPCNetServerSDKManager::Del_IOREADITEMPROC^ Del_IOREADITEMPROC1 =
gcnew OPCNetServerSDK::OPCNetServerSDKManager::Del_IOREADITEMPROC(ReadItem);
```

## 4.13. WriteItem

```
HRESULT WriteItem (  [in] BSTR strstrTagName
                   , [in] VARIANT Value
                   , [out] FILETIME * Timestamp
                   , [out] WORD* UshoQuality
                   , [out] HRESULT* hr)
```

**Description**

This function allows the DLL to write a value for the specified cached tag to the physical device. Consequently, ushoQuality and timestamp information should also be updated.

| Parameter Name | Description |
|---|---|
| StrTagName | The tag name to update its attributes. |
| Value | The value to write in the device. |
| UshoQuality | The ushoQuality of the written item. |
| Timestamp | The timestamp of the written item. |
| hr | Returned error code (see possible values in the next table). |

**Table 69: WriteItem Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. |

| | |
|---|---|
| OPC_E_UNKNOWNITEMID | The ItemID is not in the server address space. |
| OPC_S_ClAMP | The value was accepted but was clamped. |
| OPC_E_RANGE | The value was out of range. |
| E_FAIL | The function failed. The function was unsuccessful for this item. |
| S_xxx (success code)<br>E_xxx (error code) | Vendor specific information. |

**Table 70: [hr] Possible Values for WriteItem**

**Comments**

Some error codes can be treated internally. For example, if the passed data type cannot be accepted for this item, the DLL will automatically generate an OPC_E_BADTYPE error code. The driver does not need to check if the given item can be written. This is handled by the DLL, as it already has the access rights information for all items available in server address space.

**VB Wrapper**

```
Sub WriteItem (ByVal strStrTagName As String _
                , ByVal ptValue As Variant _
                , ByRef Time As FILETIME _
                , ByRef intUshoQuality As Integer _
                , ByRef lngError As Long)
```

**C# .NET**

```
delegate void Del_IOWRITEITEMPROC(string strStrTagName,
                object objValue,
                out System.Runtime.InteropServices.ComTypes.FILETIME TS,
                out UInt16 ushoUshoQuality,
                out IntPtr hr)
```

**Code example**

```
/*static void WriteItem(String strStrTagName,
        object objValue,
        out System::Runtime::InteropServices::ComTypes::FILETIME fTS,
        out UInt16 ushoUshoQuality,
        out IntPtr hr )*/

OPCNetServerSDKManager.Del_IOWRITEITEMPROC del_IOWRITEITEMPROC = new
OPCNetServerSDKManager.Del_IOWRITEITEMPROC(WriteItem);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOWRITEITEMPROC(ByVal strStrTagName As String,
                                 ByVal objValue As Object,
                                 ByRef tS As System.Runtime.InteropServices.ComTypes.FILETIME,
                                 ByRef UshoQuality As UInt16, ByRef hr As IntPtr)
```

**Code example**

```
'Public Shared Sub WriteItem(ByVal strStrTagName As String,
            'ByVal objValue As Object,
            'ByRef TS As System.Runtime.InteropServices.ComTypes.FILETIME,
            'ByRef UshoQuality As UInt16,
            'ByRef hr As IntPtr)

Private del_IOWRITEITEMPROC As New
OPCNetServerSDKManager.Del_IOWRITEITEMPROC(AddressOf WriteItem)
```

**C++ .NET**

```
delegate void Del_IOWRITEITEMPROC(String^ strStrTagName,
                                  Object^ %objValue,
                                  System::Runtime::InteropServices::ComTypes::FILETIME %tS,
                                  UInt16 %ushoUshoQuality,
                                  IntPtr %hr )
```

**Code example**

```
/*static void WriteItem(String^ strStrTagName,
                Object^ %objValue,
                System::Runtime::InteropServices::ComTypes::FILETIME %TS,
                UInt16 %ushoUshoQuality,
                IntPtr %hr )*/

OPCNetServerSDK::OPCNetServerSDKManager::Del_IOWRITEITEMPROC^ del_IOWRITEITEMPROC
= gcnew OPCNetServerSDK::OPCNetServerSDKManager::Del_IOWRITEITEMPROC(WriteItem);
```

## 4.14. WriteVQT

```
HRESULT WriteVQT (      [in] BSTR          strTagName
                      , [in] VARIANT&    cValue
                      , [in] BOOL        bTimeSpecified
                      , [out] FILETIME*  pftTimestamp
                      , [in] BOOL        bUshoQualitySpecified
                      , [out] WORD*      pwUshoQuality
                      , [out] HRESULT*   hr)
```

**Description**

An OPC client can send a request to write the value, ushoQuality and timestamp for the item specified to the physical device at the server or group level (see section Supported

Interfaces). This is functionally similar to the WriteItem method except that ushoQuality and timestamp may also be written. A client attempts to write VQ, VT or VQT.

| Parameter Name | Description |
|---|---|
| StrTagName | The tag name to update its attributes. |
| cValue | Identifies the value to write in the device. |
| bTimeSpecified | TRUE: Pointer to timestamp value is not null ➔ writing timestamp to device for the specified tag.<br>FALSE: Ignore the pftTimestamp parameter. |
| pftTimestamp | Identifies the timestamp to write in the device. It can be a NULL pointer. |
| pUshoQualitySpecified | TRUE: Pointer to ushoQuality value is not null ➔ writing ushoQuality to device for the specified tag.<br>FALSE: Ignore the pwUshoQuality parameter. |
| pwUshoQuality | Identifies the ushoQuality to write in the device. It can be a NULL pointer. |
| hr | Returned error code (see possible values in the next table). |

**Table 71: WriteVQT Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| OPC_E_UNKNOWNITEMID | The ItemID is not in the server address space. |
| OPC_S_ClAMP | The value was accepted but was clamped. |
| OPC_E_RANGE | The value was out of range. |
| E_FAIL | The function failed. The function was unsuccessful for this item. |
| S_xxx (success code)<br>E_xxx (error code) | Vendor specific information. |

**Table 72: [hr] Possible Values for WriteVQT**

**Comments**

UshoQuality and timestamp can be NULL pointers. The driver attempts to write only the value, and update ushoQuality and timestamp as WriteItem method. Similarly, some

error codes can be handled internally. For example, if the passed data type cannot be accepted for this item, the DLL will automatically generate an OPC_E_BADTYPE error code. The driver does not need to check if the given item can be written. This is treated by the DLL, as it already has the access rights information for all items available in server address space.

**VB Wrapper**

```
Sub WriteVQT (ByVal StrTagName As String _
                , ByRef Value As Variant _
                , ByVal TimeFlag As Integer _
                , ByRef Time As FILETIME _
                , ByVal QualFlag As Integer _
                , ByRef UshoQuality As Integer _
                , ByRef error As Long)
```

**C# .NET**

```
delegate void Del_IOWRITEVQTPROC(string strStrTagName,
                ref object objValue,
                bool blnTS,
                out System.Runtime.InteropServices.ComTypes.FILETIME tS,
                bool blnQ,
                out UInt16 ushoUshoQuality,
                out IntPtr hr)
```

**Code example**

```
/*static void WriteVQT(String strStrTagName,
            Object objValue,
            ool blnTS,
            System.Runtime.InteropServices.ComTypes.FILETIME tS,
            out bool blnQ,
            out  UInt16 ushoUshoQuality,
            out IntPtr hr);*/
OPCNetServerSDKManager.Del_IOWRITEVQTPROC del_IOWRITEITEMPROC = new
OPCNetServerSDKManager.Del_IOWRITEVQTPROC(WriteVQT);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOWRITEVQTPROC(ByVal strStrTagName As String, ByVal objValue As
Object, ByVal blnT As bool, ByRef tS As System.Runtime.InteropServices.ComTypes.FILETIME,
ByVal blnQ As bool, ByRef ushoUshoQuality As UInt16, ByRef hr As IntPtr)
```

**Code example**

```
'Shared Sub WriteVQT(ByVal strStrTagName As [String],
      'ByRef objValue As [Object],
      'ByVal blnTS As Boolean,
      'ByVal tS As System.Runtime.InteropServices.ComTypes.FILETIME,
      'ByVal blnQ As Boolean,
```

```
        'ByRef ushoUshoQuality As UInt16,
        'ByRef hr As IntPtr)

Private del_IOWRITEVQTPROC As New
OPCNetServerSDKManager.Del_IOWRITEVQTPROC(AddressOf WriteVQT)
```

**C++ .NET**

```
delegate void Del_IOWRITEVQTPROC (String^ strTagName,
Object^ %objValue,
Boolean blnTS,
System::Runtime::InteropServices::ComTypes::FILETIME tS,
Boolean blnQ,
UInt16 %ushoUshoQuality,
IntPtr %hr )
```

**Code example**

```
/*static void WriteVQT(String^ strTagName,
                Object^ %objValue,
                Boolean blnTS,
                System::Runtime::InteropServices::ComTypes::FILETIME tS,
                Boolean blnQ,
                UInt16 %ushoUshoQuality,
                IntPtr %hr);*/

OPCNetServerSDK::OPCNetServerSDKManager::Del_IOWRITEVQTPROC^ del_IOWRITEITEMPROC =
gcnew OPCNetServerSDK::OPCNetServerSDKManager::Del_IOWRITEVQTPROC(WriteVQT);
```

## 4.15. GetErrorString

VOID GetErrorString ([in] DWORD hrError
, [in] DWORD dwLocal = 0
, [out] BSTR* errmsg)

**Description**

This function will be called by the DLL to interpret driver specific error codes that it does not understand. Standard OPC error codes are handled by the DLL.

| Parameter Name | Description |
|---|---|
| HrError | A driver specific error code that needs to be described. |
| Errmsg | NULL or a valid LPWSTR describing the given error code. |

**Table 73: GetErrorString Parameters**

**VB Wrapper**

```
Sub GetErrorString (ByVal arg1 As Long _
                    , ByVal arg2 As Long _
                    , ByRef msg As String)
```

**C# .NET**

```
delegate void Del_IOLOOKUPITEMIDSPROC(String strItemID ,
                                      UInt32 uintCount,
                                      out UInt32[] hPropertyIDs ,
                                      out IntPtr hr)
```

**Code example**

```
/*public static void LookupItemIDs(String strTagName,
                                  UInt32 uintCount,
                                  out UInt32[] hPropertyIDs ,
                                  out IntPtr hr)*/

OPCNetServerSDKManager.Del_IOLOOKUPITEMIDSPROC    del_IOLOOKUPITEMIDSPROC    =    new
OPCNetServerSDKManager.Del_IOLOOKUPITEMIDSPROC(LookupItemIDs);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOLOOKUPITEMIDSPROC(ByVal strItemID  As [String],
                                     ByVal uintCount As UInt32,
                                     ByRef hPropertyIDs  As UInt32(),
                                     ByRef hr As IntPtr)
```

**Code example**

```
'Public Shared Sub LookupItemIDs(ByVal strItemID  As String,
                                 'ByVal uintCount As UInt32,
                                 'ByRef hPropertyIDs  As UInt32(),
                                 'ByRef hr As IntPtr)

Dim del_IOLOOKUPITEMDSPROC As New
OPCNetServerSDKManager.Del_IOLOOKUPITEMIDSPROC(LookupItemIDs)
```

**C++ .NET**

```
delegate void Del_IOLOOKUPITEMIDSPROC(String^ strItemID ,
                                      UInt32 uintCount,
                                      array<UInt32^> %hPropertyIDs ,
                                      IntPtr %hr);
```

**Code example**

```
/*public static void LookupItemIDs(String^ strTagName,
                                  UInt32 uintCount,
                                  array<UInt32^> %hPropertyIDs ,
                                  IntPtr %hr)*/
```

```
OPCNetServerSDKManager.Del_IOLOOKUPITEMIDSPROC del_IOLOOKUPITEMIDSPROC = gcnew
OPCNetServerSDKManager.Del_IOLOOKUPITEMIDSPROC(LookupItemIDs);
```

## 4.16. ReadRawDatum

```
VOID ReadRawDatum(    [in] HANDLE          intItem
                     ,[in] FILETIME*      ftStart
                     ,[in] FILETIME*      ftEnd
                     ,[in] BOOL             bBounds
                     ,[in/out] long*        toRetrieve
                     ,[out] VARIANT** datavalues
                     ,[out] DWORD**     qualities
                     ,[out] FILETIME**  timestamps
                     ,[out] HRESULT*    hr)
```

**Description**

This function reads the values, qualities, and timestamps from the history for the specified time domain and the specified item.

| Parameter Name | Description |
|---|---|
| intItem | Identifies the tag of interest. |
| ftStart | The beginning of the history period to be read. |
| ftEnd | The end of the history period to be read. |
| bBounds | True if bounding values should be returned. |
| toRetrieve | The number of occurrences to be read. If history contains less data than requested, this parameter will just return the available number of occurrences. |
| datavalues | Array of VARIANT containing the occurrences values. |
| qualities | Array of DWORD containing the occurrences qualities. |
| timestamps | Array of FILETIME containing the occurrences times. |
| hr | Returned error code (see possible values in the next table). |

**Table 74: ReadRawDatum Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. |

| | |
|---|---|
| OPC_E_BADRIGHTS | The Item's Access Rights do not allow the operation. |
| OPC_E_INVALIDHANDLE | Invalid handle. |
| OPC_E_NODATA | There is no data within the specified parameters. |
| OPC_S_MOREDATA | There is more data satisfying the query than was returned. |
| E_INVALIDARG | Invalid argument. |
| E_OUTOFMEMORY | Out of memory. |
| E_FAIL | The function failed. The function was unsuccessful for this item. |

**Table 75: [hr] Possible Values for ReadRaw**

(Default behavior is E_FAIL).

**VB Wrapper**

```
Sub ReadRawDatum (  ByVal ItemHandle As Long _
                  , ByRef InTime1 As FILETIME _
                  , ByRef InTime2 As FILETIME _
                  , ByVal bounds As Integer _
                  , ByRef toRetrieve As Long _
                  , data() As Variant _
                  , qualities() As Long _
                  , times() As FILETIME _
                  , ByRef error As Long)
```

**C# .NET**

```
delegate void Del_IOREADRAWDATUMPROC(int intItem,
                ref System.Runtime.InteropServices.ComTypes.FILETIME ftStart,
                ref System.Runtime.InteropServices.ComTypes.FILETIME ftEnd,
                Int32 intBounds,
                ref Int32 intToRetrieve,
                out object[] objDatavalues,
                out UInt32[] uintQualities,
                out System.Runtime.InteropServices.ComTypes.FILETIME[] timestamps,
                out IntPtr hr)
```

**Code example**

```
/*public static void ReadRawDatum(
        int intItem,
        ref System.Runtime.InteropServices.ComTypes.FILETIME ftStart,
```

```
        ref System.Runtime.InteropServices.ComTypes.FILETIME ftEnd,
        Int32 intBounds,
        ref Int32 intToRetrieve,
        out object[] objDatavalues,
        out UInt32[] uintQualities,
    out System.Runtime.InteropServices.ComTypes.FILETIME[] timestamps,
        out IntPtr hr)*/

OPCNetServerSDKManager.Del_IOREADRAWDATUMPROC del_IOREADRAWDATUMPROC = new
OPCNetServerSDKManager.Del_IOREADRAWDATUMPROC(ReadRawDatum);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOREADRAWDATUMPROC(
ByVal intItem As Integer,
ByRef ftStart As System.Runtime.InteropServices.ComTypes.FILETIME,
ByRef ftEnd As System.Runtime.InteropServices.ComTypes.FILETIME,
ByVal intBounds As Int32,
ByRef intToRetrieve As Int32,
ByRef objDatavalues As Object(),
ByRef uintQualities As UInt32(),
ByRef timestamps As System.Runtime.InteropServices.ComTypes.FILETIME(), ByRef hr As
IntPtr)
```

**Code example**

```
'Public Shared Sub ReadRawDatum(
ByVal intItem As Integer,
ByVal ftStart As System.Runtime.InteropServices.ComTypes.FILETIME,
ByVal ftEnd As System.Runtime.InteropServices.ComTypes.FILETIME,
ByVal intBounds As Int32,
ByRef intToRetrieve As Int32,
ByRef objDatavalues As Object(),
ByRef uintQualities As UInt32(),
ByRef timestamps As System.Runtime.InteropServices.ComTypes.FILETIME(), ByRef hr As IntPtr)

Private del_IOREADRAWDATUMPROC As New
OPCNetServerSDKManager.Del_IOREADRAWDATUMPROC(AddressOf ReadRawDatum)
```

**C++ .NET**

```
delegate void Del_IOREADRAWDATUMPROC(int intItem,
 ref System::Runtime::InteropServices::ComTypes::FILETIME ftStart,
 ref System System::Runtime::InteropServices::ComTypes::FILETIME ftEnd,
 Int32 intBounds,
 ref Int32 intToRetrieve,
 array <object^> %objDatavalues,
 array <UInt32^> %uintQualities,
 array <System::Runtime::InteropServices::ComTypes::FILETIME^> %timestamps,
 IntPtr %hr)
```

**Code example**

```
/*public static void ReadRawDatum(
int intItem,
ref System::Runtime::InteropServices::ComTypes::FILETIME ftStart,
ref System::Runtime::InteropServices::ComTypes::FILETIME ftEnd,
Int32 intBounds,
ref Int32 intToRetrieve,
array<object^> %datavalues,
array<UInt32^> %qualities,
array<System::Runtime::InteropServices::ComTypes::FILETIME^> %timestamps,
IntPtr %hr)*/

OPCNetServerSDKManager::Del_IOREADRAWDATUMPROC del_IOREADRAWDATUMPROC = gcnew
OPCNetServerSDKManager::Del_IOREADRAWDATUMPROC(ReadRawDatum);
```

## 4.17. ReadProcessed

```
VOID        ReadProcessed (  [in] HANDLE    intItem
                           , [in] DWORD     Aggregate
                           , [in] FILETIME *time1
                           , [in] FILETIME *time2
                           , [out] VARIANT *value
                           , [out] DWORD    *ushoQuality
                           , [out] FILETIME *mintime  = NULL
                           , [out] FILETIME *maxtime = NULL
                           , [out] HRESULT *hr)
```

**Description**

This function computes aggregate value, ushoQuality and timestamp for a given tag, at a specified time domain.

| Parameter Name | Description |
|---|---|
| intItem | Identifies the tag of interest. |
| Aggregate | Identifies the aggregate of interest. |
| time1 | The beginning of the history period to be read. |
| time2 | The end of the history period to be read. |
| Value | Variant in which the item aggregate value is returned. |
| UshoQuality | The ushoQuality of the returned data. |
| Hr | Returned error code (see possible values in the next table). |

**Table 76: ReadProcessed Parameters**

| Code | Description |
|---|---|

| | |
|---|---|
| S_OK | The function succeeded. |
| OPC_E_NOT_AVAIL | The requested aggregate is not available from the provided item. |
| E_FAIL | The function failed. The function was unsuccessful for this item. |
| S_xxx (success code)<br>E_xxx (error code) | Vendor specific information. |

**Table 77: [hr] Possible Values for ReadProcessed**

Default behavior returns E_FAIL.

**VB Wrapper**

```
Sub ReadProcessed (ByVal ItemHandle As Long _
                   , ByVal Aggregate As Long _
                   , ByRef InTime1 As FILETIME _
                   , ByRef InTime2 As FILETIME _
                   , ByRef Value As Variant _
                   , ByRef UshoQuality As Long _
                   , ByRef OutTime1 As FILETIME _
                   , ByRef OutTime2 As FILETIME _
                   , ByRef error As Long)
```

**C# .NET**

```
delegate void Del_IOREADPROCESSEDPROC(
                int intItem,
                UInt32 uintAggregate,
                ref System.Runtime.InteropServices.ComTypes.FILETIME time1,
                ref System.Runtime.InteropServices.ComTypes.FILETIME time2,
                out object objValue,
                out UInt32 uintQual,
                out System.Runtime.InteropServices.ComTypes.FILETIME mintime,
                out System.Runtime.InteropServices.ComTypes.FILETIME maxtime,
                out IntPtr hr)
```

**Code example**

```
/*static void ReadProcessed(int intItem,
        UInt32 uintAggregate,
        ref System.Runtime.InteropServices.ComTypes.FILETIME time1,
        ref System.Runtime.InteropServices.ComTypes.FILETIME time2,
        out object objValue,
```

```
        out UInt32 uintQual,
        out System.Runtime.InteropServices.ComTypes.FILETIME mintime,
        out System.Runtime.InteropServices.ComTypes.FILETIME maxtime,
        out IntPtr hr)*/

OPCNetServerSDKManager.Del_IOREADPROCESSEDPROC del_IOREADPROCESSEDPROC = new
OPCNetServerSDKManager.Del_IOREADPROCESSEDPROC(ReadProcessed);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOREADPROCESSEDPROC(
    ByVal intItem As Integer,
    ByVal Aggregate As UInt32,
    ByRef time1 As System.Runtime.InteropServices.ComTypes.FILETIME,
    ByRef time2 As System.Runtime.InteropServices.ComTypes.FILETIME,
    ByRef value As Object,
    ByRef uintQual As UInt32,
    ByRef mintime As System.Runtime.InteropServices.ComTypes.FILETIME,
    ByRef maxtime As System.Runtime.InteropServices.ComTypes.FILETIME,
    ByRef hr As IntPtr)
```

**Code example**

```
'Public Shared Sub ReadProcessed(ByVal intItem As Integer,
ByVal Aggregate As UInt32,
ByVal time1 As System.Runtime.InteropServices.ComTypes.FILETIME,
ByVal time2 As System.Runtime.InteropServices.ComTypes.FILETIME,
ByRef value As Object,
ByRef uintQual As UInt32,
ByRef mintime As System.Runtime.InteropServices.ComTypes.FILETIME, ByRef maxtime As
System.Runtime.InteropServices.ComTypes.FILETIME, ByRef hr As IntPtr)

Private del_IOREADPROCESSEDPROC As New
OPCNetServerSDKManager.Del_IOREADPROCESSEDPROC(AddressOf ReadProcessed)
```

**C++ .NET**

```
delegate void Del_IOREADPROCESSEDPROC(
        int intItem,
        UInt32 Aggregate,
        System::Runtime::InteropServices::ComTypes::FILETIME %time1,
        System::Runtime::InteropServices::ComTypes.FILETIME %time2,
        object %value,
        UInt32 %uintQual,
        System::Runtime::InteropServices::ComTypes::FILETIME %mintime,
        System::Runtime::InteropServices::ComTypes.FILETIME %maxtime,
        IntPtr %hr)
```

**Code example**

```
/*static void ReadProcessed(int intItem,
```

```
        UInt32 uintAggregate,
        System::Runtime::InteropServices::ComTypes::FILETIME %time1,
        System::Runtime::InteropServices::ComTypes.FILETIME %time2,
        object %objValue,
        UInt32 %uintQual,
        System::Runtime::InteropServices::ComTypes::FILETIME %mintime,
        System::Runtime::InteropServices::ComTypes.FILETIME %maxtime,
        %hr)*/

OPCNetServerSDKManager::Del_IOREADPROCESSEDPROC del_IOREADPROCESSEDPROC =
gcnew OPCNetServerSDKManager::Del_IOREADPROCESSEDPROC(ReadProcessed);
```

## 4.18. ValidAggregate

VOID ValidAggregate ( [in] HANDLE intItem
,[in] DWORD aggregate
,[out] BOOL bReturnVal)

**Description**

This function checks if the given aggregate could be performed or not for the specified item.

| Parameter Name | Description |
|---|---|
| IntItem | Identifies the tag of interest. |
| Aggregate | Identifies the aggregate of interest. |
| BReturnVal | TRUE: the given aggregate is valid.<br>FALSE: the given aggregate is not valid. |

**Table 78: ValidAggregate Parameters**

**VB Wrapper**

```
Sub ValidateAggregate (ByVal ItemHandle As Long _
                     , ByVal aggid As Long _
                     , ByRef bResult As Integer)
```

**C# .NET**

```
delegate void Del_IOVALIDAGGREGATEPROC(int intItem,
                                       UInt32 uintAggregate,
                                       out Int32 intReturn)
```

**Code example**

```
/*static void ValidAggregate(int intItem,
                            UInt32 uintAggregate,
                            out Int32 intReturn)*/
```

```
OPCNetServerSDKManager.Del_IOVALIDAGGREGATEPROC del_IOVALIDAGGREGATEPROC = new
OPCNetServerSDKManager.Del_IOVALIDAGGREGATEPROC(ValidAggregate);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOVALIDAGGREGATEPROC(ByVal intItem As Integer, ByVal uintAggregate
As UInt32, ByRef intReturn As Int32)
```

**Code example**

```
'Public Shared Sub ValidAggregate(ByVal intItem As Integer,
                        'ByVal aggregate As UInt32,
                        'ByRef intReturn As Int32)

Private del_IOVALIDAGGREGATEPROC As New
OPCNetServerSDKManager.Del_IOVALIDAGGREGATEPROC(AddressOf ValidAggregate)
```

**C++ .NET**

```
delegate void Del_IOVALIDAGGREGATEPROC(int intItem,
                                UInt32 uintAggregate,
                                Int32 %intReturn)
```

**Code example**

```
/*static void ValidAggregate(int intItem,
                        UInt32 uintAggregate,
                        Int32 % intReturn)*/

OPCNetServerSDKManager::Del_IOVALIDAGGREGATEPROC del_IOVALIDAGGREGATEPROC = new
OPCNetServerSDKManager::Del_IOVALIDAGGREGATEPROC(ValidAggregate);
```

## 4.19. ReadAtTime

```
VOID ReadAtTime(   [in] HANDLE       intItem
                  ,[in] FILETIME   *TimeStamp
                  ,[out] VARIANT *objValue
                  ,[out] DWORD    *pUshoQuality
                  ,[out] HRESULT *hr)
```

**Description**

This function reads the value and ushoQuality for a given item, at a specific timestamp.

| Parameter Name | Description |
|---|---|
| IntItem | Identifies the tag of interest |
| TimeStamp | The timestamp for the requested data |
| ObjValue | Variant in which the item value is returned. |
| PUshoQuality | The item ushoQuality |

| | |
|---|---|
| Hr | Returned error code (see possible values in the next table) |

**Table 79: ReadAtTime Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| E_FAIL | The function failed. The function was unsuccessful for this item. |
| OPC_E_INVALIDHANDLE | Invalid handle. |
| S_xxx (success code)<br>E_xxx (error code) | Vendor specific information. |

**Table 80: [hr] Possible Values for ReadAtTime**

(Default behavior is returning E_FAIL).

**VB Wrapper**

```
Sub ReadAtTime (ByVal ItemHandle As Long, _
                ByRef Time As FILETIME, _
                ByRef Value As Variant, _
                ByRef UshoQuality As Long, _
                ByRef error As Long)
```

**C# .NET**

```
delegate void Del_IOREADATTIMEPROC(
        int intItem,
        ref System.Runtime.InteropServices.ComTypes.FILETIME ftTimeToReadAt,
        out object objValue,
        out UInt16 ushoQuality,
        out IntPtr hr)
```

**Code example**

```
/*static void ReadAtTime(
   int intItem,
   ref System.Runtime.InteropServices.ComTypes.FILETIME ftTimeToReadAt,
   out object objValue,
   out UInt16 ushoQuality,
   out IntPtr hr)*/

OPCNetServerSDKManager.Del_IOREADATTIMEPROC del_IOREADPROCESSEDPROC = new
OPCNetServerSDKManager.Del_IOREADATTIMEPROC (ReadAtTime);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOREADATTIMEPROC(
    ByVal intItem As Integer,
    ByRef time1 As System.Runtime.InteropServices.ComTypes.FILETIME,
    ByRef objValue As Object,
    ByRef ushoQuality As UInt16,
    ByRef hr As IntPtr)
```

**Code example**

```
/*static void ReadAtTime(int intItem,
  ref System.Runtime.InteropServices.ComTypes.FILETIME ftTimeToReadAt,
  out object objValue,
  out UInt16 ushoQuality,
  out IntPtr hr)*/

Private del_IOREADATTIMEPROC As New
OPCNetServerSDKManager.Del_IOREADATTIMEPROC(AddressOf ReadAtTime)
```

**C++ .NET**

```
delegate void Del_IOREADATTIMEPROC(int intItem,
        System::Runtime::InteropServices::ComTypes::FILETIME %time1,
        object %objValue,
        UInt16 %ushoQuality,
        IntPtr %hr)
```

**Code example**

```
/*static void ReadAtTime(int intItem,
    System.Runtime.InteropServices.ComTypes.FILETIME %ftTimeToReadAt,
    object %objValue,
    UInt16 %ushoQuality,
    IntPtr %hr)*/

OPCNetServerSDKManager::Del_IOREADATTIMEPROC Del_IOREADPROCESSEDPROC1 = new
OPCNetServerSDKManager::Del_IOREADATTIMEPROC (ReadAtTime);
```

## 4.20. ReadCurrentAttribute

```
VOID ReadCurrentAttribute(  [in] HANDLE    intItem
                           ,[in]DWORD     UintAttributeID
                           ,[out] VARIANT *objValue
                           ,[out] HRESULT*  hr)
```

**Description**

This function reads the current attribute value for the specified tag and attribute.

| Parameter Name | Description |
|---|---|
| intItem | Identifies the tag of interest. |
| UintAttributeID | Identifies the attribute of interest. |
| objValue | Variant in which the item's attribute value is returned. |
| Hr | Returned error code (see possible values in the next table). |

**Table 81: ReadCurrentAttribute Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| OPC_E_INVALIDATTRID | The attribute ID is not valid. |
| E_FAIL | The function failed. The function was unsuccessful for |

| | |
|---|---|
| | this item. |
| S_xxx (success code)<br>E_xxx (error code) | Vendor specific information. |

**Table 82: [hr] Possible Values for ReadCurrentAttribute**

(Default behavior is returning E_FAIL).

**VB Wrapper**

```
Sub ReadCurrentAttribute (ByVal ItemHandle As Long _
                        , ByVal AttrID As Long _
                        , ByRef ptValue As Variant _
                        , ByRef error As Long)
```

**C# .NET**

```
delegate void Del_IOREADCURRENTATTRIBUTEPROC(int intItem,
                                    UInt32 uintUintAttributeID,
                                    out object objValue,
                                    out IntPtr hr);
```

**Code example**

```
/*static void ReadCurrentAttribute(int intItem,
                        UInt32 uintUintAttributeID,
                        out object objValue,
                        out IntPtr hr)*/

OPCNetServerSDKManager.Del_IOREADCURRENTATTRIBUTEPROC
del_IOREADCURRENTATTRIBUTEPROC = new
OPCNetServerSDKManager.Del_IOREADCURRENTATTRIBUTEPROC(ReadCurrentAttribute);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOREADCURRENTATTRIBUTEPROC (ByVal intItem As Integer,
                                    ByVal uintUintAttributeID As UInt32,
                                    ByRef objValue As Object,
                                    ByRef hr As IntPtr)
```

**Code example**

```
'ReadCurrentAttribute(ByVal intItem As Integer,
                ByVal uintUintAttributeID As UInt32,
                ByRef objValue As Object,
                ByRef hr As IntPtr)

Dim del_IOREADCURRENTATTRIBUTEPROC As New
OPCNetServerSDKManager.Del_IOREADCURRENTATTRIBUTEPROC(AddressOf
ReadCurrentAttribute)
```

**C++ .NET**

```
delegate void Del_IOREADCURRENTATTRIBUTEPROC(int intItem,
                                             UInt32 uintUintAttributeID,
                                             object %objValue,
                                             IntPtr %hr)
```

**Code example**

```
/*static void ReadCurrentAttribute(int intItem,
                                  UInt32 UintAttributeID,
                                  out object objValue,
                                  out IntPtr hr)*/

OPCNetServerSDKManager::Del_IOREADCURRENTATTRIBUTEPROC
del_IOREADCURRENTATTRIBUTEPROC = gcnew
OPCNetServerSDKManager::Del_IOREADCURRENTATTRIBUTEPROC(ReadCurrentAttribute);
```

## 4.21. InsertIntoHistorian

```
VOID InsertIntoHistorian (   [in] HANDLE   intItem
                            ,[in] FILETIME  ftTimeStamp
                            ,[in] VARIANT  vDataValue
                            ,[in] DWORD    dwUshoQuality
                            ,[in] long          lInsertionMode
                            ,[out] HRESULT* hr)
```

**Description**

This function allows updating history data for a given item according to the specified mode.

| Parameter Name | Description |
|---|---|
| intItem | Identifies the tag of interest. |
| ftTimeStamp | The timestamp for the new value. |
| vDataValue | The new item value. |
| dwUshoQuality | The ushoQuality flag for the new value. |
| lInsertionMode | Identifies the mode of update.<br>0: Add unless event(s) exist at same time.<br>1: Replace existing event (fail if no event at time).<br>2: Add event, replace if event at same time. |
| Hr | Returned error code (see possible values in the next table). |

**Table 83: InsertIntoHistorian Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| OPC_E_BADRIGHTS | The Items Access Rights do not allow the operation. |
| OPC_E_INVALIDHANDLE | Invalid handle. |
| OPC_E_DATAEXISTS | Unable to insert - data already present. |
| OPC_E_NODATAEXISTS | The server has no value for the specified time and Item ID. |
| OPC_S_INSERTED | The requested insert occurred. |
| OPC_S_REPLACED | The requested replace occurred. |
| E_INVALIDARG | Invalid argument. |
| E_FAIL | The function failed. The function was unsuccessful for this item. |

**Table 84: [hr] Possible Values for InsertIntoHistorian**

**VB Wrapper**

```
Sub InsertIntoHistorian (ByVal ItemHandle As Long _
                        , Time As FILETIME _
                        , ByVal Value As Variant _
                        , ByVal UshoQuality As Long _
                        , ByVal InsertionMode As Long _
                        , ByRef error As Long)
```

**C# .NET**

```
delegate void Del_IOINSERTINTOHISTORIANPROC(
        int intItem,                    System.Runtime.InteropServices.ComTypes.FILETIME
        ftTimeStamp,
        object objDataValue,
        UInt16 ushoQuality,
        int intInsertionMode,
        out IntPtr hr)
```

**Code example**

```
/*public static void InsertIntoHistorian(int intItem,
        System.Runtime.InteropServices.ComTypes.FILETIME ftTimeStamp,
        object objDataValue,
        UInt16 ushoQuality,
        int intInsertionMode,
        out IntPtr hr)*/
```

```
OPCNetServerSDKManager.Del_IOINSERTINTOHISTORIANPROC
del_IOINSERTINTOHISTORIANPROC = new
OPCNetServerSDKManager.Del_IOINSERTINTOHISTORIANPROC(InsertIntoHistorian);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOINSERTINTOHISTORIANPROC(
        ByVal intItem As Integer,
        ByVal ftTimeStamp As System.Runtime.InteropServices.ComTypes.FILETIME,
        ByVal vDataValue As Object,
        ByVal dwUshoQuality As UInt16,
        ByVal intInsertionMode As Integer,
        ByRef hr As IntPtr)
```

**Code example**

```
'Public Shared Sub InsertIntoHistorian(
 ByVal intItem As Integer,
 ByVal ftTimeStamp As System.Runtime.InteropServices.ComTypes.FILETIME,
 ByVal objDataValue As Object,
 ByVal ushoQuality As UInt16,
 ByVal intInsertionMode As Integer,
 ByRef hr As IntPtr)

Private del_IOINSERTINTOHISTORIANPROC As New
OPCNetServerSDKManager.Del_IOINSERTINTOHISTORIANPROC(AddressOf InsertIntoHistorian)
```

**C++ .NET**

```
delegate void Del_IOINSERTINTOHISTORIANPROC(
        int                                                    intItem,
        System::Runtime::InteropServices::ComTypes::FILETIME ftTimeStamp,
        object objDataValue,
        UInt16 ushoQuality,
        int intInsertionMode,
        IntPtr %hr)
```

**Code example**

```
/*public static void InsertIntoHistorian(
        int intItem,
        System.Runtime.InteropServices.ComTypes.FILETIME ftTimeStamp,
        object objDataValue,
        UInt16 ushoQuality,
        int intInsertionMode,
        IntPtr %hr)*/

OPCNetServerSDKManager::Del_IOINSERTINTOHISTORIANPROC
del_IOINSERTINTOHISTORIANPROC = gcnew
OPCNetServerSDKManager::Del_IOINSERTINTOHISTORIANPROC(InsertIntoHistorian);
```

## 4.22. RemoveFromHistorian

VOID RemoveFromHistorian ( [in] HANDLE intItem
,[in] FILETIME *ftStartTime
,[in] FILETIME *ftEndTime
,[in] int iHistoryPeriod
,[out] HRESULT* hr)

**Description**

| Parameter Name | Description |
|---|---|
| intItem | Identifies the tag of interest. |
| ftStartTime | The beginning of the history period to be deleted, or the timestamp at which you wish to perform data deletion. |
| ftEndTime | The end of the history period to be deleted. |
| iHistoryPeriod | Identifies the mode of deletion:<br>0: Delete data at a specified time (ftStartTime).<br>1: Delete data for a specified time domain [ftStartTime, ftEndTime]. |
| hr | Returned error code (see possible values in the next table). |

**Table 85: RemoveFromHistorian Parameters**

| Code | Description |
|---|---|
| S_OK | The function succeeded. |
| OPC_E_BADRIGHTS | The Items Access Rights do not allow the operation. |
| OPC_E_INVALIDHANDLE | Invalid handle. |
| OPC_E_NODATA | There is no data within the specified parameters. |
| E_INVALIDARG | Invalid argument. |
| E_FAIL | The function failed. The function was unsuccessful for this item. |

**Table 86: [hr] Possible Values for RemoveFromHistorian**

**VB Wrapper**

Sub RemoveFromHistorian (ByVal ItemHandle As Long _

```
                                , ByRef InTime1 As FILETIME _
                                , ByRef InTime2 As FILETIME _
                                , ByVal Mode As Integer _
                                , ByRef error As Long)
```

**C# .NET**

```
delegate void Del_IOREMOVEFROMHISTORIANPROC(
                int intItem,
                ref System.Runtime.InteropServices.ComTypes.FILETIME ftStartTime,
                ref System.Runtime.InteropServices.ComTypes.FILETIME ftEndTime,
                int intHistoryPeriod,
                out IntPtr hr)
```

**Code example**

```
/*public static void RemoveFromHistorian(
        int intItem,
        ref System.Runtime.InteropServices.ComTypes.FILETIME ftStartTime,
        ref System.Runtime.InteropServices.ComTypes.FILETIME ftEndTime,
        int intHistoryPeriod,
        out IntPtr hr)*/

OPCNetServerSDKManager.Del_IOREMOVEFROMHISTORIANPROC
del_IOREMOVEFROMHISTORIANPROC = new
OPCNetServerSDKManager.Del_IOREMOVEFROMHISTORIANPROC(RemoveFromHistorian);
```

**VB .NET Wrapper**

```
Delegate Sub Del_IOREMOVEFROMHISTORIANPROC(
 ByVal intItem As Integer,
 ByRef ftStartTime As System.Runtime.InteropServices.ComTypes.FILETIME,
 ByRef ftEndTime As System.Runtime.InteropServices.ComTypes.FILETIME,
 ByVal intHistoryPeriod As Integer,
 ByRef hr As IntPtr)
```

**Code example**

```
'Public Shared Sub RemoveFromHistorian(
'ByVal intItem As Integer,
'ByVal ftStartTime As System.Runtime.InteropServices.ComTypes.FILETIME, 'ByVal ftEndTime As
System.Runtime.InteropServices.ComTypes.FILETIME, 'ByVal intHistoryPeriod As Integer,
'ByRef hr As IntPtr)

Private del_IOREMOVEFROMHISTORIANPROC As New
OPCNetServerSDKManager.Del_IOREMOVEFROMHISTORIANPROC(AddressOf
RemoveFromHistorian)
```

**C++ .NET**

```
delegate void Del_IOREMOVEFROMHISTORIANPROC(
        int intItem,
        System.Runtime.InteropServices.ComTypes.FILETIME %ftStartTime,
        System.Runtime.InteropServices.ComTypes.FILETIME %ftEndTime,
        int intHistoryPeriod,
        IntPtr %hr)
```

**Code example**

```
/*public static void RemoveFromHistorian(
        int intItem,
        System.Runtime.InteropServices.ComTypes.FILETIME %ftStartTime,
        System.Runtime.InteropServices.ComTypes.FILETIME %ftEndTime,
        int intHistoryPeriod, IntPtr %hr)*/

OPCNetServerSDKManager ::Del_IOREMOVEFROMHISTORIANPROC
del_IOREMOVEFROMHISTORIANPROC = new
OPCNetServerSDKManager ::Del_IOREMOVEFROMHISTORIANPROC(RemoveFromHistorian);
```

# 5. Helper Functions

In this section, we present helper functions that make server development easier.
There are three categories of helper functions; they are presented in the following table:

| Category | Helper Functions | Description |
|---|---|---|
| Memory Management | io_allocmemory<br>io_reallocmemory<br>io_freememory | In some methods, memory is allocated inside the DLL then freed by the OPC server simulator and vice versa. The IMalloc object, that manages memory, needs to be shared between the DLL and the server application through exported functions. |
| Tracing function | io_traceevent | We need to record errors and debugging information for both the DLL and the simulator. Tracing the activity of the whole server in a log file can be extremely valuable for troubleshooting. |
| Utilities | io_changetype<br>io_opcvariantcopy<br>io_opcvariantclear | These utilities are for VARIANT management. |

| | | |
|---|---|---|
| | io_matchtohdaushoQuality | This function matches DA ushoQuality with HDA ushoQuality (for HDA use). |

**Table 87: Helper Functions**

## Memory Management

### 5.1.1. io_allocmemory

void *io_allocmemory( [in] ULONG cb )

**Description**

This call will allocate a block of memory that is accessible by the DLL and can be successfully freed by the DLL.

| Parameter Name | Description |
|---|---|
| cb | The size of the memory block to allocate, in bytes. |

**Table 88: Io_allocmemory Parameters**

**Return Values**

A pointer to the allocated block of memory is returned. Upon error, the return value is NULL.

### 5.1.2. io_reallocmemory

void *io_reallocmemory( [in] void *pv
, [in] ULONG cb)

**Description**

io_reallocmemory ( ) reallocates a block of memory. The pv argument points to the beginning of the memory block. If pv is NULL, io_reallocmemory ( ) allocates a new memory block in the same way that io_allocmemory( ) does. If pv is not NULL, it should be a pointer returned by a prior call to io_allocmemory( ).

The cb argument specifies the size (in bytes) of the new block. The contents of the block are unchanged up to the shorter of the new and old sizes, although the new block can be in a different location.

| Parameter Name | Description |
|---|---|
| cb | The size of the memory block to be reallocated, in bytes. |
| Pv | The pointer to the memory block to be reallocated. The pointer can have a NULL value. |

**Table 89: Io_reallocmemory Parameters**

**Return Values**
A pointer to the reallocated block of memory is returned. Upon error, the return value is NULL.

### 5.1.3. io_freememory

void io_freememory( [in] void *pv )

**Description**
This call will free a block of memory that was previously allocated using one of the DLL Memory Management helper functions. "pv" can be NULL, in this case the call has no effect.

| Parameter Name | Description |
|---|---|
| pv | The pointer to a block of memory previously allocated using io_allocmemory ( ) or io_reallocmemory ( ). |

**Table 90: Io_freememory Parameters**

## 5.2. Tracing function

The DLL tracing capabilities are described in the following section.
void io_traceevent ([in] int level, [in] char *pszDesc)

| Parameter Name | Description |
|---|---|
| level | The level of tracing. |
| pszDesc | Message to write in the log file. |

**Table 91: Io_traceevent Parameters**

**Return Codes**
None

**VB Wrapper**

```
Declare Sub io_traceevent Lib "DXServerDll.dll" _
                (ByVal level As Integer, _
                 ByVal pszDesc As String)
```

**C# .NET**

```
Void IO_TraceEvent(int level, string pszDesc)
```

## 5.3. Utilities

### 5.3.1. io_changetype

```
HRESULT io_changetype( [out] VARIANT*      vDst
                       ,[in] VARIANT       vSrc
```

```
,[in] LCID          lcid
,[in] VARTYPE       vtType)
```

**Description**

This function extends the role of VariantChangeTypeEx( ) to support complex types of variant (arrays).

| Parameter Name | Description |
|---|---|
| vDst | A pointer to the coerced argument (simple or complex type of variant). If this is the same as vSrc, the variant will be converted in place. |
| vSrc | The argument to be coerced. |
| Lcid | The LCID for the variant to coerce. The LCID is useful when the type of the source or destination VARIANTARG is VT_BSTR, VT_DISPATCH, or VT_DATE. |
| vtType | The type to coerce to. If the return code is S_OK, the vt field of the vDst is guaranteed to be equal to this value. |

**Table 92: Io_changetype Parameters (*)**

(*) These descriptions were taken from MSDN documentation.

**VB Wrapper**

```
Declare Function io_changetype Lib "DXServerDll.dll" _
                    (ByRef cDst As Variant, _
                     ByVal cSrc As Variant, _
                     ByVal lcid As Integer, _
                     ByVal vtType As Integer) As Long
```

### 5.3.2. io_opcvariantcopy

```
HRESULT io_opcvariantcopy (   VARIANT* vDst
                            , VARIANT vSrc)
```

**Description**

Initialize the target VARIANT (vDst) and make a copy of source VARIANT (vSrc).

### 5.3.3. io_opcvariantclear

```
HRESULT io_opcvariantclear(VARIANT* vVar)
```

**Description**

Clear a given variant.

### 5.3.4. io_matchtohdaushoQuality

```
BOOL io_matchtohdaushoQuality ( [in] long          DAFlag
                              ,[in] long           xxx_qual_status
                              ,[in] WORD           HDAQual
                              ,[out] DWORD* HDAQualR)
```

**Description**

This function matches Data Access ushoQuality with Historical Data Access ushoQuality giving the server specific ushoQuality.

| Parameter Name | Description |
|---|---|
| DAFlag | Data Access Flag. |
| xxx_qual_status | Server specific ushoQuality. |
| HDAQual | History Data Access UshoQuality. |
| HDAQualR | The HDA ushoQuality that results in DA, server and HDA flags' combination. |

**Table 93: Io_matchtohdaushoQuality Parameters**

This function returns Boolean values:

- TRUE: The combination of the given flags succeeded.
- FALSE: The combination of the given flags failed.

# TOOLKIT TRACING CAPABILITIES

The DLL has tracing capabilities. Developer can record the DLL errors and debugging information (COM and OPC messages) in a log file named "SrvToolkit_LogEvent.LOG". If difficulties occur with the DLL the log file can be extremely valuable for troubleshooting. Under normal operation, the DLL logs very little information.

This log file is generated at the start-up where server executable is located. The toolkit incorporates a configuration file "SrvToolkit_CfgFile.ini" which includes several logging parameters. All these parameters have default settings and can be changed at start-up by editing the configuration file.
To change this file:

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

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

**Table 94: Configuration File Parameters**

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

**Sample Configuration File**

```
[LogSetting]
LogFileName = SrvToolkit_LogEvent
```

**LogFileMaxSize** = 2097152

**LogLevel** = 0

**ArchiveLastLog** = False

**LogFileFolderPath** =

# IOPCSIMULATOR SAMPLES

The OPC Server Toolkit package includes several OPC Server samples. For example, the C#.NET and VC6 based OPC servers for simulation are compliant and self-test certified for both OPC DA and OPC HDA specifications.
In this chapter, we describe the main features of the following simulators:

- The MFC sample,
- The C# .NET sample,
- And the service samples implemented in C++ .NET and C# .NET.

## 1. Executing the MFC Sample

### 1.1. Server Registration

This server is registered manually by running one of the following line commands (Start Menu ➔ Run):
**"Simulator_EXE_Path" /regserver or "Simulator_EXE_Path" –regserver.**

So, new entries are added to the Windows registry with "**IntegrationObjects.DAHDASimulatorC++VS2008.1**" progID (server name).

To remove server entries from the registry, you should type one of the following line commands (Start Menu ➔ Run):
**"Simulator_EXE_Path" /unregserver" or "Simulator_EXE_Path" –unregserver.**

### 1.2. Server Features

In case of a demo version, this sample server runs for 2 hours. After that, it will be suspended. You should launch it again for a new session.

The main window looks like:

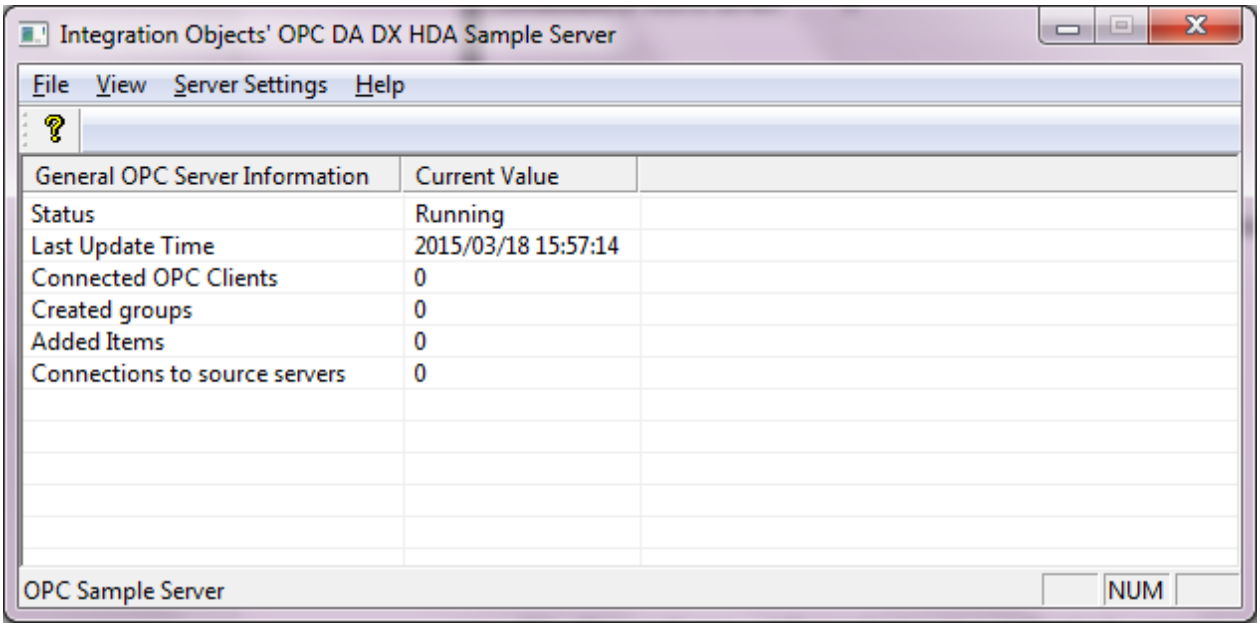


**Figure 23: Main Interface**

**General OPC server information**

- Status.
- Last update time.
- Connected OPC clients.
- Created groups.
- Added items.
- Connections to source servers.

All the above information is collected using io_getserverstatistic ( ) and io_getlastupdatetime ( ).

**View menu**

This menu contains only one item: “Log File”. We can view all log messages (of both the DLL and the server).

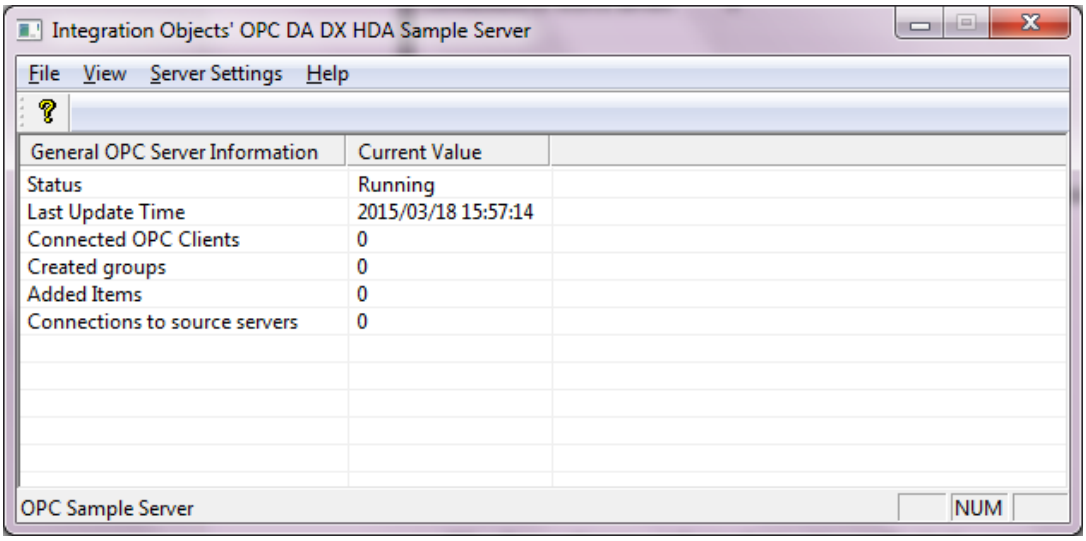


**Figure 24: OPC Server Statistics**

### Server Settings menu

This menu contains one menu item: "Change Status". It allows the user to change the OPC server status.

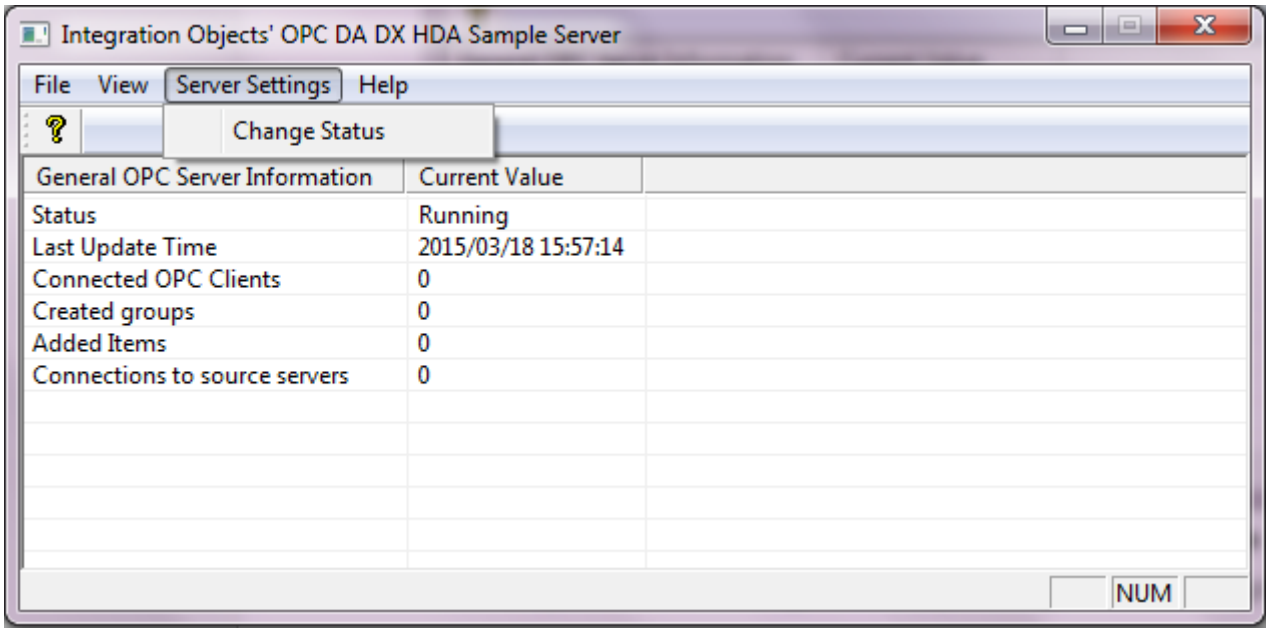


**Figure 25: Change Server Status**

If you click on "Change Status", you will get the following dialog box:

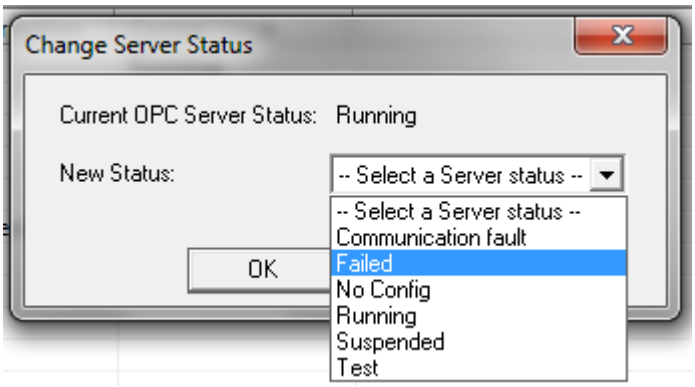


**Figure 26: Server Status**

# 2. The MFC Application Architecture

In this section, we describe the main implemented classes in the MFC OPC server sample.

## 2.1. Main Lists

In this sample, we used a local map list "**MapTAGITEM"** that holds all simulated Data Access tags. This map holds <string (full tag name), TAGITEM structure> pairs.

For Historical Data Access simulation, we also used a local map list "**TagMappedByHandle**". This map holds <HANDLE (item handle), MyOPCHDA_ITEM*> pairs. The OPCHDA_ITEM is a structure that contains, for a given item handle, a list of values, timestamps, and qualities. This list is updated every time Data Access item changes. Using a local list means that data is not archived in a permanent storage support such as a file, database, etc. The server will lose all the data when it restarts.

**Note that the hClient field in the MyOPCHDA_ITEM structure does not have any relation with the 'hClient definition' used in OPC server/client transactions (such as ReadAtTime method).**

## 2.2. Classes

**CIODXserverSimulatorApp**

The main methods are:

- InitInstance( ): Contains the main program. In this method, you will find initialization of the DLL and configuration of items for DA and HDA services.
- ExitInstance( ): Here we free all allocated resources in the server application. The io_closeall( ) is also called in order to un-initialize the COM library and free resources allocated inside the DLL.

**CItemtree**

This class manages a map list that holds <string (full tag name), TAGITEM structure> pairs. We can add/update a node in the list. We can also search for an item giving its full name.

The following are the main methods:

1. Insert: Adds a node to the map.
2. FillInDAItem: Updates a node in the map.
3. FindItem: Searches in the list for the specified key (full tag name).

**IODXServerSimulatorView**

This class is used for the graphical user interface.

**MyListControl**

This class extends the CListCtrl MFC class. It contains the following methods:

1. Init: Initializes the collection.
2. AddItem: Adds an item to the collection.
3. AddColumn: Adds a column to the collection.

Data Access and Historical Data Access callbacks are implemented in the **SimCallBack.cpp** file.

This file also contains functions for building the DA server address space, HDA settings and for freeing resources.

The following are the main functions:

1. CreateBrowser: creates a local list that holds all available Data Access tags. It calls io_createtag to build the server address space in the OPC Server Toolkit.
2. SetHdaConfiguration: used for HDA initialization. It calls io_hdasethdaitems( ), io_hdasetitemaggregates ( ) and io_hdasetupdatemethods ( ).
3. SetHdaConfiguration uses the following local functions:
    a. GetTagAttributes: retrieves HDA attributes such as IO_OPCHDA_DATA_TYPE, IO_OPCHDA_ITEMID, etc.
    b. GetTags: retrieves items for which we want to archive values.
    c. GetSimAggregates: retrieves HDA aggregates such as IO_OPCHDA_INTERPOLATIVE, IO_OPCHDA_AVERAGE, etc.
4. FreeResources: used for server memory release by calling the io_freememory() and clearing the lists.

**ChangeStatusDialog**

This class manages a dialog box control that allows users to change server status.

# 3. Building the MFC Sample

## 3.1. VS 2012 & VS 2017 build

In order to successfully build the MFC sample in x64 / Win32 mode, the following conditions should be satisfied:

- Make sure to use the MFC in a static library and use the Multi-byte character set as shown below

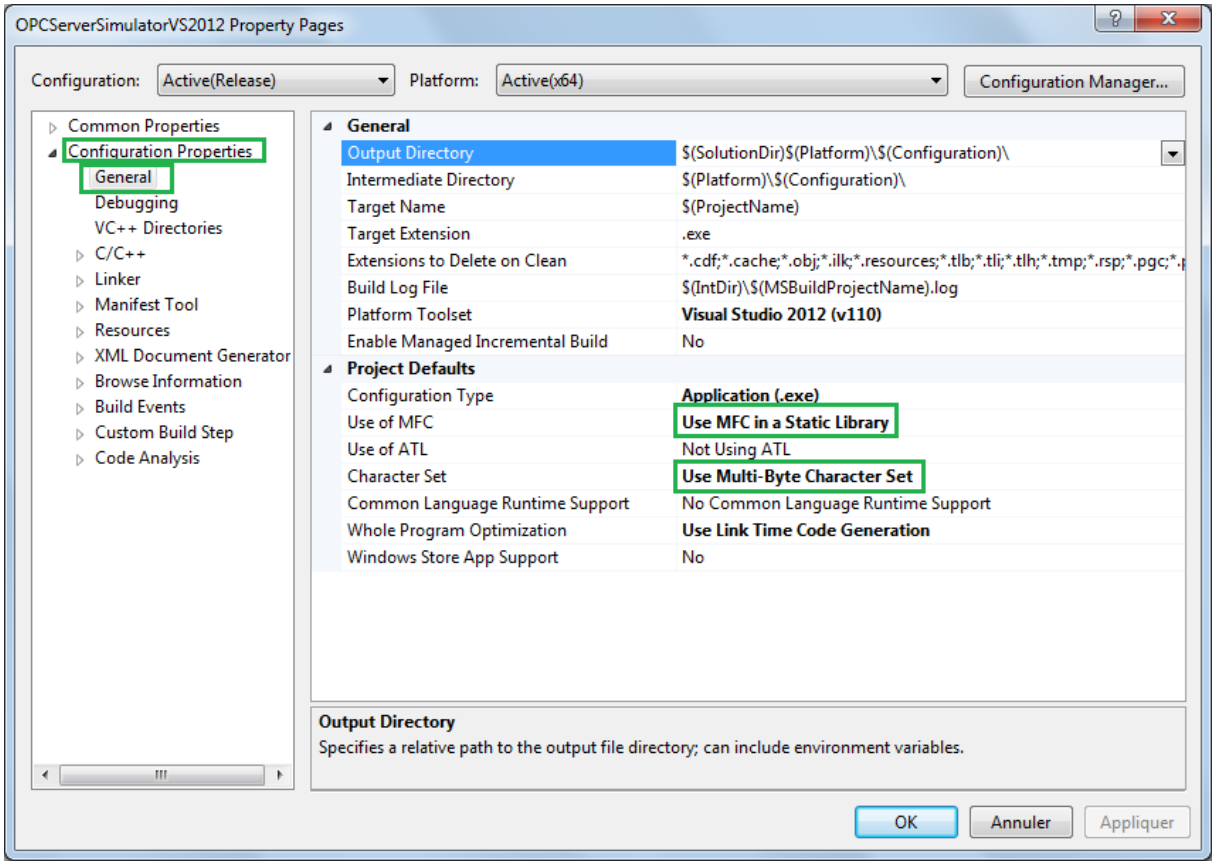


**Figure 27: Configuration Properties General Parameters**

- Make sure to choose the **No (/Zc:wchar_t-)** and **No (/Zc:forScope-)** in the C/C++ language parameters as shown below:

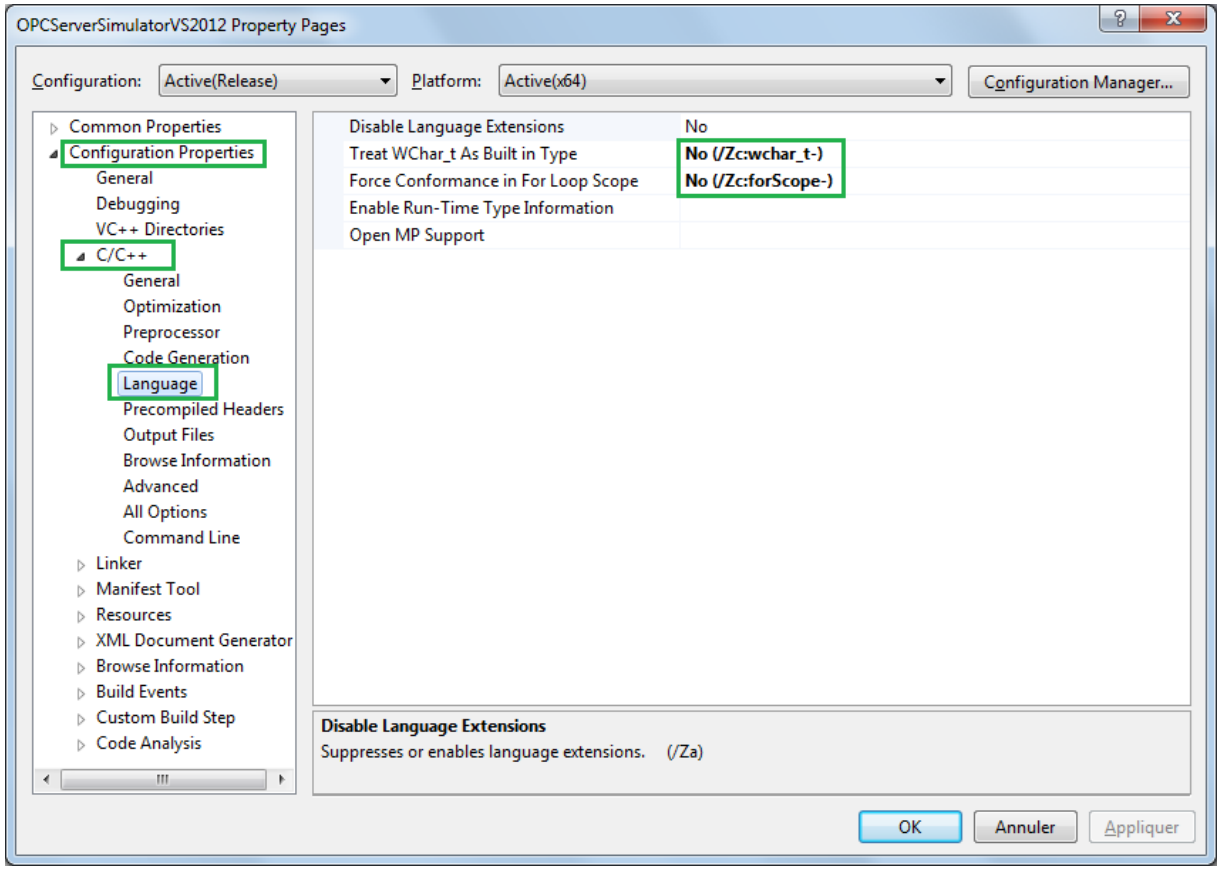


**Figure 28: C/C++ Language Parameters**

- Make sure to add the correct DXServerDll.lib, depending on the build mode, to the additional dependencies.
  - *X64 mode DXServerDll.lib path*: .\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\lib\x64
  - *Win32 mode DXServerDll.lib path*: .:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\lib\x86

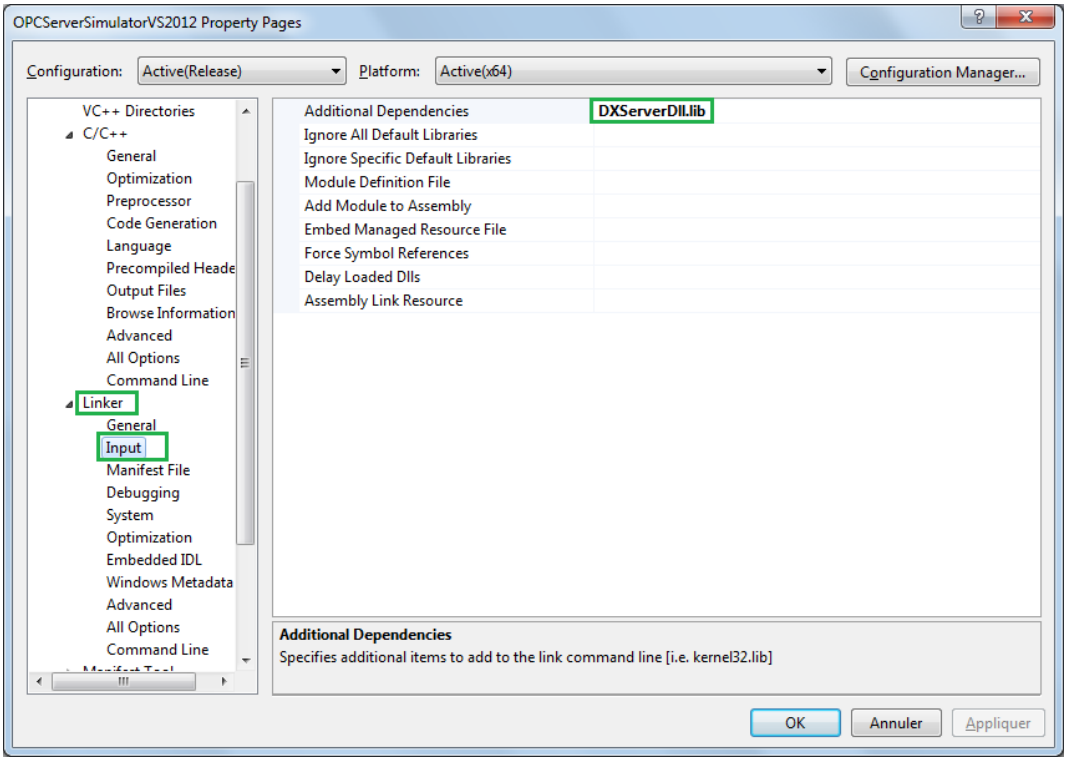


**Figure 29: DXServerDll.lib Additional Dependencies**

- Copy the following files in the output project folder with respect to the build mode:
  - *DXServerDll.dll*
  - *License.dll*
  - *OPCDADLL.dll*

  These files are included in the installation folder as described below:
  - *X64 build mode*: .:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\DLLs\x64
  - *Win32 build mode*: .:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\DLLs\x86

**In case you are running the MFC Sample project with Microsoft Visual Studio 2013 you need to install the MFC MBCS DLL Add-on from the below link to be able to build and run the application http://msdn.microsoft.com/library/dn251007.aspx**

# 4. Executing the C# .NET Sample

## 4.1. Server Registration

This server is registered manually by running one of the following line commands (Start Menu ➔ Run):

**"Simulator_EXE_Path" /regserver or "Simulator_EXE_Path" –regserver.**

So, new entries are added to the Windows registry with "**IntegrationObjects.DAHDASimulatorC#2008.1**" progID (server name).

To remove server entries from the registry, you should type one of the following line commands (Start Menu ➔ Run):
**"Simulator_EXE_Path" /unregserver" or "Simulator_EXE_Path" –unregserver.**

## 4.2. Server Features

In case of demo version, this sample server runs for 2 hours. After that, it will be suspended. You should launch it again for a new session.
The main window looks like:

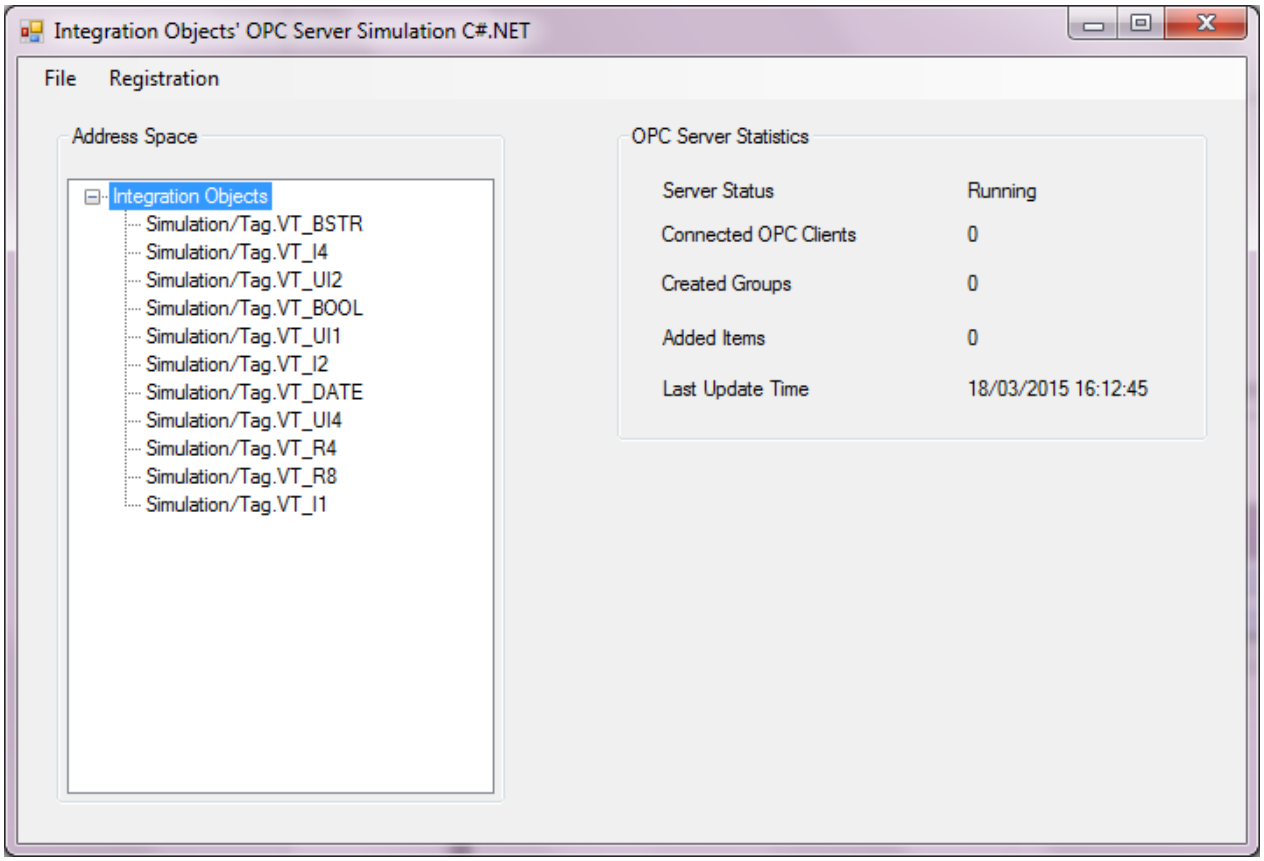


**Figure 30: Main Interface**

**General OPC server information**

- Status.
- Last update time.
- Connected OPC clients.
- Created groups.
- Added items.

All the above information is collected using io_getserverstatistic( ) and io_getlastupdatetime( ) methods.

**Server Address Space**

The address space of the server is built after the server startup. It can also be updated on runtime by right clicking the "Integration Objects" treeview node to add a new tag as shown below.

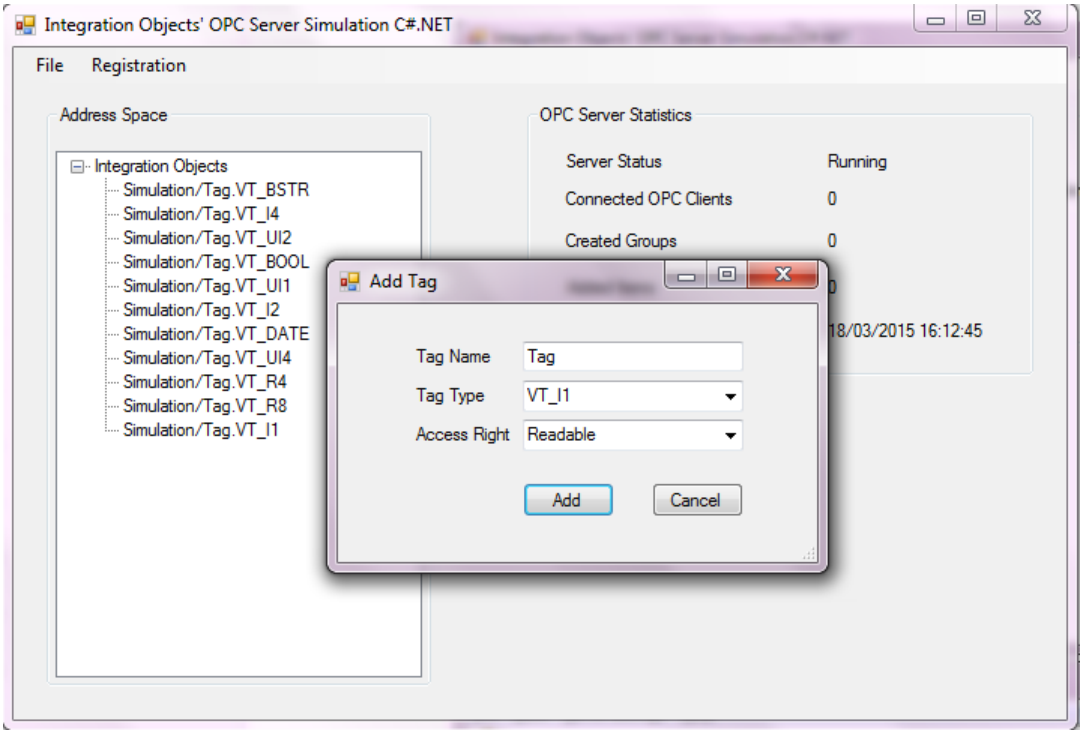


**Figure 31: Add New Tag**

**Parameters:**

1. Tag Name: Is the tag ID and it has to be unique
2. Tag Type: Is the data type. It can be:
   - VT_I1: data type is 1-byte signed short.
   - VT_I2: data type is 2-byte signed integer.
   - VT_I4: data type is 4-byte signed long.
   - VT_R4: data type is 4-byte real.
   - VT_R8: data type is 8-byte real.
   - VT_UI1: data type is an unsigned short.
   - VT_UI2: data type is an unsigned integer.
   - VT_UI4: data type is an unsigned long.
   - VT_BSTR: data type is binary string.
   - VT_BOOL: data type is Boolean.

- VT_DATE: data type is a date

3. Access Right: Is the access right. It can be :
   - Real Only
   - Or Read/Write

# 5. The C# .NET Application Architecture

In this section, we describe the main implemented classes in the C# .NET OPC server sample.

## 5.1. Main Lists

In this sample, we used a hashtable "**AddressSpace**" which represents a collection of <string (full tag name), TAGITEM structure> pairs that are organized based on the hash code of the key. The Address Space holds all simulated OPC DA tags.

For historical data simulation, we also used a hashtable "**TagNameMappedByHandle**". This hashtable holds <int (item handle), string (item ID)> pairs. We used another hashtable called "**HDATagList**" which handles the < string (item ID), **HDA_ITEM** (HDA item structure)>.
The HDA_ITEM is a structure that contains, for a given item ID, a list of values, timestamps, and qualities. This list is updated every time an OPC DA item value changes.

**The hClient field in the HDA_ITEM structure does not have any relation with the 'hClient definition' used in OPC server/client transactions (such as ReadAtTime method).**

## 5.2. Classes

**OPCServer**

This class implements the OPC DA/HDA callbacks and includes the functions for building the server address space, and setting the HDA configuration.
The following are the main functions:

1. CreateAddressSpace(): creates a local list that holds all available OPC Data Access tags. It calls io_createtag to build the server address space in the OPC Server Toolkit.
2. SetHdaConfiguration: used for HDA initialization. It calls io_hdasethdaitems( ), io_hdasetitemaggregates( ) and io_hdasetupdatemethods( ).
3. SetHdaConfiguration uses the following local functions:
   a. GetTagAttributes: retrieves HDA attributes such as IO_OPCHDA_DATA_TYPE, IO_OPCHDA_ITEMID, etc.
   b. GetTags: retrieves the items for which we want to store historical values.
   c. GetSimAggregates: retrieves HDA aggregates such as IO_OPCHDA_INTERPOLATIVE, IO_OPCHDA_AVERAGE, etc.

**Add_Tag**
This class is a Windows form that is responsible for setting the tag properties before being added to the address space.

**OPCServerSimulation**
This class is a Windows form that displays both the address space treeview in the left side and server statistics (groups' number, server status, items number, etc.) in the right side. It also manages adding/removing tags to/from the address space hashtable that holds <string (full tag name), TAGITEM structure> pairs.

# 6. The C# WPF Application Architecture

In this section, we describe the main implemented classes in the C# WPF OPC server sample.

## 6.1. Main Lists

In this sample, we used a hashtable "**AddressSpace**" which represents a collection of <string (full tag name), TAGITEM structure> pairs that are organized based on the hash code of the key. The Address Space holds all simulated OPC DA tags.

For historical data simulation, we also used a hashtable "**TagNameMappedByHandle**". This hashtable holds <int (item handle), string (item ID)> pairs. We used another hashtable called "**HDATagList**" which handles the < string (item ID), **HDA_ITEM** (HDA item structure)>.
The HDA_ITEM is a structure that contains, for a given item ID, a list of values, timestamps, and qualities. This list is updated every time an OPC DA item value changes.

**The hClient field in the HDA_ITEM structure does not have any relation with the 'hClient definition' used in OPC server/client transactions (such as ReadAtTime method).**

## 6.2. Classes

**OPCServer**
This class implements the OPC DA/HDA callbacks and includes the functions for building the server address space, and setting the HDA configuration.
The following are the main functions:

4. CreateAddressSpace(): creates a local list that holds all available OPC Data Access tags. It calls io_createtag to build the server address space in the OPC Server Toolkit.
5. SetHdaConfiguration: used for HDA initialization. It calls io_hdasethdaitems( ), io_hdasetitemaggregates( ) and io_hdasetupdatemethods( ).
6. SetHdaConfiguration uses the following local functions:
    d. GetTagAttributes: retrieves HDA attributes such as IO_OPCHDA_DATA_TYPE, IO_OPCHDA_ITEMID, etc.
    e. GetTags: retrieves the items for which we want to store historical values.

f. GetSimAggregates: retrieves HDA aggregates such as IO_OPCHDA_INTERPOLATIVE, IO_OPCHDA_AVERAGE, etc.

**Add_Tag**

This class is a Windows form that is responsible for setting the tag properties before being added to the address space.

**MainWindow**

This class is a Windows form that displays both the address space treeview in the left side and server statistics (groups' number, server status, items number, etc.) in the right side. It also initializes the OPC Server using a windows thread and manages adding/removing tags to/from the address space hashtable that holds <string (full tag name), TAGITEM structure> pairs.

# 7. Executing the VS 2008 C++ Service Sample

## 7.1. Service Installation

You can install the service manually by opening the command prompt window as administrator and typing:

*Prompt> OPCServerSimulatorVS2008Service.exe -install.*

*Prompt* is the path of the target directory where the "OPCServerSimulatorVS2008Service" executable is located.

## 7.2. Service Uninstallation

You can uninstall the service manually by opening the command prompt window as administrator and typing:

*Prompt> OPCServerSimulatorVS2008Service.exe -remove .*

*Prompt* is the path of the target directory where the "OPCServerSimulatorVS2008Service" executable is located.

## 7.3. Service Log on

In case the local connection to the OPC Server Service failed due to an access deny error, you need to follow the steps below to change the service log on to start with an administrator account:

1. Open the Windows service manager
2. Select the "Integration Objects' OPC Server Toolkit C++ Simulator"
3. Right click then select properties.
4. Select the Log On tab.
5. Check the "This account" option
6. Enter your administrator account credentials as illustrated in the following figure:

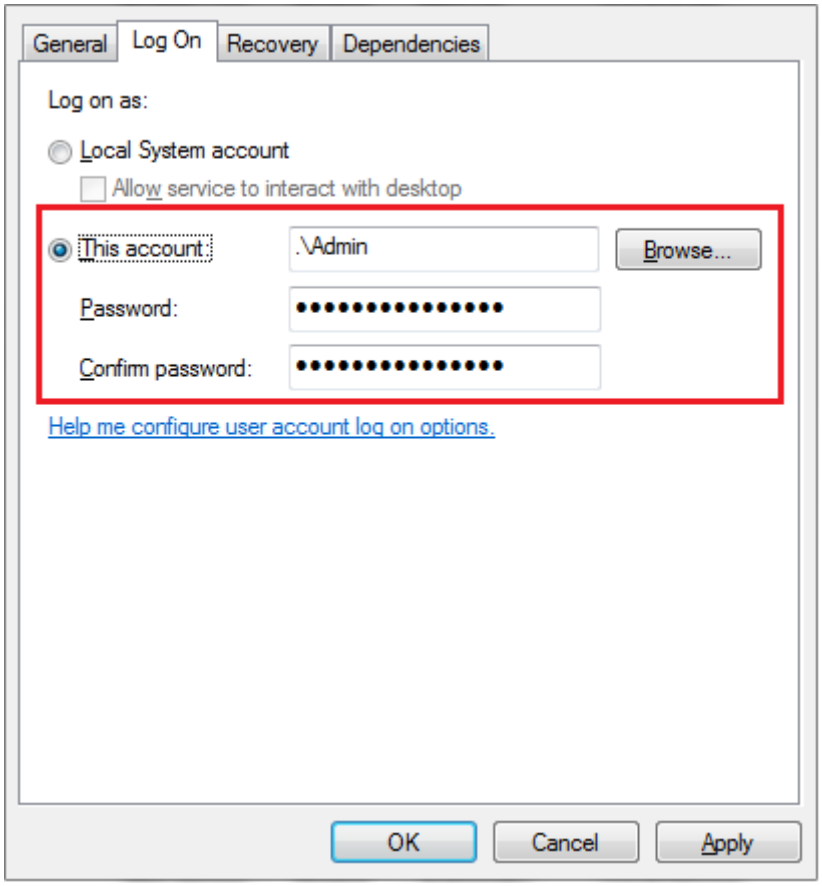


**Figure 32: OPC Server C++ Sample Service Administrator Log On**

# 8. Executing the VS 2012 C++ Service Sample

## 8.1. Service Installation

You can install the service manually by opening the command prompt window as administrator and typing:

*Prompt> OPCServerSimulatorVS2012Service.exe -install.*

*Prompt* is the path of the target directory where the "OPCServerSimulatorVS2012Service" executable is located.

## 8.2. Service Uninstallation

You can uninstall the service manually by opening the command prompt window as administrator and typing:

*Prompt> OPCServerSimulatorVS2012Service.exe -remove .*

*Prompt* is the path of the target directory where the "OPCServerSimulatorVS2012Service" executable is located.

## 8.3. Service Log on

In case the local connection to the OPC Server Service failed due to an access deny error, you need to follow the steps below to change the service log on to start with an administrator account:

7. Open the Windows service manager
8. Select the "Integration Objects' OPC Server Toolkit VS 2012 C++ Simulator"
9. Right click then select properties.
10. Select the Log On tab.
11. Check the "This account" option
12. Enter your administrator account credentials as illustrated in the following figure:

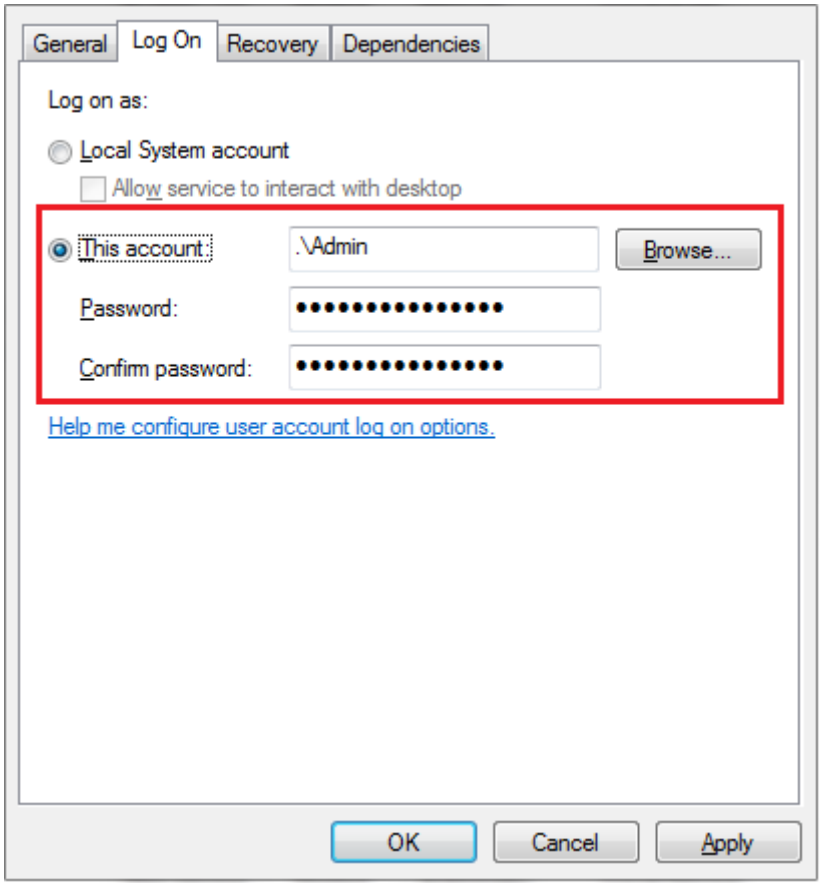


**Figure 33: OPC Server C++ Sample Service Administrator Log On**

## 8.4. Application build

### 8.4.1.X64 build mode

In order to successfully build and run the OPC Server Simulator Service in 64 bits mode, the following steps should be followed:

- Copy the 64 bits DXServerDLL.lib from the installation folder ".:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\lib\x64" folder to the project folder and add it to the *Additional Dependencies*

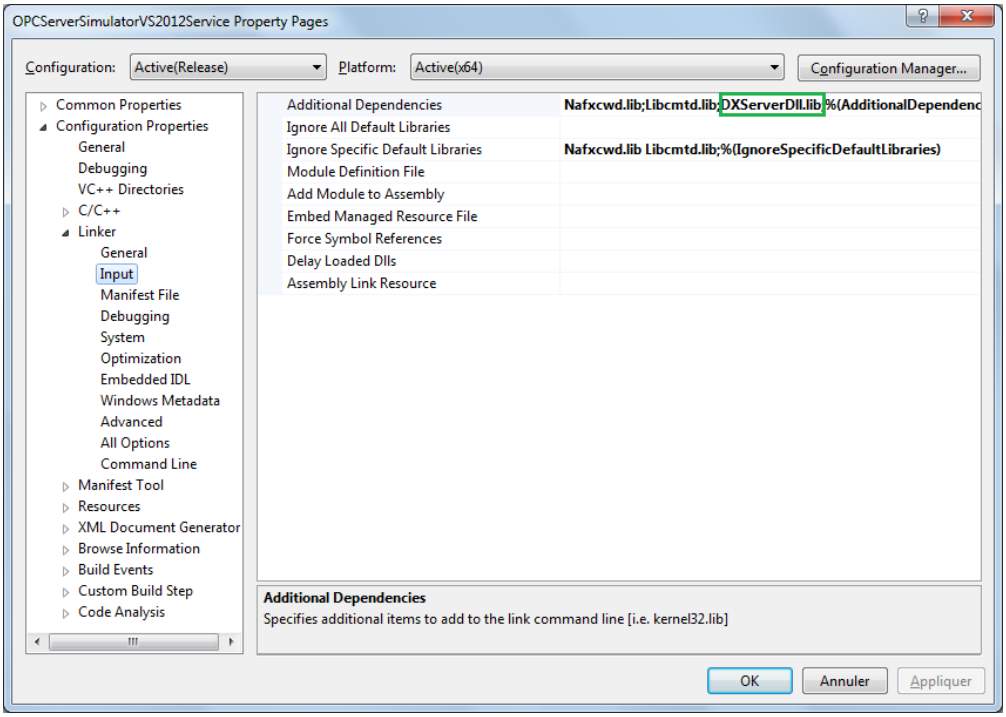


**Figure 34: DXServerDll.lib Additional Dependencies**

- Copy the following files from the installation folder " .:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\DLLs\x64" to the output project folder:
  - DXServerDll.dll
  - License.dll
  - OPCDADLL.dll

## 8.4.2.Win32 build mode

In order to successfully build and run the OPC Server Simulator Service in 32 bits mode, the following steps should be followed:

- Copy the 64 bits DXServerDLL.lib from the installation folder ".:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\lib\x86" folder to the project folder and add it to the *Additional Dependencies*

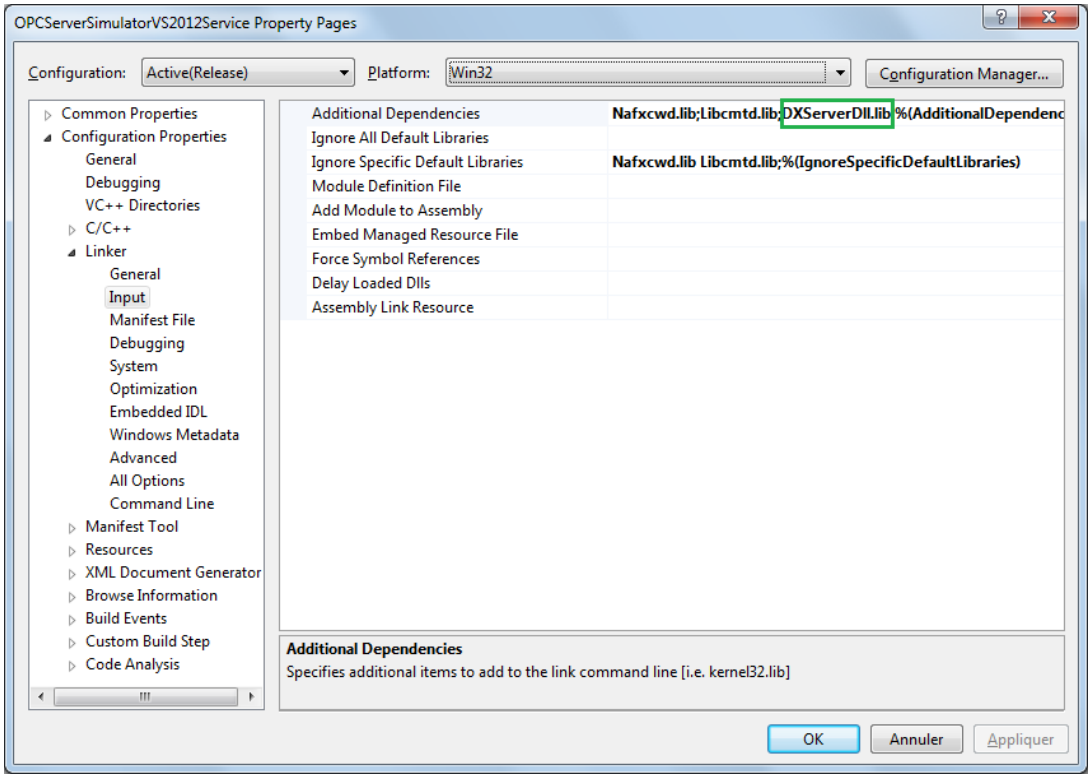


**Figure 35: DXServerDll.lib Additional Dependencies**

- Copy the following files from the installation folder " .:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\DLLs\x86**"** to the output project folder:
  - DXServerDll.dll
  - License.dll
  - OPCDADLL.dll

# 9. The VS 2012 C++ Service Sample Architecture

## 9.1. Files Description

**ServiceInstaller**

This class implements the functions that install and uninstall the service which are:

1. InstallService(PWSTR pszServiceName,
   PWSTR pszDisplayName,
   DWORD dwStartType,
   PWSTR pszDependencies,
   PWSTR pszAccount,
   PWSTR pszPassword)

This method installs the OPC Server Simulator as a Windows service.

2. UninstallService(PWSTR pszServiceName)

This method stops and removes the service from the Windows services list.

**CServiceBase**
This class provides a base class for the OPC server service and manages the service start and stop actions.

**CppWindowsService**
The file defines *wmain* function which is the entry point of the OPC server service application. According to the arguments in the command line, the function installs or uninstalls the service by calling into different routines.

**CSampleService**
This class implements the OPC server initialization and the address space creation. It also implements the OPC DA and HDA callbacks activation.

**CItemtree**
This class manages a map list that holds <string (full tag name), TAGITEM structure> pairs. We can add/update a node in the list. We can also search for an item giving its full name. The following are the main methods:

1. Insert: Adds a node to the map.
2. FillInDAItem: Updates a node in the map.
3. FindItem: Searches in the list for the specified key (full tag name).

# 10. Executing the VS2010 C# .NET Service Sample

## 10.1. Service Installation

You can install the service by editing the "InstallService.bat" file as following:

- Change the "*C:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\exe\ OPCServerSimulationCS2010Service.exe*" to the path of the "*OPCServerSimulationCS2010Service*" executable in your machine:
- Save the file then run it as administrator to install the service

## 10.2. Service Uninstallation

You can uninstall the service by editing the "UninstallService.bat" file as following:

- Change the "*C:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\exe\ OPCServerSimulationCS2010Service.exe*" to the path of the "*OPCServerSimulationCS2010Service*" executable in your machine:
- Save the file then run it as administrator to uninstall the service

## 10.3. Service Log On

In case the local connection to the OPC Server Service failed due to an access deny you need to follow the steps below to change the service log on to start with an administrator account:

1. Open the windows service manager
2. Select the Integration Objects' OPC Server Toolkit C# Simulator Service.
3. Right click then select properties.
4. Select the Log On tab.
5. Check the "This account" radio button.
6. Enter your administrator account credentials as shown in the following figure:

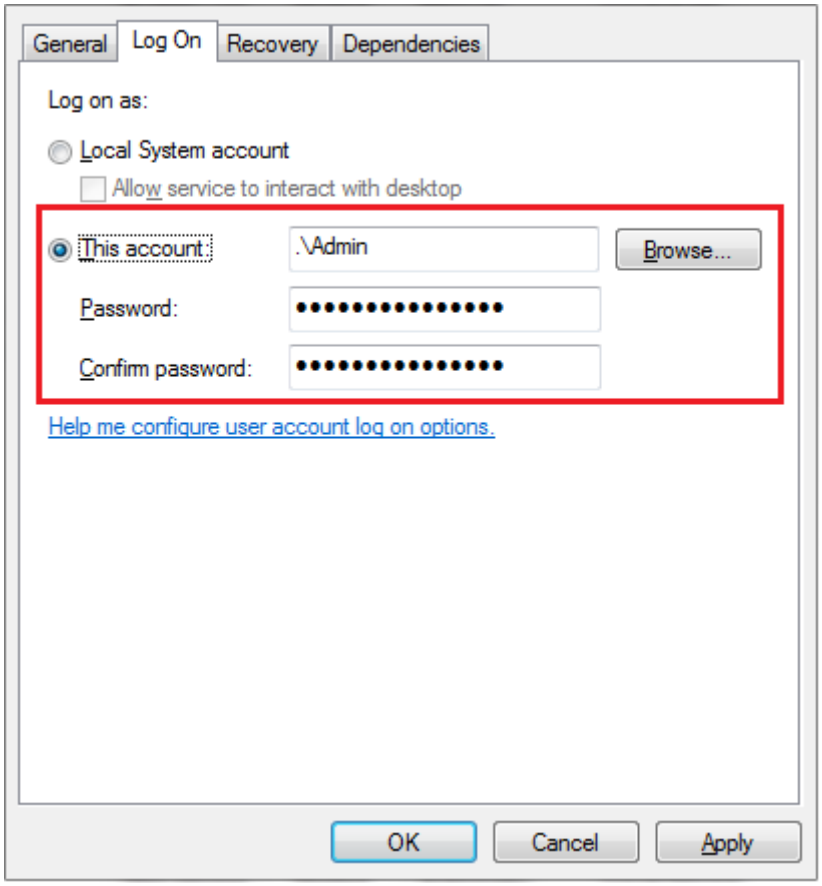


**Figure 36: OPC Server C# Sample Service Administrator Log On**

# 11. The VS2010 C# .NET Service Sample Architecture

## 11.1. Classes

**WindowsService**

This class include the OPC Server Service parameters initialization. It implements the StartService() thread method.

**OPCServer**

This class implements the OPC DA/HDA callbacks and includes the functions for building the server address space, and setting the HDA configuration.

# 12. Executing the VS2012 C# .NET Service Sample

## 12.1. Service Installation

You can install the service by editing the "InstallService.bat" file as following:

- Change the "*C:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\exe\ OPCServerSimulationCS2010Service.exe*" to the path of the "*OPCServerSimulationCS2010Service*" executable in your machine:
- Save the file then run it as administrator to install the service

## 12.2. Service Uninstallation

You can uninstall the service by editing the "UninstallService.bat" file as following:

- Change the "*C:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\exe\ OPCServerSimulationCS2010Service.exe*" to the path of the "*OPCServerSimulationCS2010Service*" executable in your machine:
- Save the file then run it as administrator to uninstall the service

## 12.3. Service Log On

In case the local connection to the OPC Server Service failed due to an access deny you need to follow the steps below to change the service log on to start with an administrator account:

7. Open the windows service manager
8. Select the Integration Objects' OPC Server Toolkit C# Simulator Service.
9. Right click then select properties.
10. Select the Log On tab.
11. Check the "This account" radio button.
12. Enter your administrator account credentials as shown in the following figure:

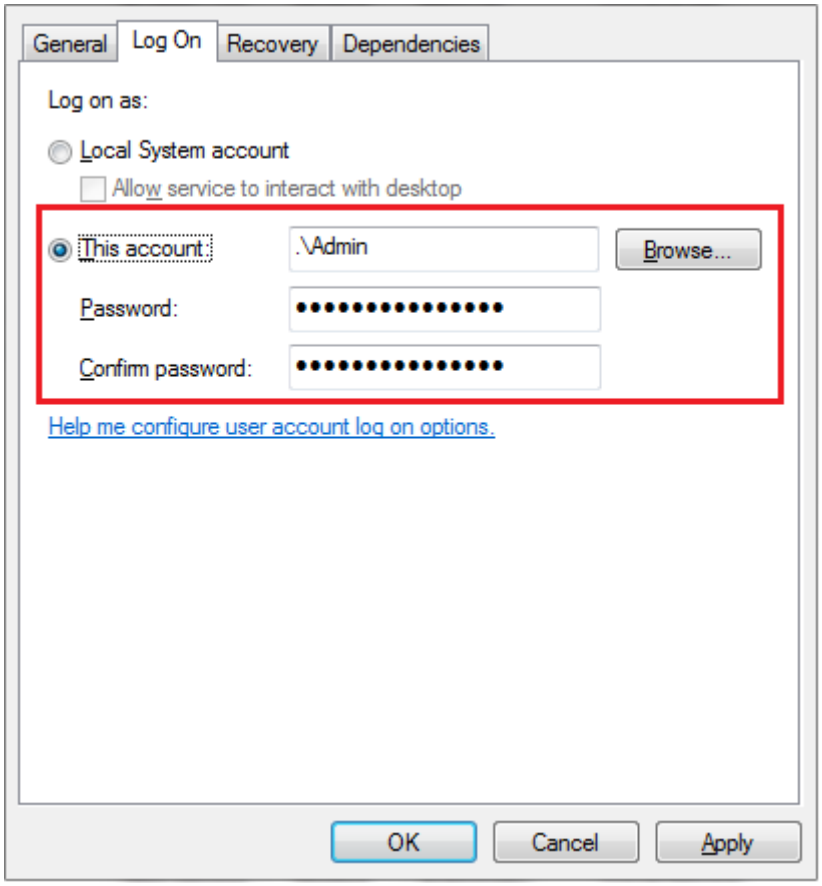


**Figure 37: OPC Server C# Sample Service Administrator Log On**

## 12.4. Application build

### 12.4.1. X64 build mode

In order to successfully build and run the OPC Server Simulator Service in 64 bits mode, the following steps should be followed:

- Add a reference to the OPCNetServerSDK that you can find in the installation folder ".:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\DLLs"

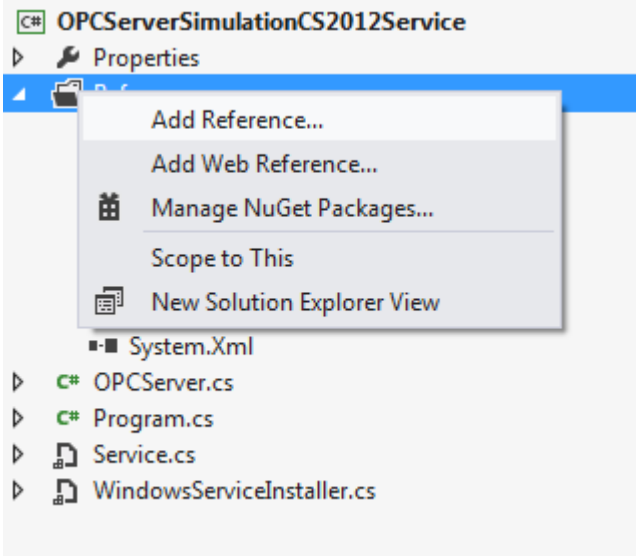


**Figure 38: Add Reference to the OPCNetServerSDK**

- Copy the following files from the installation folder " .:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\DLLs\x64**"** to the output project folder:
  - DXServerDll.dll
  - License.dll
  - OPCDADLL.dll
- Copy the OPCNetServerSDK.dll file from the installation folder " .:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\DLLs" to the output project folder:

### 12.4.2. Win32 build mode

In order to successfully build and run the OPC Server Simulator Service in 32 bits mode, the following steps should be followed:

- Add a reference to the OPCNetServerSDK that you can find in the installation folder ".:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\DLLs"

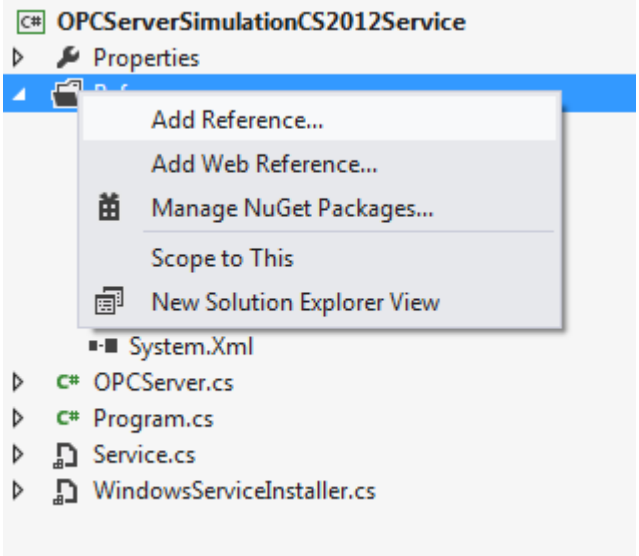


**Figure 39: Add Reference to the OPCNetServerSDK**

- Copy the following files from the installation folder " .:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\DLLs\x86" to the output project folder:
  - DXServerDll.dll
  - License.dll
  - OPCDADLL.dll
- Copy the OPCNetServerSDK.dll file from the installation folder " .:\Program Files (x86)\Integration Objects\Integration Objects' OPC Server Toolkit\DLLs" to the output project folder:

# 13. The VS2012 C# .NET Service Sample Architecture

## 13.1. Classes

**WindowsService**

This class include the OPC Server Service parameters initialization. It implements the StartService() thread method.

**OPCServer**

This class implements the OPC DA/HDA callbacks and includes the functions for building the server address space, and setting the HDA configuration.

# APPENDIX A

## 1. Item Properties

The following sections are prepared based on OPC DA specifications version 2.0 and 3.0.

### 1.1. OPC Property Sets

This is a set of property IDs that are common to many servers. In addition to those listed in the Supported Item properties section, property IDs from 9 to 99 are reserved by OPC Foundation for future OPC use.

### 1.2. Recommended Properties

This is additional information which is commonly associated with Items. This includes additional ranges of values that are reserved for use by other future OPC specifications. **A server can provide any subset of these values (or none of them).**

| ID | DATATYPE of returned VARIANT | STANDARD DESCIPTION |
|---|---|---|
| | | **Properties related to the Item Value** |
| 100 | VT_BSTR | "EU Units"<br>e.g. "DEGC" or "GALLONS" |
| 101 | VT_BSTR | "Item Description"<br>e.g. "Evaporator 6 Coolant Temp" |
| 102 | VT_R8 | "High EU"<br>Present only for 'analog' data. This represents the highest value likely to be obtained in normal operation and is intended for such use as automatically scaling a bar graph display.<br>e.g. 1400.0 |
| 103 | VT_R8 | "Low EU"<br>Present only for 'analog' data. This represents the lowest value likely to be obtained in normal operation and is intended for such use as automatically scaling a bar graph display.<br>e.g. -200.0 |

| | | |
|---|---|---|
| 104 | VT_R8 | "High Instrument Range"<br>Present only for 'analog' data. This represents the highest value that can be returned by the instrument.<br>e.g. 9999.9 |
| 105 | VT_R8 | "Low Instrument Range"<br>Present only for 'analog' data. This represents the lowest value that can be returned by the instrument.<br>e.g. -9999.9 |
| 106 | VT_BSTR | "Contact Close Label"<br>Present only for 'discrete' data. This represents a string to be associated with this contact when it is in the closed (non-zero) state.<br>e.g. "RUN", "CLOSE", "ENABLE", "SAFE" ,etc. |
| 107 | VT_BSTR | "Contact Open Label"<br>Present only for 'discrete' data. This represents a string to be associated with this contact when it is in the open (zero) state.<br>e.g. "STOP", "OPEN", "DISABLE", "UNSAFE" ,etc. |
| 108 | VT_I4 | "Item Timezone" The difference in minutes between the item's UTC Timestamp and the local time in which the item value was obtained. See the OPCGroup TimeBias property. Also see the WIN32 TIME_ZONE_INFORMATION structure. |
| 109-199 | | Reserved for future OPC use. Additional IDs may be added without revising the interface ID. |
| | | |
| | | **Properties related to operator displays** |
| 200 | VT_BSTR | "Default Display"<br>The name of an operator display associated with this ItemID. |

| | | |
|---|---|---|
| 201 | VT_I4 | "Current Foreground Color"<br>The COLORREF in which the item should be displayed. |
| 202 | VT_I4 | "Current Background Color"<br>The COLORREF in which the item should be displayed. |
| 203 | VT_BOOL | "Current Blink"<br>Should a display of this item blink? |
| 204 | VT_BSTR | "BMP File"<br>e.g. C:\MEDIA\FIC101.BMP |
| 205 | VT_BSTR | "Sound File"<br>e.g. C:\MEDIA\FIC101.WAV, or .MID |
| 206 | VT_BSTR | "HTML File"<br>e.g. http:\\mypage.com/FIC101.HML |
| 207 | VT_BSTR | "AVI File"<br>e.g. C:\MEDIA\FIC101.AVI |
| 208-299 | | Reserved for future OPC use. Additional IDs may be added without revising the interface ID. |
| | | |
| | | **Properties Related to Alarm and Condition Values (preliminary), etc.**<br>**IDs 300 to 399 are reserved for use by OPC Alarms and Events.**<br>**See the OPC Alarm and Events specification for additional information.** |
| 300 | VT_BSTR | "Condition Status"<br>The current alarm or condition status associated with the Item.<br>e.g. "NORMAL", "ACTIVE", "HI ALARM", etc. |
| 301 | VT_BSTR | "Alarm Quick Help"<br>A short text string providing a brief set of instructions for the operator to follow when this alarm occurs. |

| | | |
|---|---|---|
| 302 | VT_BSTR<br>\|VT_ARRAY | "Alarm Area List"<br>An array of stings indicating the plant or alarm areas which include this ItemID. |
| 303 | VT_BSTR | "Primary Alarm Area"<br>A string indicating the primary plant or alarm area including this ItemID |
| 304 | VT_BSTR | "Condition Logic"<br>An arbitrary string describing the test being performed.<br>e.g. "High Limit Exceeded" or "TAG.PV >= TAG.HILIM" |
| 305 | VT_BSTR | "Limit Exceeded"<br>For multistate alarms, the condition exceeded.<br>e.g. HIHI, HI, LO, LOLO |
| 306 | VT_R8 | "Deadband" |
| 307 | VT_R8 | "HiHi Limit" |
| 308 | VT_R8 | "Hi Limit" |
| 309 | VT_R8 | "Lo Limit" |
| 310 | VT_R8 | "LoLo Limit" |
| 311 | VT_R8 | "Rate of Change Limit" |
| 312 | VT_R8 | "Deviation Limit" |
| 313-399 | | Reserved for future OPC Alarms and Events use. Additional IDs may be added without revising the interface ID. |
| | | |
| 400-4999 | | Reserved for future OPC use. Additional IDs may be added without revising the interface ID. |

**Table 95: Recommended Properties**

# APPENDIX B

This appendix is reserved for constants defined in the header file provided with this OPC Server Toolkit. Descriptions are taken from the OPC specifications published by the OPC Foundation.

## 1. Data Access

### 1.1. Item Qualities

This section presents all OPC DA qualities that you can use for items' qualities.

| UshoQuality | Value |
|---|---|
| IO_OPC_USHOQUALITY_BAD | 0 |
| IO_OPC_USHOQUALITY_UNCERTAIN | 0x40 |
| IO_OPC_USHOQUALITY_GOOD | 0xc0 |
| IO_OPC_USHOQUALITY_CONFIG_ERROR | 0x4 |
| IO_OPC_USHOQUALITY_NOT_CONNECTED | 0x8 |
| IO_OPC_USHOQUALITY_DEVICE_FAILURE | 0xc |
| IO_OPC_USHOQUALITY_SENSOR_FAILURE | 0x10 |
| IO_OPC_USHOQUALITY_LAST_KNOWN | 0x14 |
| IO_OPC_USHOQUALITY_COMM_FAILURE | 0x18 |
| IO_OPC_USHOQUALITY_OUT_OF_SERVICE | 0x1c |
| IO_OPC_USHOQUALITY_WAITING_FOR_INITIAL_DATA | 0x20 |
| IO_OPC_USHOQUALITY_LAST_USABLE | 0x44 |
| IO_OPC_USHOQUALITY_SENSOR_CAL | 0x50 |
| IO_OPC_USHOQUALITY_EGU_EXCEEDED | 0x54 |
| IO_OPC_USHOQUALITY_SUB_NORMAL | 0x58 |
| IO_OPC_USHOQUALITY_LOCAL_OVERRIDE | 0xd8 |

| | |
|---|---|
| IO_OPC_LIMIT_OK | 0 |
| IO_OPC_LIMIT_LOW | 0x1 |
| IO_OPC_LIMIT_HIGH | 0x2 |
| IO_OPC_LIMIT_CONST | 0x3 |

**Table 96: Item Qualities**

# 2. Historical Data Access

## 2.1. Attributes

The following table contains all of the defined OPC attributes (with reserved values):

| AttrID | Description | Data Type | Value |
|---|---|---|---|
| IO_OPCHDA_DATA_TYPE | Specifies the data type for the item. See the definition of a VARIANT for valid values (VT_R4, etc.). | VT_I2 | 0x01 |
| IO_OPCHDA_DESCRIPTION | Describes the item. | VT_BSTR | 0x02 |
| IO_OPCHDA_ENG_UNITS | Specifies the label to use in displays to define the units for the item (e.g., kg/sec). | VT_BSTR | 0x03 |
| IO_OPCHDA_STEPPED | Specifies whether data from the history repository should be displayed as interpolated (sloped lines between points) or stepped (vertically-connected horizontal lines between points) data. Value of 0 indicates interpolated. | VT_BOOL | 0x04 |
| IO_OPCHDA_ARCHIVING | Indicates whether the historian is recording data for this item (0 means no). | VT_BOOL | 0x05 |
| IO_OPCHDA_DERIVE_EQUATION | Specifies the equation to be used by a derived item to calculate its value. This is a free-form string. | VT_BSTR | 0x06 |

| | | | |
|---|---|---|---|
| IO_OPCHDA_NODE_NAME | Specifies the machine which is the source for the item. This is intended to be the broadest category for defining sources. For an OPC Data Access Server source, this is the node name or IP address of the server. For non-OPC sources, the meaning of this field is server-specific. | VT_BSTR | 0x07 |
| IO_OPCHDA_PROCESS_NAME | Specifies the process which is the source for the item. This is intended to the second-broadest category for defining sources. For an OPC DA server, this would be the registered server name. For non-OPC sources, the meaning of this field is server-specific. | VT_BSTR | 0x08 |
| IO_OPCHDA_SOURCE_NAME | Specifies the name of the item on the source. For an OPC DA server, this is the ItemID. For non-OPC sources, the meaning of this field is server-specific. | VT_BSTR | 0x09 |
| IO_OPCHDA_SOURCE_TYPE | Specifies what sort of source produces the data for the item. For an OPC DA server, this would be "OPC". For non-OPC sources, the meaning of this field is server-specific. | VT_BSTR | 0x0a |
| IO_OPCHDA_NORMAL_MAXIMUM | Specifies the upper limit for the normal value range for the item. OPCHDA_NORMAL_MAXIMUM is used for trend display default scaling and exception deviation limit calculations.<br>OPCHDA_ NORMAL _MAXIMUM should be the normal high value for the item. | VT_R8 | 0x0b |

| | | | |
|---|---|---|---|
| IO_OPCHDA_NORMAL_MINIMUM | Specifies the lower limit for the normal value range for the item. OPCHDA_ NORMAL _MINIMUM is used for trend display default scaling and exception deviation limit calculations.<br>OPCHDA_ NORMAL _MINIMUM should be the normal low value for the item. | VT_R8 | 0x0c |
| IO_OPCHDA_ITEMID | Specifies the Item ID. This is used to allow filtering in the CreateBrowse method. | VT_BSTR | 0x0d |

**Table 97: Attributes**

You may need to define your own attributes. Your specific attributes must be defined with values beginning at **0x80000000**.

## 2.2. Aggregates

The following table contains all of the defined OPC aggregates (with reserved values):

| Aggregate Value | Description |
|---|---|
| IO_OPCHDA_INTERPOLATIVE | Does not retrieve an aggregate. This is used for retrieving interpolated values. |
| IO_OPCHDA_TOTAL | Retrieves the totalized value (time integral) of the data over the resample interval. |
| IO_OPCHDA_AVERAGE | Retrieves the average data over the resample interval. |
| IO_OPCHDA_TIMEAVERAGE | Retrieves the time weighted average data over the resample interval. |
| IO_OPCHDA_COUNT | Retrieves the number of raw values over the resample interval. |
| IO_OPCHDA_STDEV | Retrieves the standard deviation over the resample interval. |
| IO_OPCHDA_MINIMUMACTUALTIME | Retrieves the minimum value in the resample interval and the timestamp of the minimum value. |

| | |
|---|---|
| IO_OPCHDA_MINIMUM | Retrieves the minimum value in the resample interval. |
| IO_OPCHDA_MAXIMUMACTUALTIME | Retrieves the maximum value in the resample interval and the timestamp of the maximum value. |
| IO_OPCHDA_MAXIMUM | Retrieves the maximum value in the resample interval. |
| IO_OPCHDA_START | Retrieves the value at the beginning of the resample interval. The time stamp is the time stamp of the beginning of the interval. |
| IO_OPCHDA_END | Retrieves the value at the end of the resample interval. The time stamp is the time stamp of the end of the interval. |
| IO_OPCHDA_DELTA | Retrieves the difference between the first and last value in the resample interval. |
| IO_OPCHDA_REGSLOPE | Retrieve the slope of the regression line over the resample interval. |
| IO_OPCHDA_REGCONST | Retrieves the intercept of the regression line over the resample interval. This is the value of the regression line at the start of the interval. |
| IO_OPCHDA_REGDEV | Retrieves the standard deviation of the regression line over the resample interval. |
| IO_OPCHDA_VARIANCE | Retrieves the variance over the sample interval. |
| IO_OPCHDA_RANGE | Retrieves the difference between the minimum and maximum value over the sample interval. |
| IO_OPCHDA_DURATIONGOOD | Retrieves the duration (in seconds) of time in the interval during which the data is good. |
| IO_OPCHDA_DURATIONBAD | Retrieves the duration (in seconds) of time in the interval during which the data is bad. |

| | |
|---|---|
| IO_OPCHDA_PERCENTGOOD | Retrieves the percent of data (1 equals 100 percent) in the interval which has good ushoQuality. |
| IO_OPCHDA_PERCENTBAD | Retrieves the percent of data (1 equals 100 percent) in the interval which has bad ushoQuality. |
| IO_OPCHDA_WORSTUSHOQUALITY | Retrieves the worst ushoQuality of data in the interval. |
| IO_OPCHDA_ANNOTATIONS | Retrieves the number of annotations in the interval. |

**Table 98: Aggregates**

You may need to define your own aggregates. Your specific aggregates must be defined with values beginning at **0x80000000**.

## 2.3. Capabilities

This section describes all available capabilities methods offered by the OPC HDA specifications.

| Capability | Description | Value |
|---|---|---|
| IO_OPCHDA_INSERTCAP | Supporting value's insertion ➔ Add new raw data if no data is found at a given timestamp. | 0x1 |
| IO_OPCHDA_REPLACECAP | Supporting value's replacement. | 0x2 |
| IO_OPCHDA_INSERTREPLACECAP | If no data is found at a given timestamp, it will be inserted in the history. Otherwise, the old value is replaced by the new one. | 0x4 |
| IO_OPCHDA_DELETERAWCAP | Supporting raw deletion for a specified time domain. | 0x8 |
| IO_OPCHDA_DELETEATTIMECAP | Supporting history data deletion at a given timestamp. | 0x10 |

**Table 99: Capabilities**

# 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**

**C**omponent **O**bject **M**odel is a specification for writing reusable software components. COM is an infrastructure that allows objects to communicate between processes and computers.

## D

**DCOM**

**D**istributed **C**omponent **O**bject **M**odel is a protocol that enables software components to communicate directly over a network in a reliable, secure and efficient manner.

## O

**OPC**

**O**LE for **P**rocess **C**ontrol 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.

## S

**SDK**
Software Development Kit.

## X

**XML**
eXtensible Markup Language (XML) is a simple, very flexible text format. Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

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