Integration Objects'

Solution for OPC A&E Servers Development using C++

OPC A&E Server SDK

Version 2.0 Rev.1

USER GUIDE

Compatibility

OPC A&E 1.02
OPC A&E 1.10

Integration Objects' OPC AE Server SDK User's Guide for C++ Version 2.0 Rev.1
Published September 2013

Copyright © 2004 - 2013 Integration Objects. All rights reserved.

All rights reserved. No part of this document may be reproduced, stored in a retrieval system, translated, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Integration Objects.

Windows®, Windows NT® and .NET are registered trademarks of Microsoft Corporation.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

PREFACE

About this User Guide

This guide:

- Presents the Integration Objects' OPC A&E Server SDK (Software Developers Kit).
- Explains how to use this toolkit to generate out-of-process server specific applications.
- Describes in details the functions exported by its API (Application Programmer Interface) and explains the returned error codes.

Target Audience

This guide is intended for developers of OPC (OLE for Process Control) Alarms and Events compliant servers. It assumes that you have a working knowledge of programming with the Visual C++ language. It also assumes that you have an idea about the OPC Alarms and Events specifications.

Related Documents

As you use this user guide, you may also find useful the following specifications:

- OPC Common Definitions and Interfaces 1.0
- OPC Alarms and Events Standards 1.02
- OPC Alarms and Events Standards 1.10

Document Conventions

Convention	Description
Bold	Click/selection action required
	Information to be noted
Blue bold italics	Reference to other sections, or to other Integration Objects User Guides

Customer Support Services

Offices	Email
Houston, USA: +1 713 609 9208	Support: customerservice@integrationobjects.com
Genova, Italy: +39 34 75 83 93 47	Sales: sales@integrationobjects.com
Tunis, Tunisia: +216 71 861 803	Online: www.integrationobjects.com

OPC AE SERVER SDK OVERVIEW

1. Overview

The Integration Objects' OPC AE Server SDK is supplied in the form of a Windows DLL (Dynamic Link Library) supporting standard Windows API calling conventions. This DLL implements all required and most of the optional OPC Alarms and Events interfaces (refer to the list of supported interfaces below) and handles all COM and OPC details necessary to interface with OPC AE clients. It is a tool for fast and easy creation of AE servers.

This API is compatible with the Microsoft Visual Basic programming Environment. For more information, you can see the *Reference Manual for VB*.

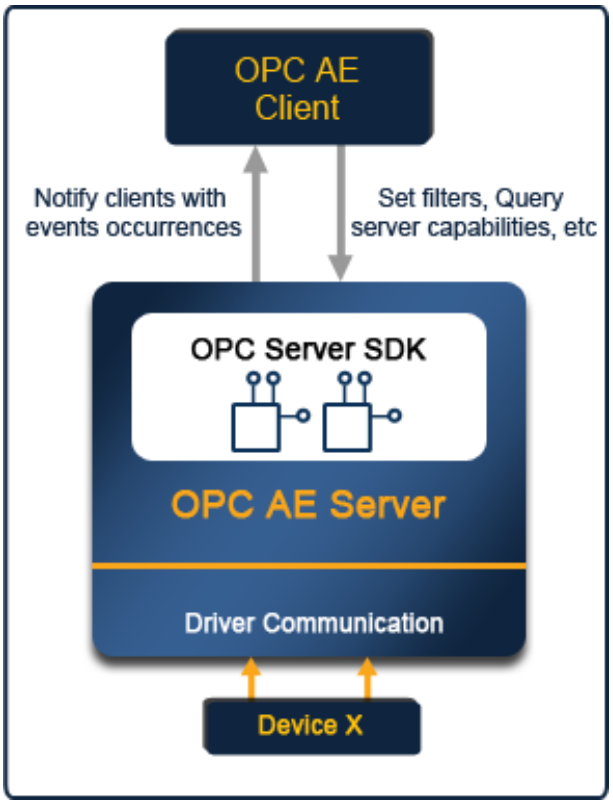


Figure 1: Deployment of the OPC AE Server SDK in a customized OPC server

Any OPC AE server offers mechanisms to notify OPC AE clients with events occurrences. It also offers the possibility for clients to set filter criteria for the alarms of their interest and to query server capabilities.

Any OPC AE server based on this SDK can be accessed locally or remotely via DCOM by any OPC AE client.

2. Features

All OPC AE servers share an important part of their code. This part covers COM/DCOM communications and the implementation of OPC AE interfaces. The aim of this SDK is to create a tool that hides the complexity of these treatments and provides an easy and flexible interface for developing server specific applications.

2.1. Registering/Unregistering the OPC server

Providing all 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 specified OPC server. So developers can use this SDK with minimal knowledge of COM/DCOM.

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

2.2. Alarms and Events Capabilities

Any OPC AE server made using the OPC AE Server SDK can communicate with OPC AE clients that are compliant with the OPC AE specification version 1.02 and 1.10. In fact, this toolkit implements all OPC Alarms and Events functionalities including the possibility to inform clients with events that occur in the system, to set filters for the returned alarms, and to query the server configuration.

The module that manages the server configuration is encapsulated into the DLL. It includes the building of the AreaSpace and the EventSpace and the initialization of the source cache.

Whenever the DLL needs to poll the equipment and/or to check the server internal actions when an important operation is taking place (such as the acknowledgement of a condition), it just invokes simple callback functions. These callbacks have been designed to handle the vendor internal actions and the communication with the real equipment.

We can describe the major functionalities offered by any OPC AE server made using this SDK as follows:

- Building the server AreaSpace.

- Building the server EventSpace.
- Initializing the server cache.
- Managing the OPC AE client connections.
- Managing the OPC Event subscriptions.
- Querying the server configuration.
- Firing all OPC event types.
- Enabling / disabling conditions.
- Activating / deactivating conditions.
- Acknowledging conditions.

During the server implementation, developers will just focus on device communication and specification.

3. OPCAE Server SDK Architecture

This SDK offers a set of functions that OPC server programs can use to treat OPC AE Client requests.

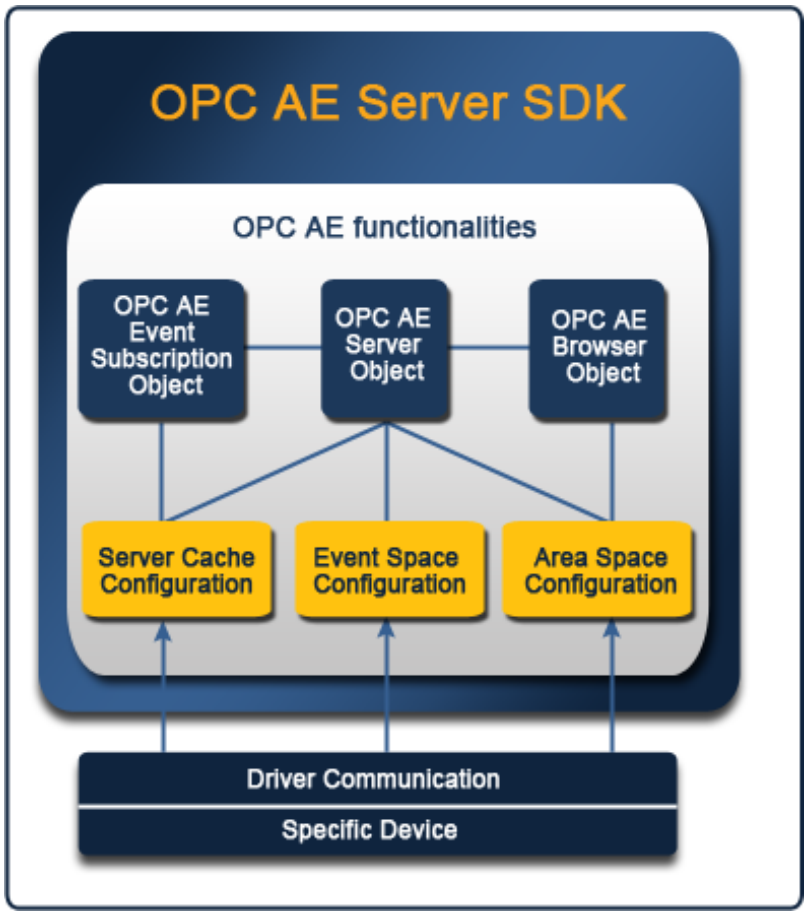


Figure 2: OPC AE Server SDK Architecture

It includes the following main modules:

- OPC AE Specification module implements all the AE specification functionalities and includes three objects:
  - OPC AE Server object is used to create OPC Event Subscription and OPC AE Event Area Browser objects, query vendor-specific event categories and event parameters, and manage conditions.
  - OPC AE Event Subscription object is used to configure filters and attributes for OPC event reporting.
  - OPC AE Browser object is used to provide a way for clients to browse the process area organization implemented by the server.
- Server Cache Configuration module is used to initialize the list of condition events already configured and being evaluated by the server.
- Event Space Configuration module: the OPC specification does not define categories, attributes and conditions. So, this module will be used to manage and customize the server event space.
- Area Space Configuration module is used to build the server area space (initialize the list of areas and sources to be supported by the server).

4. Supported OPC Interfaces

The OPC AE Server SDK is compliant with the latest released specifications for Alarms and Events (AE). It supports OPC AE 1.02 and 1.10.
We can recapitulate the supported interfaces in tables below:

- **OPC Event Server Object**
  The following table lists implemented OPC AE interfaces for the OPC Event Server object.

Interface	Required	Implemented
IUnknown	Yes	Yes
IOPCCommon	Yes	Yes
IOPCEventServer	Yes	Yes
IOPCEventServer2 (optional)	No	Yes
IOPCBrowseServerAddressSpace (optional)	No	Yes
IConnectionPointContainer	Yes	Yes

Table 1: implemented OPC AE interfaces for OPC Event Server object

Callbacks for events notification are handled by the *IConnectionPointContainer* and *IConnectionPoint* interfaces which are the standard DCOM interfaces for connectable objects. The *IConnectionPoint* interface is supported by the *OPCEventSink* object.

- **OPC Event Subscription Object**
  The following table lists the implemented OPC AE interfaces for the OPC Event Subscription object.

Interface	Required	Implemented
IUnknown	Yes	Yes
IOCEventSubscriptionMgt	Yes	Yes
IOCEventSubscriptionMgt2	No	Yes
IConnectionPointContainer	Yes	Yes

Table 2: implemented OPC AE interfaces for OPC Event Subscription object

  This object interacts with the *OPCEventSink* object.

- **OPC Event Area Browse Object**
  This is an optional object that is exposed by any OPC AE server based on this SDK. This object supports the *IOPCEventAreaBrowser* interface.

5. Operating system compatibility

This SDK runs under the following operating systems:

- Windows NT 4.0
- Windows XP
- Windows 2000
- Windows 2003
- Windows Server 2008
- Windows 7

GETTING STARTED

1. Pre-Installation Considerations

In order to run properly any OPC A&E server made using this SDK, you should install the following software components on the target system:

- The **OPC core components 2.00** that consist of all shared OPC modules including the DCOM proxy/stub libraries, the OPC Server Enumerator, .NET wrappers, etc.
  You can apply the OPC Core Components 2.00 Redistributable 1.06 delivered with the current package or download it from the OPC Foundation site.

2. Files Included In the Distribution

The following table shows the different folders and files created on your system after the installation:

Folder	Description
Include	**IOAEServerToolKit.h**: Includes all exported constants, structures and signatures of methods and functions. It represents the interaction between the server application and the toolkit. **Callbacks.bas**: Provides a Visual Basic implementation of the SDK callbacks. It can be used by VB developers to implement VB server application. **Wrapper.bas**: Provides the Visual Basic wrappers for the SDK exported functions. Useful for developing VB server application.
Configuration files	**IOAESimServer.csv**: used to demonstrate how the server can generate different event types **ConfigOPCAEServerSDK.ini**: contains configurable parameters for logging.

OPC AE Server Samples projects	This distribution contains: ▪ The source code of a basic C++ AE simulator made in MFC. ▪ The source code of a basic VB AE simulator. ▪ The source code of a basic .NET AE simulator.
EXE	**IOAESimServer.exe**: the C++ AE simulator executable. **IO.OPCAE.Simulator.1.exe**: the VB AE simulator executable. **IntegrationObjects.OPCAEServerVBNETSample**:the VB.NET AE simulator executable. **IntegrationObjects.OPCAEServerCSNETSample**:the CS.NET AE simulator executable.
OPC AE Explorer	This folder contains a free OPC A&E Client: the Integration Objects' OPC A&E Explorer. To start this client, double click on the AEExplorer.exe file.
Documentation	**OPC-AE Server SDK for .NET.pdf**: this User Guide. **OPC-AE Server SDK for C++.pdf:** User Guide of the C++ Wrapper of the OPC AE Server SDK. **OPC-AE Server SDK for VB.pdf:** User Guide of the VB Wrapper of the OPC AE Server SDK. And OPC Specification documents.
Dlls	This folder Contains: the OPC AE Server SDK(IOAESDK.dll) and License.dll

Table 3: Folders in the Distribution

3. Compiling and Linking Applications

To successfully create your server application, you have to:

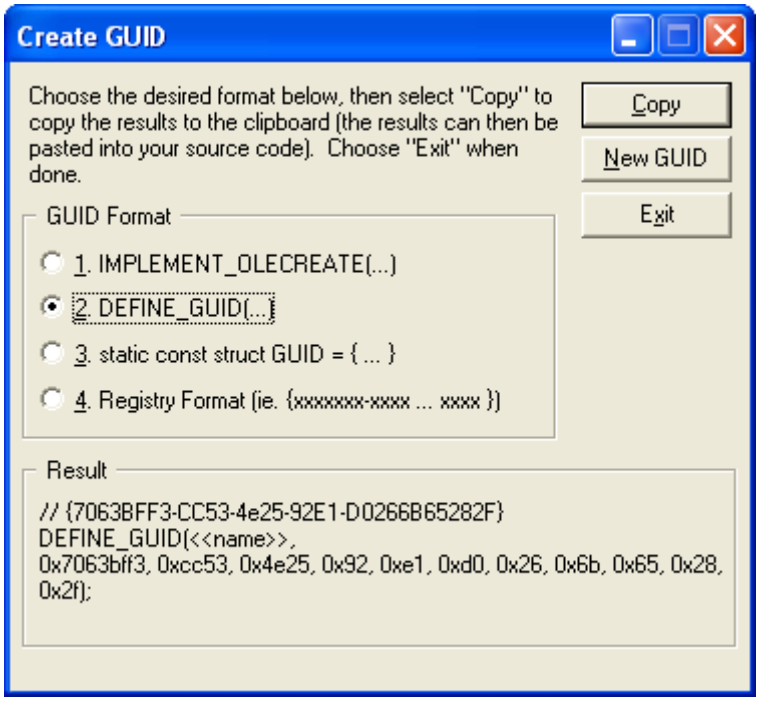
- **For C++ application:**
  1. Include the **IOAEServerToolKit.h** file within the project files for the custom C++ application.
  2. Add **IOAESDK.DLL** to your project folder.
- **For VB application:**
  1. Include **Callbacks.bas** and **Wrapper.bas** files within the project files for the custom VB application.
  2. Add **IOAEServerToolkit.dll** to your project folder.

4. Generate a new CLSID

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

You have to:

1. Launch *GUIDGen.exe* from [Your Drive]:\Program Files\Microsoft Visual Studio\Common\Tools\. A similar dialog screen will appear:



2. Generate a new CLSID using this utility by clicking the “New GUID” button.
3. Copy it to the clipboard (by clicking the “Copy” button) to be pasted into your server application source code.

Sample CLSID

// {785D6306-F436-4aac-A298-43393DC4AA9A}
DEFINE_GUID (CLSID_OPCAESampleServer, 0x785d6306, 0xf436, 0x4aac, 0xa2, 0x98, 0x43, 0x39, 0x3d, 0xc4, 0xaa, 0x9a);

EXPECTED INTERACTIONS WITH THE TOOLKIT

1. 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: Server Initialization routines, EventSpace Management, AreaSpace Management, Server Source Cache Management, and Event Notification Management.

This section doesn't include callback routines.

1.1. Server Initialization routines

1.1.1. IO_ComInitialize

HRESULT IO_ComInitialize**()**

Description

After the DLL initialization, the client application has the responsibility to properly initialize the COM library. The OPC Server SDK exports an API function named **IO_ComInitialize** that performs the basic initialization functions. **IO_ComInitialize** initializes COM to run in a multi-threaded environment.

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_COMCHANGMODE	A previous COM library initialization specified a different concurrency model for the calling thread.
IO_E_COMINVARG	An invalid argument was passed.
IO_E_COMOUTMEM	The memory could not be allocated.

IO_E_COMUNEXPECT	An unexpected error occurred.
IO_E_FAILED	An unknown error occurred.
IO_E_EXCEPTEDERROR	Exception error occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 4:IO_ComInitialize error codes

1.1.2. IO_Initialize

HRESULT **IO_Initialize** (

[IN] CLSID&	objServerCLSID,
[IN] LPCSTR	szServerName,
[IN] LPCSTR	szVendorInfo,
[IN] WORD	wMajorVersion,
[IN] WORD	wMinorVersion,
[IN] WORD	wBuildNumber,
[IN] LPCSTR	szNameSpaceDelimiter

)

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 should be able to handle effectively OPC client originated requests only after calling the IO_SetStatus function.

Parameter	Description
objServerCLSID	The class ID for the OPC server (generated using the GUIDGen.exe utility).
szServerName	Server name information.
szVendorInfo	Vendor information.
wMajorVersion	Major version of the server application.
wMinorVersion	Minor version of the server application.
wBuildNumber	Build number of the server application.
szNameSpaceDelimiter	Separator used to build the server area space.

Table 5: IO_Initialize parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_OUTOFMEM	Memory cannot be allocated.
IO_E_SVRNIT	The Initialize function already called.
IO_E_NULLSTRING	An invalid null string was passed.
IO_E_CFFAIL	The server instance cannot be published.
IO_E_EXCEPTEDERROR	Exception error occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 6: IO_Initialize error codes

1.1.3. IO_Uninitialize

HRESULT **IO_Uninitialize** ()

Description

Free all the DLL allocated resources and close the COM library. This will forcibly disconnect any connected OPC AE client. The server developer can use the **IO_IsServerInUse** function to determine if clients are still using the server.

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_EXCEPTEDERROR	Exception error occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 7: IO_Uninitialize error codes

1.1.4. IO_RegisterServer

HRESULT **IO_RegisterServer** (**[IN]** LPCSTR szServerPath)

Description

Add the necessary server entries in the Windows registry as a COM / DCOM based executable that supports the OPC Alarms and Events interfaces. Then, OPC AE clients will be able to identify the OPC server. It also allows

COM / DCOM to launch the application using its standard activation mechanisms.

Parameter	Description
szServerPath	The path of server executable.

Table 8: IO_RegisterServer parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_SVRREG	The server is already registered.
IO_E_NULLSTRING	Invalid null string.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 9: IO_RegisterServer error codes

1.1.5. IO_ RegisterServerASService

HRESULT **IO_RegisterServerASService([IN]** LPCSTR ExePath**,**
[IN] LPCSTR ServiceName**,**
[IN] LPWSTR ServiceParameters**)**

Description

Add the necessary server entries in the Windows registry to be used as a Windows service.

Parameter	Description
szServerPath	The path of server executable.
ServiceName	The service name
ServiceParameters	The service parameters

Table 10 IO_ RegisterServerASService parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_SVRREG	The server is already registered.
IO_E_NULLSTRING	Invalid null string.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 11: IO_ RegisterServerASService error codes

1.1.6. IO_UnRegisterServer

HRESULT **IO_UnRegisterServer** ()

Description

Remove all server entries that were created by the ***IO_RegisterServer*** function from the system registry. Consequently, the server will no longer be available to OPC AE clients.

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_SVRINUSE	Server is currently in use.
E_EXCEPTION_ERROR	Memory exception occurred.
E_UNKNOWN_ERROR	Unknown error occurred.

Table 12: IO_UnRegisterServer error codes

1.1.7. IO_SetCallBackObjects

```
HRESULT IO_SetCallBackObjects
 (
  [IN] ONCLIENTCONNECT              OnConnectCallBack_,
  [IN] ONCLIENTDISCONNECT           OnDisconnectCallBack_,
  [IN] GETERRORSTRING               GetErrorStringCallBack_,
  [IN] ONACKCONDITION               OnAckConditionCallBack_,
  [IN] SERVERCANUNLOADNOW           ServerCanunloadCallBack_,
  [IN] TRANSLATETOITEMID            TranslateToItemIDCallBack_,
  [IN] CONDITIONSCANBEENABLED       ConditionCanBeEnabledCallBack_,
  [IN] ONENABLECONDITION            OnEnableConditionCallBack_,
  [IN] GETATTRIBCURRENTVALUE        GetAttribCurrentValueCallBack_
  )
```

Description

The developer should implement the callback functions 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 the description of these callbacks later in the guide.

Parameter	Description
OnConnectCallBack	Pointer to OnClientConnect callback.
OnDisconnectCallBack	Pointer to OnClientDisconnect callback.
GetErrorStringCallBack	Pointer to GetErrorString callback.
OnAckConditionCallBack	Pointer to OnAckCondition callback.
ServerCanunloadCallBack	Pointer to ServerCanunload callback.
TranslateToItemIDCallBack	Pointer to TranslateToItemID callback.
ConditionCanBeEnabledCallBack	Pointer to ConditionCanBeEnabled callback.
OnEnableConditionCallBack	Pointer to OnEnableCondition callback.
GetAttribCurrentValueCallBack	Pointer to GetAttribCurrentValue callback.

Table 13: IO_SetCallBackObjects parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 14: IO_SetCallBackObjects error codes

1.1.8. IO_SetStatus

HRESULT **IO_SetStatus** (**[IN]** DWORD hServerStatus)

Description
Set the current OPC AE server status.

Parameter	Description
hServerStatus	The status of the server.

Table 15: IO_SetStatus parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_INVSVRSTATUS	The operation failed.
IO_E_EXCEPTEDERROR	Unknown exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 16: IO_SetStatus error codes

1.1.9. IO_ServerInUse

HRESULT **IO_ServerInUse** ()

Description
This function is used to check if the server is currently in use.

Return Code	Description
IO_E_SVRINUSE	The server is actually in use. One or more clients are currently connected.

IO_E_NSVINUSE	The server is not actually in use.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 17: IO_ServerInUse error codes

1.1.10. IO_SetServerConfig

HRESULT **IO_SetServerConfig** (
[IN] DWORD dwMaxBufferSize,
[IN] DWORD dwMaxBufferTime,
[IN] DWORD dwMinBufferTime,
[IN] DWORD dwMaxEventServer,
[IN] DWORD dwMaxEventSubscription
)

Description
This method set the server configuration.

Parameter	Description
dwMaxBufferSize	The max buffer size supported by the server.
dwMaxBufferTime	The max buffer time supported by the server.
dwMinBufferTime	The min buffer time supported by the server.
dwMaxEventServer	The max server instance allowed by the server.
dwMaxEventSubscription	For each server instance; the max event subscription instance allowed by the server.

Table 18: IO_SetServerConfig parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 19: IO_SetServerConfig error codes

1.1.11. IO_SetServerPeriodTime

HRESULT **IO_SetServerPeriodTime** (**[IN]** DWORD dwPeriodTime)

Description
Used to set (or update) the max amount of time the server sends a new notification.

Parameter	Description
dwPeriodTime	The max amount of time the server send a new notification

Table 20: IO_SetServerPeriodTime parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 21: IO_SetServerPeriodTime error codes

1.1.12. IO_GetConnectedClientNumber

DWORD **IO_GetConnectedClientNumber** ()

Description
Return the number of current connected OPC AE Clients.

Return Code	Description
Number	Number of current connected clients.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 22: IO_GetConnectedClientNumber error codes

1.1.13. IO_RequestDisconnect

HRESULT **IO_RequestDisconnect** (**[IN]** LPCSTR szReason)

Description
Used to send a request for disconnect to all connected clients.

Parameter	Description
szReason	A text string indication the reason for the shutdown request.

Table 23: IO_RequestDisconnect parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown exception occurred.

Table 24: IO_RequestDisconnect error codes

1.1.14. IO_GetStatus

HRESULT **IO_GetStatus** (**[OUT]** DWORD *hServerStatus)

Description
This method returns the current OPC Server status.

Parameter	Description
hServerStatus	A DWORD used to store the current status information of the OPC Server.

Table 25: IO_GetStatus parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 26: IO_GetStatus error codes

1.1.15. IO_SetLogTraceLevel

HRESULT **IO_SetLogTraceLevel** (**[IN]** DWORD dwTraceLevel)

Description
This function is used to set (or update) the trace level to be used by the toolkit.

Parameter	Description
dwTraceLevel	The trace level to be used by the ToolKit.

Table 27: IO_SetLogTraceLevel parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 28: IO_SetLogTraceLevel error codes

1.1.16. IO_LogTrace

HRESULT **IO_LogTrace** (
[IN] DWORD dwLevel,
[IN] LPCSTR pszDesc
)

Description
This function is used to add entry to the log event file.

Parameter	Description
dwLevel	The trace level to use.
pszDesc	The log text.

Table 29: IO_LogTrace parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_EXCEPTEDERROR	Memory exception occurred.

IO_E_UNKERROR	Unknown error occurred.

Table 30: IO_SetLogTraceLevel error codes

1.1.17. IO_GetAttributeValue

```
HRESULT     IO_GetAttributeValue (
                [IN]    LPCSTR    szServerCLSID,
                [IN]    LPCSTR    szServerNode,
                [IN]    LPCSTR    szItemID,
                [IN]    DWORD     dwPropertyID,
                [OUT] VARIANT     *objVariant,
            )
```

Description

Given the interested itemID information; this function will retrieve the current property value for this itemID.

Parameter	Description
szServerCLSID	The OPC Data Access Server CLSID.
szServerNode	The OPC Data Access Serve Address
szItemID	The requested ItemID.
dwPropertyID	The requested Item Property ID.
objVariant	A variant used to store the returned value.

Table 31: IO_GetAttributeValue parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_CFSWRITEREGDB	The CLSID corresponding to the class string was not found in the registry.
E_ IO_E_CFSFAILED	CLSID from ProgID failed for unknown error.
IO_E_GIPFALSE	Get item properties completed with one or more errors. Refer to individual error returns for failure analysis.
IO_E_GIPUNKITEMID	Get item properties failed. The ItemID is not in the

	server address space.
IO_E_GIPINVITEMID	Get item properties. The ItemID is not syntactically valid.
IO_E_GIPOUTOMEM	Get item properties failed. Not enough memory.
IO_E_GIPINVARG	Get item properties failed. An invalid argument was passed.
IO_E_GIPFAILED	Get item properties failed.
IO_E_GIPUNKERROR	Get item properties failed. Unknown error occurred.
IO_E_QINOINTERFACE	Query interface failed. Not supported interface reason.
IO_E_QIUNKERROR	Query interface failed. Unknown error occurred.
IO_E_CIINVARG	Invalid arguments.
IO_E_CINO_ALLINTERFACES	Failed to retrieve IID_IOPCServer.
IO_E_CINOINTERFACE	None of the interfaces were successfully retrieved.
IO_E_CISVRUNAVAILABLE	The RPC Server is not available.
IO_E_CIACCESSDENIED	Access denied.
IO_E_CICLASSNOTREG	CLSID is not properly registered. Can also indicate that the value you specified is not in the registry.
IO_E_CIREADREGDB	Error reading the registration database.
IO_E_CIDLLNOTFOUND	In-process DLL or handler DLL not found (depends on context).
IO_E_CIAPPNOTFOUND	EXE not found (CLSCTX_LOCAL_SERVER only).
IO_E_CIERRORINDLL	EXE has error in image.
IO_E_CIAPPDIDNTREG	EXE was launched, but it didn't register class object (may or may not have shut down.
IO_E_CIUNEXPECTERROR	Unknown error.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 32: IO_GetAttributeValue error codes

1.2. EventSpace Management

1.2.1. IO_AddCategory

HRESULT **IO_AddCategory** (
[IN] DWORD dwCategoryID,
[IN] LPCSTR szCategoryName,
[IN] DWORD dwCategoryEventType
)

Description

This operation will add a new category to the server categories list.

Parameter	Description
dwCategoryID	ID of the category.
szCategoryName	Description of the category.
dwCategoryEventType	Event Type of the category.

Table 33: IO_AddCategory parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_CATADDED	Category already exists in the server Config having the same CategoryID as the one to add.
IO_E_EXCEPTEDERROR	Exception error occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 34: IO_AddCategory error codes

1.2.2. IO_InitCategories

HRESULT **IO_InitCategories** (
[IN] DWORD dwNumCategories,
[IN] DWORD* dwCategoriesIDs,
[IN] LPCSTR* szCategoriesNames,
[IN] DWORD* dwCategoriesEventTypes
)

Description

This operation will add one or more categories to the server categories list in the same call.

Parameter	Description
dwNumCategories	The number of condition names being added.
dwCategoriesIDs	Array of DWORD containing the IDs of the categories being added.
szCategoriesNames	Array of string containing the names of the categories being added.
dwCategoriesEventTypes	Array of DWORD containing the event types of the categories being added. The Event Types can be: ○ 4 ➔ OPC_CONDITION_EVENT. ○ 2 ➔ OPC_TRACKING_EVENT. ○ 1 ➔ OPC_SIMPLE_EVENT.

Table 35: IO_InitCategories parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_S_FALSE	One or more categories initialization failed.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 36: IO_InitCategories error codes

1.2.3. IO_RemoveCategory

HRESULT **IO_RemoveCategory** (**[IN]** DWORD dwCategoryID)

Description

This operation will remove one category from the server categories list.

Parameter	Description
dwCategoryID	ID of the category being added.

Table 37: IO_RemoveCategory parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_S_FALSE	The category to be removed is related to one or more attributes and/or conditions.
IO_E_INVCATID	An invalid category ID was given.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 38: IO_RemoveCategory error codes

1.2.4. IO_GetCategory

HRESULT **IO_GetCategory** (

[IN]	DWORD	dwCategoryID,
[OUT]	LPWSTR*	pszCategoryName,
[OUT]	DWORD*	dwCategoryEventType

)

Description

This operation return information related to the given category ID.

Parameter	Description
dwCategoryID	ID of the category of interest.
pszCategoryName	A reference to the string where the corresponding category name will be stored.
dwCategoryEventType	A reference to the DWORD where the corresponding category event type will be stored.

Table 39: IO_GetCategory parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_INVCATID	An invalid category ID was given.
IO_E_EXCEPTEDERROR	Memory exception occurs.

IO_E_UNKERROR	Unknown error occurs.

Table 40: IO_GetCategory error codes

1.2.5. IO_CategoriesList

HRESULT IO_CategoriesList (
[OUT] DWORD* dwNumCategories,
[OUT] DWORD** dwCategoriesIDs,
[OUT] LPWSTR** pszCategoriesNames,
[OUT] DWORD** dwCategoriesEventTypes
)

Description

This operation returns the list of event categories implemented by the server.

Parameter	Description
dwNumCategories	The number of available event categories.
dwCategoriesIDs	Array of DWORD codes of the vendor-specific event categories.
pszCategoriesNames	Array of the correspondent strings that stores the name or description of each event category ID.
dwCategoriesEventTypes	Array of the correspondent Event Type of each event category ID.

Table 41: IO_CategoriesList parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_OUTOFMEM	Operation failed. Not enough memory.
IO_E_EXCEPTEDERROR	Memory exception occurs.
IO_E_UNKERROR	Unknown error occurs.

Table 42: IO_CategoriesList error codes

1.2.6. IO_ClearCategories

HRESULT **IO_ClearCategories** ( )

Description
This operation will remove all event categories from server categories list.

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_FAILED	One or more categories are related to one or more attributes and/or condition.
IO_E_EXCEPTEDERROR	Memory exception occurs.
IO_E_UNKERROR	Unknown error occurs.

Table 43: IO_ClearCategories error codes

1.2.7. IO_AddAttribute

HRESULT **IO_AddAttribute** (
[IN] DWORD dwAttributeID,
[IN] LPCSTR szAttributeName,
[IN] VARTYPE vtAttributeType,
[IN] DWORD dwCategoryID
)

Description
This operation will add a new attribute to the server vender specific attributes list.

Parameter	Description
dwAttributeID	ID of the attribute that you want to add.
szAttributeName	Description of the attribute.
vtAttributeType	Attribute data type.
dwCategoryID	Related category ID.

Table 44: IO_AddAttribute parameters

Return Code	Description
IO_S_OK	The operation succeeded.

IO_E_INVCATID	An invalid category was given.
IO_E_ATTREXIST	The attribute that you want to add already exists in the server Config related with the given category ID.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 45: IO_AddAttribute error codes

1.2.8. IO_InitAttributes

HRESULT **IO_InitAttributes** (

[IN] DWORD	dwNumAttributes,
[IN] DWORD*	dwAttributesIDs,
[IN] LPCSTR*	szAttributesNames,
[IN] VARTYPE*	vtAttributesTypes,
[IN] DWORD*	dwCategoryID

)

Description

This operation will add one or more attributes to the server attributes list in the same call.

Parameter	Description
dwNumAttributes	The number of condition names being added.
dwAttributesIDs	Array of DWORD containing the IDs of the attributes being added.
szAttributesNames	Array of string containing the names of the attributes being added.
vtAttributesTypes	Array of DWORD containing the event types of the attributes being added.
dwCategoryID	Array of DWORD containing the related categories ID of the attributes being added.

Table 46: IO_InitAttributes parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_S_FALSE	One or more attributes initialization was failed.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 47: IO_InitAttributes error codes

1.2.9. IO_RemoveAttribute

HRESULT **IO_RemoveAttribute** (**[IN]** DWORD dwAttributeID)

Description
Remove an attribute from the server EventSpace.

Parameter	Description
dwAttributeID	The ID of the attribute to remove.

Table 48: IO_RemoveAttribute parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_INVATTRID	Invalid attribute ID.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 49: IO_RemoveAttribute error codes

1.2.10. IO_GetAttribute

HRESULT **IO_GetAttribute** (
[IN] DWORD dwAttributeID,
[OUT] LPWSTR* szAttributeName,
[OUT] VARTYPE* vtAttributeType,
[OUT] DWORD* dwCategoryID
)

Description
Retrieve attribute information.

Parameter	Description
dwAttributeID	The ID of the attribute to return.
szAttributeName	A string to store the returned attribute name.
vtAttributeType	A varType to store the returned attribute type.
dwCategoryID	A DWORD used to store the ID of the related category.

Table 50: IO_GetAttribute parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_INVATTRID	Invalid attribute ID.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 51: IO_GetAttribute error codes

1.2.11. IO_GetAttributesList

HRESULT **IO_GetAttributesList** (
[OUT] DWORD* dwNumAttributes,
[OUT] DWORD** dwAttributesIDs,
[OUT] LPWSTR** pszAttributesNames,
[OUT] VARTYPE** vtAttributesTypes,
[OUT] DWORD** dwCategoriesIDs
)

Description
Return all available event categories.

Parameter	Description
dwNumAttributes	The number of attribute returned by the function.
dwAttributesIDs	Array of DWORD codes for the vendor-specific attributes

	ID implemented by the server.
pszAttributesNames	Array of strings for the text names for each of the event attribute IDs.
vtAttributesTypes	Array of VARTYPE for the data type for each event attribute IDs.
dwCategoriesIDs	Array of DWORD codes for the related categories for each event attribute IDs.

Table 52: IO_GetAttributesList parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_OUTOFMEM	Operation failed. Not enough memory.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 53: IO_GetAttributesList error codes

1.2.12. IO_GetCategoryAttributesList

HRESULT **IO_GetCategoryAttributesList** (
[IN] DWORD dwCategoryID,
[OUT] DWORD* dwNumAttributes,
[OUT] DWORD** dwAttributesIDs,
[OUT] LPWSTR** pszAttributesNames,
[OUT] VARTYPE** vtAttributesTypes
)

Description
Return the list of available attributes within an event category.

Parameter	Description
dwCategoryID	A DWORD event category code, as returned by the GetCategoriesList method. Only the names of attributes within this event category are returned.
dwNumAttributes	The number of attributes being returned.

dwAttributesIDs	Array of DWORD containing the attributes ID for the specified event attribute.
pszAttributesNames	Array of strings containing the attributes name for the specified event attribute.
vtAttributesTypes	Array of VARTYPE containing the attributes type for the specified event attribute.

Table 54: IO_GetCategoryAttributesList parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_OUTOFMEM	Operation failed for no enough memory.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 55: IO_GetCategoryAttributesList error codes

1.2.13. IO_ClearAttributes

HRESULT **IO_ClearAttributes** ()

Description
Remove all available attributes from the server EventSpace.

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 56: IO_ClearAttributes error codes

1.2.14. IO_AddCondition

```
HRESULT IO_AddCondition (
                          [IN] DWORD      dwCategoryID,
                          [IN] LPCSTR     szConditionName
                         )
```

Description
Add a condition to the server EventSpace.

Parameter	Description
dwCategoryID	The ID of the condition related category being added.
szConditionName	The name of the condition being added.

Table 57: IO_AddCondition parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_INVCATID	Invalid category ID.
IO_E_CONDEXIST	Condition name already exist.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 58: IO_AddCondition error codes

1.2.15. IO_InitConditions

HRESULT **IO_InitConditions** (
[IN] DWORD dwNumCondition,
[IN] DWORD* dwCategoriesIDs,
[IN] LPCSTR* szConditionsNames
)

Description
Add a list of sub-condition to the server EventSpace.

Parameter	Description
dwNumCondition	The number of conditions being added.
dwCategoriesIDs	Array of DWORD codes for the conditions related event categories.
szConditionsNames	Array of strings for the condition names being added.

Table 59: IO_InitConditions parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_S_FALSE	One or more conditions initialization was failed.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 60: IO_InitConditions error codes

1.2.16. IO_RemoveCondition

HRESULT **IO_RemoveCondition** (**[IN]** LPCSTR szConditionsName)

Description
Remove condition from the server EventSpace.

Parameter	Description
szConditionsName	The name of the condition being added.

Table 61: IO_RemoveCondition parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_INVCONDNAME	Invalid condition name
IO_E_FAILED	Failed for related sub-conditions.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 62: IO_RemoveCondition error codes

1.2.17. IO_GetCondition

HRESULT **IO_GetCondition** (
[IN] LPCSTR szConditionsName,
[OUT] DWORD* dwCategoryID
)

Description
Retrieve condition information.

Parameter	Description
szConditionsName	The name of condition being added.
dwCategoryID	A DWORD code for the correspondent category ID

Table 63: IO_GetCondition parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_INVCONDNAME	Invalid condition name.
IO_E_EXCEPTEDERROR	Memory condition occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 64: IO_GetCondition error codes

1.2.18. IO_GetCategoryConditionsList

HRESULT **IO_GetCategoryConditionsList** (
[IN] DWORD dwCategoryID,
[OUT] DWORD* dwNumConditions,
[OUT] LPWSTR** pszConditionsNames
)

Description
Return the list of available sub-conditions within an event category.

Parameter	Description
dwCategoryID	A DWORD event category code, as returned by the GetCategoriesList method. Only the names of conditions within this event category are returned.
dwNumConditions	The number of conditions being returned.
pszConditionsNames	Array of strings containing the conditions name.

Table 65: IO_GetCategoryConditionsList parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_OUTOFMEM	Operation failed for no enough memory.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 66: IO_GetCategoryConditionsList error codes

1.2.19. IO_GetConditionsList

HRESULT **IO_GetConditionsList** (
[OUT] DWORD* dwNumConditions,
[OUT] DWORD** dwCategoriesIDs,
[OUT] LPWSTR** pszConditionsNames
)

Description
Return all available conditions.

Parameter	Description
dwNumConditions	The number of conditions returned by the function.
dwCategoriesIDs	Array of DWORD codes for the event categories related to available conditions.
pszConditionsNames	Array of strings for the available condition names implemented by the server.

Table 67: IO_GetConditionsList parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_OUTOFMEM	Operation failed. Not enough memory.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 68: IO_GetConditionsList error codes

1.2.20. IO_ClearConditions

HRESULT **IO_ClearConditions** ()

Description

Remove all available conditions from the server EventSpace.

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_FAILED	Failed for related sub-conditions.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 69: IO_ClearConditions error codes

1.2.21. IO_AddSubCondition

HRESULT **IO_AddSubCondition** (
[IN] LPCSTR szConditionName,
[IN] LPCSTR szSubConditionName,
[IN] LPCSTR szSubConditionDescription
)

Description

Add a sub-condition to the server EventSpace.

Parameter	Description
szConditionName	The related condition name for the sub-condition being added.
szSubConditionName	The name of sub-condition being added.
szSubConditionDescription	The description of sub-condition being added.

Table 70: IO_AddSubCondition parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_INVCONDNAME	Invalid condition name.

IO_E_SUBCONDEXIST	Sub-condition name already exist.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 71: IO_AddSubCondition error codes

1.2.22. IO_InitSubConditions

HRESULT **IO_InitSubConditions** (
[IN] DWORD dwNumSubConditions,
[IN] LPCSTR* szConditionsNames,
[IN] LPCSTR* szSubConditionsNames,
[IN] LPCSTR* szSubConditionsDescriptions
)

Description
Add a list of sub-condition to the server EventSpace.

Parameter	Description
dwNumSubConditions	The number of sub-condition being added.
szConditionsNames	Array of strings for condition names related to sub-condition names being added.
szSubConditionsNames	Array of strings for sub-condition names being added.
szSubConditionsDescriptions	Array of strings for sub-condition descriptions being added.

Table 72: IO_InitSubConditions parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_S_FALSE	One or more sub-conditions initialization failed.
IO_E_EXCEPTEDERROR	Memory exception occurred
IO_E_UNKERROR	Unknown error occurred.

Table 73: IO_InitSubConditions error codes

1.2.23. IO_RemoveSubCondition

HRESULT **IO_RemoveSubCondition** (
[IN] LPCSTR szConditionName,
[IN] LPCSTR szSubConditionName
)

Description
Remove sub-condition from the server EventSpace.

Parameter	Description
szConditionName	The condition name related to the sub-condition name being removed.
szSubConditionName	The name of sub-condition name being added.

Table 74: IO_RemoveSubCondition parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_FAILED	Operation failed for invalid couple (Condition-SubCondition).
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 75: IO_RemoveSubCondition error codes

1.2.24. IO_GetSubCondition

HRESULT **IO_GetSubCondition** (
[IN] LPCSTR szConditionName,
[OUT] LPCSTR szSubConditionName,
[OUT] LPWSTR* szSubConditionDescription
)

Description
Retrieve sub-condition information.

Parameter	Description
szConditionName	The name of condition name related to the sub-condition to return information.
szSubConditionName	The name of sub-condition to return information.
szSubConditionDescription	A string for the description of the condition.

Table 76: IO_GetSubCondition parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_INVCONDSUBCONDNAME	Invalid Condition and/or Sub-Condition name.
IO_E_EXCEPTEDERROR	Memory exception occurred
IO_E_UNKERROR	Unknown error occurred

Table 77: IO_GetSubCondition error codes

1.2.25. IO_GetSubConditionsList

HRESULT **IO_GetSubConditionsList** (
[OUT] DWORD* dwNumSubConditions,
[OUT] LPWSTR** pszConditionsNames,
[OUT] LPWSTR** pszSubConditionsNames,
[OUT] LPWSTR* pszSubConditionsDescriptions
)

Description
Return all available sub-conditions.

Parameter	Description
dwNumSubConditions	The number of sub-condition supported by the server.
pszConditionsNames	Array of strings for the condition names.
pszSubConditionsNames	Array of strings for the sub-condition names.
pszSubConditionsDescriptions	Array of strings for the sub-condition descriptions.

Table 78: IO_GetSubConditionsList parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_OUTOFMEM	Operation failed for no enough of memory.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 79: IO_GetSubConditionsList error codes

1.2.26. IO_GetCondSubConditionsList

HRESULT
IO_GetCondSubConditionsList (
[IN] LPCSTR szConditionName,
[OUT] DWORD* dwNumSubConditions,
[OUT] LPWSTR ** pszSubConditionsNames,
[OUT] LPWSTR ** pszSubConditionsDescriptions
)

Description
Return the list of available sub-conditions within a condition name.

Parameter	Description
szConditionName	A condition name code, as returned by the GetConditionList method. Only the names of sub-conditions within this condition are returned.
dwNumSubConditions	The number of sub-condition related to this condition.
pszSubConditionsNames	Array of strings for the sub-condition names.
pszSubConditionsDescriptions	Array of strings for the sub-condition descriptions.

Table 80: IO_GetCondSubConditionsList parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_OUTOFMEM	Operation failed. Not enough memory.

IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 81: IO_GetCondSubConditionsList error codes

1.2.27. IO_ClearSubConditions

HRESULT **IO_ClearSubConditions** ()

Description
Remove all sub-conditions from the server EventSpace.

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 82: IO_ClearSubConditions error codes

1.3. AreaSpace Management

1.3.1. IO_AddAreaNode

HRESULT **IO_AddAreaNode** (**[IN]** LPCSTR szAreaPath)

Description
Add an area to the server areaspace.

Parameter	Description
szAreaPath	The path of the area being added.

Table 83: IO_AddAreaNode parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred

Table 84: IO_AddAreaNode error codes

1.3.2. IO_InitAreaNodes

HRESULT **IO_InitAreaNodes** (
[IN] DWORD dwNumAreas,
[IN] LPCSTR* szAreasPaths
)

Description
Add a list of areas to the server areaspace.

Parameter	Description
dwNumAreas	The number of areas to add.
szAreasPaths	Array of strings for area paths to add.

Table 85: IO_InitAreaNodes parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_S_FALSE	One or more area nodes initialization failed.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 86: IO_InitAreaNodes error codes

1.3.3. IO_RemoveAreaNode

HRESULT **IO_RemoveAreaNode** (**[IN]** LPCSTR szAreaPath)

Description
Remove an area from the server areaspace.

Parameter	Description
szAreaPath	The area path to remove.

Table 87: IO_RemoveAreaNode parameters

Return Code	Description
IO_S_OK	The operation succeeded.

IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 88: IO_RemoveAreaNode error codes

1.3.4. IO_InitSourceNodes

HRESULT **IO_InitSourceNodes** (
[IN] DWORD dwNumSources,
[IN] LPCSTR* szSourcesPaths
)

Description
Add a list of sources to the server areaspace.

Parameter	Description
dwNumSources	The number of sources being added.
szSourcesPaths	Array of strings for the source paths being added.

Table 89: IO_InitSourceNodes parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_S_FALSE	One or more source nodes initialization failed.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 90: IO_InitSourceNodes error codes

1.3.5. IO_AddSourceNode

HRESULT **IO_AddSourceNode** (**[IN]** LPCSTR szSourcePath)

Description
Add a source to the server areaspace.

Parameter	Description
szSourcePath	The path of the source to add.

Table 91: IO_AddSourceNode error codes

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 92: IO_AddSourceNode error codes

1.3.6. IO_RemoveSourceNode

HRESULT **IO_RemoveSourceNode** (**[IN]** LPCSTR szSourcePath)

Description
Remove a source from the server areaspace.

Parameter	Description
szSourcePath	The path of the source to remove.

Table 93: IO_RemoveSourceNode parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred

Table 94: IO_RemoveSourceNode error codes

1.4. Server Source Cache Management

1.4.1. IO_AddSourceConditionToCache

HRESULT **IO_AddSourceConditionToCache** (
[IN] LPCSTR szSourceName,
[IN] LPCSTR szConditionName
)

Description
Add a condition to the server cache.

Parameter	Description
szSourceName	The source name being added to the server cache.
szConditionName	The condition name being added to the server cache.

Table 95: IO_AddSourceConditionToCache parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_NULLSTRING	Invalid null string for source and/or condition name.
IO_E_INVSRCNAME	Source name cannot be found.
IO_E_INVCONDNAME	Condition name cannot be found.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 96: IO_AddSourceConditionToCache error codes

1.4.2. IO_RemoveConditionFromCache

HRESULT **IO_RemoveConditionFromCache** (
[IN] LPCSTR szSourceName,
[IN] LPCSTR szConditionName
)

Description
Remove a condition from the server cache.

Parameter	Description
szSourceName	The source name to remove from the server cache.
szConditionName	The condition name to remove from the server cache.

Table 97: IO_RemoveConditionFromCache parameters

Return Code	Description
IO_S_OK	The operation succeeded.

IO_E_NULLSTRING	Invalid null string for source and/or condition name.
IO_E_INVSRCNAME	Source name cannot be found.
IO_E_INVCONDNAME	Condition name cannot be found.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 98: IO_RemoveConditionFromCache error codes

1.4.3. IO_AddSubConditionToCache

HRESULT **IO_AddSubConditionToCache** (
[IN] LPCSTR szSourceName,
[IN] LPCSTR szConditionName,
[IN] LPCSTR szSubConditionsName,
[IN] DWORD dwSeverity = 100,
[IN] LPCSTR szSubConditionsMessage = NULL)

Description
Add a sub-condition to the server cache.

Parameter	Description
szSourceName	The source name related to the sub-condition being added to the server cache.
szConditionName	The condition name related to the sub-condition being added to the server cache.
szSubConditionsName	The sub-condition name being added to the server cache.
dwSeverity	The severity of the sub-condition being added to the server cache.
szSubConditionsMessage	The message related to the sub-condition being added to the server cache.

Table 99: IO_AddSubConditionToCache parameters

Return Code	Description
IO_S_OK	The operation succeeded.

IO_E_NULLSTRING	Invalid null string for source, condition and/or sub-condition name.
IO_E_INVSRCNAME	Source name cannot be found.
IO_E_INVCONDNAME	Condition name cannot be found.
IO_E_INVSUBCONDNAME	Sub-condition name not found.
IO_E_INVSEVERITY	Invalid severity value.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 100: IO_AddSubConditionToCache error codes

1.4.4. IO_RemoveSubConditionFromCache

HRESULT **IO_RemoveSubConditionFromCache** (
[IN] LPCSTR szSourceName,
[IN] LPCSTR szConditionName,
[IN] LPCSTR szSubConditionsName
)

Description
Remove a sub-condition from the server cache.

Parameter	Description
szSourceName	The source name related to the sub-condition to remove from the server cache.
szConditionName	The condition name related to the sub-condition to remove from the server cache.
szSubConditionsName	The sub-condition name being removed from the server cache.

Table 101: IO_RemoveSubConditionFromCache parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_NULLSTRING	Invalid null string for source, condition

	and/or sub-condition name.
IO_E_INVSRCNAME	Source name cannot be found.
IO_E_INVCONDNAME	Condition name cannot be found.
IO_E_INVSUBCONDNAME	Sub-condition name not found.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 102: IO_RemoveSubConditionFromCache error codes

1.5. Event Notification Management

1.5.1. IO_GenerateConditionEvent

HRESULT **IO_GenerateConditionEvent** (

[IN] LPCSTR szSourceName,
[IN] LPCSTR szConditionName,
[IN] LPCSTR szActiveSubCondition,
[IN] BOOL bActive,
[IN] LPCSTR szMessage,
[IN] DWORD dwSeverity,
[IN] DWORD dwQuality,
[IN] BOOL bAck,
[IN] FILETIME pftTime
)

Description
This function is used to send event notification related to OPC_CONDITION_EVENT.

Parameter	Description
szSourceName	The source of event notification.
szConditionName	The name of condition related to this event notification.
szActiveSubCondition	The name of the current sub-condition, for multi-state conditions. For a single-state condition, this contains the condition name.
bActive	This flag indicates the state of this event notification.

szMessage	Event notification message describing the event.
dwSeverity	Event severity (0..1000).
dwQuality	Quality associated with the condition state.
bAck	This flag indicates that the related condition requires acknowledgment of this event. The determination of those events which require acknowledgment is server specific.
pftTime	Time of the event occurrence

Table 103: IO_GenerateConditionEvent parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_NULLSTRING	Invalid NULL string.
IO_E_INVSRCNAME	Invalid source name.
IO_E_INVCONDNAME	Invalid condition name.
IO_E_INVSUBCONDNAME	Invalid sub-condition name.
IO_E_INVCATID	Invalid event category ID.
IO_E_INVSEVERITY	Invalid severity value.
IO_E_MULTISTATECOND	The condition is multi-state and should have an active sub-condition.
IO_E_EXCEPTEDERROR	Memory exception.
IO_E_UNKERROR	Unknown error.

Table 104: IO_GenerateConditionEvent error codes

1.5.2. IO_GenerateSimpleEvent

HRESULT **IO_GenerateSimpleEvent** (

[IN] LPCSTR szSourceName,
[IN] DWORD dwEventCat,
[IN] DWORD dwSeverity,
[IN] LPCSTR szMessage,
[IN] FILETIME *pftTime

)

Description
This function is used to send event notification related to OPC_SIMPLE_EVENT.

Parameter	Description
szSourceName	The source of event notification.
dwEventCat	The category related to this event notification.
dwSeverity	Event severity (0..1000).
szMessage	Event notification message describing the event.
pftTime	Time of the event occurrence

Table 105: IO_GenerateSimpleEvent parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_NULLSTRING	Invalid NULL string.
IO_E_INVSRCNAME	Invalid source name.
IO_E_INVCATID	Invalid category ID.
IO_E_INVSEVERITY	Invalid severity value.
IO_E_EXCEPTEDERROR	Memory exception.
IO_E_UNKERROR	Unknown error.

Table 106: IO_GenerateSimpleEvent error codes

1.5.3. IO_GenerateTrackingEvent

HRESULT **IO_GenerateTrackingEvent** (

[IN] LPCSTR szSourceName,
[IN] DWORD dwEventCat,
[IN] DWORD dwSeverity,
[IN] LPCSTR szMessage,
[IN] LPCSTR szActor,
[IN] FILETIME *pftTime)

Description
This function is used to send event notification related to OPC_TRACKING_EVENT.

Parameter	Description
szSourceName	The source of event notification.
dwEventCat	The category related to this event notification.
dwSeverity	Event severity (0..1000).
szMessage	Event notification message describing the event.
szActor	This is the actor ID for the event notification.
pftTime	Time of the event occurrence

Table 107: IO_GenerateTrackingEvent parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_NULLSTRING	Invalid NULL string.
IO_E_INVSRCNAME	Invalid source name.
IO_E_INVCATID	Invalid category ID.
IO_E_INVSEVERITY	Invalid severity value.
IO_E_EXCEPTEDERROR	Memory exception.
IO_E_UNKERROR	Unknown error.

Table 108: IO_GenerateTrackingEvent error codes

1.5.4. IO_EnableCondition

```
HRESULT IO_EnableCondition (
                                [IN] LPCSTR     szSourceName,
                                [IN] LPCSTR     szConditionName,
                                [IN] DWORD      dwQuality,
                                [IN] FILETIME       *pftTime   )
```

Description
This method is used to enable monitoring of condition events related to a specified source name.

Parameter	Description
szSourceName	The source name related to the condition being enabled.
szConditionName	The condition name being enabled.
dwQuality	Quality associated with the condition state.
pftTime	Time of the operation.

Table 109: IO_EnableCondition parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_NULLSTRING	Source name and/or condition name cannot be null.
IO_E_INVSRCMAP	Source name not found in the server cache.
IO_E_INVCONDMAP	Condition name not found in the server cache.
IO_E_CONDENABLED	The condition is already enabled.
IO_E_EXCEPTEDERROR	Memory exception.
IO_E_UNKERROR	Unknown error.

Table 110: IO_EnableCondition error codes

1.5.5. IO_DisableCondition

HRESULT **IO_DisableCondition** (
[IN] LPCSTR szSourceName,
[IN] LPCSTR szConditionName,
[IN] DWORD dwQuality,
[IN] FILETIME *pftTime
)

Description
This method is used to disable monitoring of condition events related to a specified source name.

Parameter	Description
szSourceName	The source name related to the condition being disabled.
szConditionName	The condition name being disabled.
dwQuality	Quality associated with the condition state.
pftTime	Time of the operation.

Table 111: IO_DisableCondition parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_NULLSTRING	Source name and/or condition name cannot be null.
IO_E_INVSRCMAP	Source name not found in the server cache.
IO_E_INVCONDMAP	Condition name not found in the server cache.
IO_E_CONDDISABLED	The condition is already disabled.
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 112: IO_DisableCondition error codes

1.5.6. IO_ActivateCondition

```
HRESULT IO_ActivateCondition (
                    [IN] LPCSTR         szSourceName,
                    [IN] LPCSTR         szConditionName,
                    [IN] LPCSTR         szActiveSubConditionName,
                    [IN] LPCSTR         szMessage,
                    [IN] BOOL           bAck,
                    [IN] DWORD          dwQuality,
                    [IN] FILETIME           ftTime
                                )
```

Description

This function is used to activate a condition. In case of multi-state condition it's used to identify the current active sub-condition.

Parameter	Description
szSourceName	The source name related to the condition being activated.
szConditionName	The condition name being activated.
szActiveSubConditionName	The sub-condition being activated.
szMessage	Message describing the operation.
bAck	This flag indicates that the related condition requires acknowledgment.
dwQuality	Quality associated with the condition state.
ftTime	Time of the operation.

Table 113: IO_ActivateCondition parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_NULLSTRING	Source name and/or condition name cannot be null.
IO_E_INVSRCMAP	Source name not found in the server cache.
IO_E_INVCONDMAP	Condition name not found in the server cache.
IO_E_INVSUBCONDMAP	Invalid sub-condition name.
IO_E_CONDDISABLED	The condition is disabled.
IO_E_SUBCONDACTIVE	The sub-condition is already active and the condition is unacknowledged. (mutli-state condition)
IO_E_CONDACTIVE	The condition is already active and the condition is unacknowledged. (mono-state condition)
IO_E_EXCEPTEDERROR	Memory exception occurred.
IO_E_UNKERROR	Unknown error occurred.

Table 114: IO_ActivateCondition error codes

1.5.7. IO_DeactivateCondition

```
HRESULT IO_DeactivateCondition (
                    [IN] LPCSTR  szSourceName,
                    [IN] LPCSTR  szConditionName,
                    [IN] LPCSTR  szSubconditionName,
                    [IN] LPCSTR  szMessage,
                    [IN] BOOL    bAck,
                    [IN] DWORD   dwQuality,
                    [IN] FILETIME ftTime
                         )
```

Description

This function is used to deactivate the condition and return it to the normal state.

Parameter	Description
szSourceName	The source name related to the condition being deactivated.
szConditionName	The condition name being deactivated.
szSubconditionName	The subcondition name related to the condition being deactivated
szMessage	Message describing the operation.
bAck	This flag indicates that the related condition requires acknowledgment.
dwQuality	Quality associated with the condition state.
ftTime	Time of the operation.

Table 115: IO_DeactivateCondition parameters

Return Code	Description
IO_S_OK	The operation succeeded.
IO_E_NULLSTRING	Source name and/or condition name cannot be null.
IO_E_INVSRCMAP	Source name not found in the server cache.
IO_E_INVCONDMAP	Condition name not found in the server cache.

IO_E_CONDDISABLED	The condition is disabled.
IO_E_CONDINACTIVE	The condition is currently inactive.
IO_E_EXCEPTEDERROR	Memory exception.
IO_E_UNKERROR	Unknown error occurred.

Table 116: IO_DeactivateCondition error codes

2. CallBacks Specification

2.1. Overview

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

Prototypes

VOID	OnClientConnect(DWORD *pdwResult)
VOID	OnClientDisconnect(DWORD *pdwResult)
VOID	GetErrorString(DWORD dwError, BSTR *pbstrError)
VOID	OnAckCondition(BSTR bstrSourceName, BSTR bstrConditionName, BSTR bstrACK, BSTR bstrComment, DWORD *pdwResult)
VOID	GetAttribCurrentValue (DWORD dwNumAttributes, VARIANT *pVar, DWORD *pdwResult)
VOID	TranslateToItemID (DWORD dwAttrID, BSTR *pbstrAttrItemID, BSTR *pbstrNodeName, BSTR *pbstrCLSID, DWORD *pdwResult)
VOID	ServerCanUnloadNow(DWORD *pdwResult)

VOID	ConditionsCanBeEnabled(DWORD *pdwResult)
VOID	OnEnableCondition(BSTR bstrSourceName, BSTR bstrConditionName, BOOL bEnable, DWORD *pdwResult)

Table 117: CallBacks methods

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

Pointer	Referenced Object (callback function)
ONCLIENTCONNECT	OnClientConnect
ONCLIENTDISCONNECT	OnClientDisconnect
GETERRORSTRING	(*) GetErrorString
TRANSLATETOITEMID	(*) TranslateToItemID
ONACKCONDITION	(*) OnAckCondition
SERVERCANUNLOADNOW	ServerCanUnloadNow
CONDITIONSCANBEENABLED	ConditionsCanBeEnabled
ONENABLECONDITION	(*) OnEnableCondition
GETATTRIBCURRENTVALUE	(*) GetAttribCurrentValue

Table 118: Callbacks matching

2.2. 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.

2.2.1. OnClientConnect

VOID **OnClientConnect** (DWORD *pdwResult)

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 is rejected.

Return Code	Description
S_OK	The operation succeeded.
E_ACCESSDENIED	The access for this client is denied.
E_UNEXPECTED	An unexpected error.
E_OUTOFMEMORY	Enough memory.
E_FAIL	The operation failed.
S_xxx (success code) E_xxx (error code)	Vendor specific error code.

Table 119: OnClientConnect error codes

Default behavior
Return S_OK.

2.2.2. OnClientDisconnect

VOID **OnClientDisconnect** (DWORD *pdwResult)

Description
This function will be called whenever a client has disconnected. The server can then perform some cleanup operations.

Return Code	Description
S_OK	The operation succeeded.
S_xxx (success code) E_xxx (error code)	Vendor specific error code.

Table 120: OnClientDisconnect error codes

Default behavior
Return S_OK.

2.2.3. GetErrorString

VOID GetErrorString ([IN] DWORD dwError, BSTR *pbstrError)

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	Description
dwError	A driver specific error code that need to be described.

Table 121: GetErrorString parameters

Returned Values
Return NULL or a valid BSTR describing the given error code
Default behavior
Return “vendor specific error” string.

2.2.4. TranslateToItemID

```
HRESULT        TranslateToItemID (
    [IN]    DWORD  dwAttrID,
    [OUT]   BSTR   *pbstrAttrItemID,
    [OUT]   BSTR   *pbstrNodeName,
    [OUT]   BSTR   *pbstrCLSID,
    [OUT]   DWORD  *pdwResult)
```

Description
Given an attribute ID code, return the item ID string, the server clsid and the server address corresponding to this attribute ID.

Parameter	Description
dwAttrID	The attribute ID.
ppszAttrItemID	A string to contain the correspondent itemID.
ppszNodeName	A string to contain the correspondent server node address.
ppCLSID	A string to contain the correspondent server clsid.

Table 122: TranslateToItemID parameters

Return Code	Description

S_OK	The operation succeeded.
E_FAIL	The operation failed.
S_xxx (success code) E_xxx (error code)	Vendor specific error code.

Table 123: TranslateToItemID error codes

Default behavior

Return S_OK with null strings for ItemID and Node Name output parameters; for CLSID the default value should be "{00000000-0000-0000-0000-000000000000}"

2.2.5. OnAckCondition

VOID **OnAckCondition** (
[IN]BSTR pszSourceName,
[IN]BSTR pszConditionName,
[IN]BSTR pszACK,
[IN]BSTR pszComment,
[OUT]DWORD *pdwResult)

Description

This function is used to acknowledge one condition in the Event Server. It is called whenever one condition should be acknowledged.

Parameter	Description
pszSourceName	Source for each condition will be acknowledged.
pszConditionName	Condition name being acknowledged.
pszACK	Acknowledge Value
pszComment	Written comment

Table 124: OnAckCondition parameters

Return Code	Description
S_OK	The operation succeeded.
E_FAIL	The operation failed.

S_xxx (success code) E_xxx (error code)	Vendor specific error code.

Table 125: OnAckCondition error codes

Default behavior
Return S_OK.

2.2.6. ServerCanUnloadNow

VOID **ServerCanUnloadNow** (DWORD *pdwResult)

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

Return Code	Description
S_OK	The operation succeeded.
S_xxx (success code) E_xxx (error code)	Vendor specific error code.

Table 126: ServerCanUnloadNow error codes

Default behavior
Return S_OK.

2.2.7. ConditionCanBeEnabled

VOID **ConditionsCanBeEnabled** (DWORD *pdwResult)

Description
This function will be called whenever any condition should be enabled or disabled.

Return Code	Description
S_OK	The operation succeeded.
S_xxx (success code) E_xxx (error code)	Vendor specific error code.

Table 127: ConditionsCanBeEnabled error codes

Default behavior
Return S_OK.

2.2.8. OnEnableCondition

```
VOID      OnEnableCondition (
                              [IN] BSTR       bstrSourceName,
                              [IN] BSTR       bstrConditionName,
                              [IN] BOOL       bEnable,
                              [OUT]DWORD  *pdwResult)
```

Description

This function will be called whenever a condition is enabled or disabled.

Parameter	Description
szSourceName	Source for condition being enabled / disabled.
szConditionName	Condition being enabled / disabled.
bEnable	This flag indicates that the condition should be enabled or disabled.

Table 128: OnEnableCondition parameters

Return Code	Description
S_OK	The operation succeeded.
S_xxx (success code) E_xxx (error code)	Vendor specific error code.

Table 129: OnEnableCondition error codes

Default behavior

Return S_OK.

2.2.9. GetAttribCurrentValue

```
VOID      GetAttribCurrentValue (
                              [IN] DWORD      dwNumAttributes,
                              [OUT] VARIANT  *pVar,
                              [OUT] DWORD    *pdwResult)
```

Description

Return the current value of the passed attribute ID.

Parameter	Description

dwAttributeID	The ID of the attribute to get current value.
pVar	A variant to contain the retrieved value.

Table 130: GetAttribCurrentValue parameters

Return Code	Description
S_OK	The operation succeeded.
E_FAIL	The operation failed.
S_xxx (success code) E_xxx (error code)	Vendor specific error code.

Table 131: GetAttribCurrentValue error codes

Default behavior

Return S_OK, with Variant Type equal to VT_EMPTY.

SDK CONFIGURATION

The DLL has tracing capabilities. Developer can record the DLL errors and debugging information (COM and OPC messages) in a log file named “AeSdk-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 start-up where the server executable is located. The toolkit incorporates a configuration file “ConfigOPCAEServerSDK.ini” which includes several logging and configuration parameters. These parameters all have default settings and can be changed at start-up by editing the configuration file.

To change this file:

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

LogSetting	Description	Default Value
LogFileMaxSize	The maximum log file size, in bytes. Once this size is reached during runtime, the log file will be overwritten.	2097152
CreateNew	True to create a new event log or to append the old log, by default it's true.	True
LogLevel	The log level. Possible Values are: 0: Lowest log level used to control the application execution 1: All critical error messages are logged. 2: All errors are logged. 3: All information is logged. 4: For Debug information. The higher the log level, the more information is recorded. We recommend using the control level for a better performance of the server.	Control
ArchiveLastLog	TRUE: Old file is copied to an intermediate file with incremental extension, before being overwritten. FALSE: Any pre-existing log file is erased and overwritten at start-up.	False
LogFileName	Used to set the log file name	AeSdk-LogEvent

FileLogConfiguration	Description	Default Value
AutoAppend	Set to true to continue writing log messages in the existed log file or to false to create a new file.	True
BufferSize	The maximum number of messages to be stored in the runtime memory before launching a write action in the hard disk. The specified value must be greater than 100.	100
FilePerDay	True: Generate log file per day	True
ApplicationSetting	**Description**	**Default Value**
Timer	Allows you to set a time interval to peridically execute the OPC AE Server methods	2000

Table 132: ConfigOPCAEServerSDK.ini Description

3. Save the file and restart your application for the new settings to take effect.

Configuration File

```
[LogSetting]
LogFileMaxSize=2097152
CreateNew=True
LogLevel=0
ArchiveLastLog=False
LogFileName=AeSdk-LogEvent
[FileLogConfiguration]
AutoAppend=True
BufferSize=100
FilePerDay=True
[ApplicationSetting]
Timer=2000
```

IOPCAESIMULATOR MFC EXAMPLE

1. Introduction

In order to simplify the use of this ToolKit, the distribution contains an MFC sample for an OPC AE server. This sample implements an OPC server that simulates a set of alarms. In this chapter, we try to describe its main features that help developers to initiate their application server. This OPC AE server for simulation is compliance test certified for AE 1.10.

2. Executing the sample

2.1. 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.OPCAE.Simulation**" 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**.

2.2. Server features

2.2.1. Overview

Integration Objects' OPC Alarms and Events Simulator is a sample AE Server based on the **Integration Objects' OPC AE Server ToolKit.** This is a prototype for OPC AE servers that OPC server developers can use to generate their own server's applications.

When this sample is running, it is launched as a small icon in the Tool Tray at the right-hand side of the Task Bar. After a right click on this icon, you will get the following menu:

Figure 3: IOPCAESimulator menu

- Register: to register the OPC Server.
- Un-Register: to remove OPC server entries from the Windows Registry.
- Shutdown: to request all connected clients to disconnect and close the server.

2.2.2. CSV Simulation file description

With the server, you find a **CSV simulation file** used to demonstrate how the server can generate different event types.

An example of the content of this file is shown at the following figure:

	A	B	C	D	E	F	G	H
1	EventType 1=OPC_S	SourceName	AreaName	ConditionNam	SubconditionName	ActiveState	Message	EventCategoryID
2	4	FIC1001	Boiler1:makeup1	PVLEVEL	HIHI	1	HIHI Alarm	
3	4	FIC1001	Boiler1:makeup1	PVLEVEL	HI	1	HI Alarm	
4	4	FIC1001	Boiler1:makeup1	PVLEVEL	HI	0	Condition Normal	
5	4	FIC1001	Boiler1:makeup1	PVLEVEL	LO	1	LO Alarm	
6	4	FIC1001	Boiler1:makeup1	PVLEVEL	LOLO	1	LOLO Alarm	
7	4	FIC1002	Boiler1:makeup2	DEVIATION	DEVIATION	1	Deviation Alarm	
8	4	FIC1002	Boiler1:makeup2	DEVIATION	DEVIATION	0	Condition Normal	
9	4	FIC1003	Water1:makeup3	PVLEVEL	HIHI	1	HIHI Alarm	
10	4	FIC1003	Water1:makeup3	PVLEVEL	HI	1	HI Alarm	
11	4	FIC1003	Water1:makeup3	PVLEVEL	HI	0	Condition Normal	
12	4	FIC1003	Water1:makeup3	PVLEVEL	LO	1	LO Alarm	
13	4	FIC1003	Water1:makeup3	PVLEVEL	LOLO	1	LOLO Alarm	
14	4	FIC1004	Water1:makeup4	DEVIATION	DEVIATION	1	Deviation Alarm	
15	4	FIC1004	Water1:makeup4	DEVIATION	DEVIATION	0	Condition Normal	
16	1	System_Event		NA	NA	0	Simple Event	
17	2	Tracking_EVENT		NA	NA	0	Setpoint changed Tracking Event	
18								

Figure 4: CSV file

This file is composed of the following fields:

- EventType: The type of the event to be generated. This field can contain one of these values:
  - 1 for Simple Event.
  - 2 for Tracking Event.
  - 4 for Condition Event.
- SourceName: The event source name to be generated.
- AreaName: The area name related to the event's source.
- ConditionName: The condition name related to the event to generate.

- SubConditionName: The current sub-condition name for multi-state conditions. For a single-state condition, this contains the condition name.
- ActiveState: The event state. This field can be 0 for inactive state and 1 for active.
- Message: A text describing the event to be generated.
- EventCategoryID: The event Category ID related to the event.
- EventCategoryName: The EventCategoryID associated name.
- Severity: A value between 0 and 1000 describing the severity level of the event being generated.
- QualityValue: The quality to be associated to the event (exp 192 for GOOD).
- AckRequired: This flag indicates that the related condition requires acknowledgment of this event. It can be 0 (not required) or 1 (required).
- ActorID: It presents the actor ID for the event notification for tracking events.

3. Application architecture

In this section, we try to give you an idea about the main implemented classes in this sample OPC server.

3.1. CIOAESimServerApp

The main methods are:

1. InitInstance ( ): contains the main program. In this method, you find the initialization of the ToolKit, the building of the Area Space and the Event Space and the initialization of the server cache.
2. ExitInstance ( ): here we free all allocated resources in the server application. The Uninitialize ( ) is also called to uninitialize the COM library and free resources allocated inside the ToolKit.

3.2. Mainframe

This class contains the following methods:

1. OnCreateClient: Initialize a timer that consists the amount of time the server parse the simulation file to fire a new notification.
2. OnTimer: This function is called every simulation period to read the CSV simulation file and produce the events.
3. Generate_SimpleEvent: This function is used to notify client with simple event occurrence.
4. Generate_TrackingEvent: This function is used to notify clients with tracking event occurrence.
5. Generate_ConditionEvent: This function is used to notify clients with condition event occurrence.

4. Steps to build your own OPC AE Server

You find below a short description of different steps required to build your own OPC AE Server using the Integration Objects' OPC AE Server Toolkit.

4.1. Initialization of the toolkit

To initialize the toolkit you have to:

- Launch GUIDGen.exe from C:\Program Files\Microsoft VisualStudio\Common\Tools\.
- Generate a new CLSID using this utility,

Sample CLSID

```
// {785D6306-F436-4aac-A298-43393DC4AA9A}
DEFINE_GUID (CLSID_OPCAESampleServer, 0x785d6306, 0xf436, 0x4aac, 0xa2, 0x98, 0x43, 0x39, 0x3d, 0xc4, 0xaa, 0x9a);
```

- Add the following code to the main of your application:

```
IO Initialize (
              NULL,
              CLSID  OPCAESampleServer,
              pszServerName,
              pszVendorInfo,
              1,
              0,
              0,
              (WCHAR *)":",
              0
             )
```

4.2. Registration of the server.

To register the OPC Server, you can add the following call just after initializing the toolkit.

```
IO RegisterServer (ExePath);
```

With ExePath = the path of the executable. If you set ExePath to null string the toolkit will try to find this path.

4.3. Initialization of the Area Space

To initialize the server Area Space, first you should prepare the list of area and source node to be added. Then, for each area, you call AddAreaNode method:

```
IO AddAreaNode (AreaName);
```

For each source call the method:

```
IO AddSourceNode (SourceName).
```

You can initialize the source and the related area at the same by calling AddSourceNode (SourcePath).

4.4. Initialization of the Event Space

To initialize the server Event Space, you should:

1. Begin by initializing the list of Event Category for each Event Type.

```
IO AddCategory (dwEventCategory, pszEventCategory, dwEventType);
```

2. For each Event Category, initialize the list of Event Attributes by calling:

```
IO AddAttribute (dwEventAttribute, szAttrib, VT R8, dwEventCategory);
```

3. For each Event Category, initialize the list of conditions by calling:

```
IO AddCondition (dwEventCategory, szCond);
```

4. Finally, for each condition, initialize the list of sub-conditions by calling:

```
IO AddSubCondition (szCond, szSubCond, szSubCondDesc);
```

4.5. Initialization of the Server Cache

Before starting the generation of condition events, you should initialize the list of available conditions. To do, you can use:

```
IO AddSourceConditionToCache (szSource, szCond);
```

After that, you add the sub-condition szSubCond related to the couple (szSource, szCond) to the server cache as follows:

```
IO AddSubConditionToCache (szSource, szCond, szSubCond, 500,"Severity
500");
```

4.6. Initialization of the Callbacks.

After initializing all server configurations, you have to implement the callback function that will be called by the toolkit when a communication with real equipment or a specific internal action is required.

You find below the implementation of the default behavior that you can use to initialize the callback functions:

```
HRESULT CALLBACK EXPORT OnClientConnect()
{
    return S_OK;
}
HRESULT CALLBACK EXPORT OnClientDisconnect()
{
    return S_OK;
}
HRESULT CALLBACK EXPORT ServerCanUnloadNow()
{
    return S_OK;
}
HRESULT CALLBACK EXPORT GetAttributeCurrentValue(
                                                DWORD       adwAttrID,
                                                VARIANT*    pValue)
{
    VariantInit(pValue);
    pValue->vt          =   VT_EMPTY;
    return S_OK;
}
LPWSTR CALLBACK EXPORT GetErrorString(DWORD dwError)
{
    USES_CONVERSION;
    return T2W("Vendor Specific Error");
}
HRESULT CALLBACK EXPORT OnAckCondition(
                                       LPWSTR pszSourceName,
                                       LPWSTR pszConditionName
                                       )
{
    return S_OK;
}

HRESULT CALLBACK EXPORT ServerCanUnload()
{
    return S_OK;
}
HRESULT CALLBACK EXPORT TranslateToItemID(
                                          DWORD     dwAttrID,
                                          LPWSTR    *ppszAttrItemID,
                                          LPWSTR    *ppszNodeName,
                                          LPWSTR    *ppCLSID
                                          )
{
    USES_CONVERSION;
    *ppszAttrItemID =   T2W("");
    *ppszNodeName   =   T2W("");
    *ppCLSID        =   T2W("00000000-0000-0000-0000-000000000000");
    return S_OK;
}
HRESULT CALLBACK EXPORT ConditionCanBeEnabled()
{
    return S_OK;
}
HRESULT CALLBACK EXPORT OnEnableCondition(LPCTSTR   szSourceName,
                                          LPCTSTR   szConditionName,
                                          BOOL      bEnable)
{
    return S_OK;
}
```

To initialize the list of callback functions, you should call this method:

```
IO SetCallBackObjects(
     &OnClientConnect,
      &OnClientDisconnect,
      &GetErrorString,
      &OnAckCondition,
      &ServerCanUnloadNow,
      &TranslateToItemID,
      &ConditionCanBeEnabled,
      &OnEnableCondition,
      &GetAttributeCurrentValue
      )
```

4.7. Setting Server Period Time

To set the max amount of time the server sends a new notification:

```
IO_SetServerPeriodTime (2000) 'for 2000 ms
```

If this method is not called then the toolkit uses the default values (1000 ms).

4.8. Setting server status

Finally, to complete the toolkit initialization, you should set the status of the server:

```
IO SetStatus(OPCAE STATUS RUNNING)
```

4.9. Firing events.

At this step, you can start the generation of the events. Then, this toolkit offers four methods:

- IO_GenerateSimpleEvent (szSource,
  dwEventCategory,
  dwSeverity,
  szMessage,
  ftTime) to fire simple event.
- IO_GenerateTrackingEvent (szSource,
  dwEventCategory,
  dwSeverity,
  szMessage,
  szActorID,
  ftTime) to fire tracking event.

And to manage condition event you can use:

- IO_ActivateCondition (szSource, _
  szConditionName, _
  szSubconditionName,
  szMessage,
  bAckRequired,
  wQuality,
  ftTime) to active condition event.
- IO_DeactivateCondition (szSource,
  szConditionName,
  szMessage,
  bAckRequired,
  wQuality,
  ftTime) to deactivate condition event.

APPENDIX A

What is OPC?

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. There are currently seven standards specifications completed or in development.

OPC Alarms and Events specifications

The OPC AE specification is a standard of interface which has the mission to develop and standard set of interface for managing alarm and event notifications. The Alarms and events standard include *(but are not limited to)* Process alarms, Operator manual action requests generated by the system, Informational messages such as Batch Completion, Tracking/Auditing messages such as a change in a setpoint or tuning parameter by an operator.

EventType

An *event* is a detectable occurrence which is of significance to the OPC Event Server, the device it represents, and it's OPC Clients. An event has no direct representation within the OPC model. Rather, its occurrence is made known via an *Event Notification*. Event Notifications are represented by objects of class OPCEventNotification, which are described in the following section. (OPCEventNotifications are not COM objects.)

There are three types of events:

1. *Condition-related* events are associated with OPCConditions, and represent transitions into or out of the states represented by OPCConditions and OPCSubConditions. An example is the tag FIC101 transitioning into the LevelAlarm condition and HighAlarm sub-condition.
2. *Tracking-related* events are not associated with conditions, but represent occurrences which involve the interaction of an OPC Client with a "target" object within the OPC Event Server. An example of such an event is a control change in which the operator, (the OPC Client), changes the set point of tag FIC101 (the "target").
3. *Simple* events are all events other than the above. An example of a simple event is a component failure within the system/device represented by the OPC Event Server.

Filter Criteria

The client can set filter criteria for each subscription. The following filter criteria are described by the OPCAE specification. Servers can choose to support any combination (including none) of these filter capabilities.

- **Filter by Event Type**: This allows the client to choose what classes of events (simple, tracking, condition) to deliver on this subscription.
- **Filter by Category:** This allows the client to choose what combination of server specific event categories to deliver on this subscription. The client can ask the server for a list of available categories.
- **Filter by Severity:** This allows the client to choose a range of severity's to deliver on this subscription.
- **Filter by Area:** This allows the client to choose what combination of server specific areas to deliver on this subscription. The client can ask the server for a list of available areas.
- **Filter by Source Name**: This allows the client to specify a set of Source Condition names to deliver on this subscription.

Area Space

The AE area space presents the list of event sources that can generate an event notification.

Event Space

The event space (or filter space) of an OPC AE server presents a logical structuring of the events. This structure includes the following objects:

- **Event Type**: The OPC A&E specification defines three event types: ***simple***, ***tracking***, and ***condition*** (including alarms). A server can generate any or all of these event types. A simple event server is one that only generates simple (and possibly tracking) events. More advanced servers will generate all three types.
- **Event Category**: EventCategories define groupings of events supported by an OPC Event server. Examples of event categories might include “Process Events”, “System Events”, or “Batch Events”. Event categories may be defined for all event types, i.e. Simple, Tracking, and Condition-Related. However, a particular event category can include events of only one type. A given Source (e.g. “System” or “FIC101”) may generate events for multiple event categories. Names of event categories must be unique within the event server. The definition of event categories is server specific and is outside the scope of the OPC AE specification.
- **Event Attribute**: In addition to the standard attributes described above, implementers of OPC Event Servers may choose to provide additional attributes with event notifications.
- **Condition**: A *condition* is a named state of the OPC Event Server, or of one of its contained OPC Items (if it is also an OPC Data Access Server), which is of interest to its OPC Clients. Conditions may be *single state*, or *multi-state*. A multi-state condition is one whose state encompasses multiple “ranges” or sub-states which are of interest. For

example, a “LevelAlarm” condition may have multiple sub-states including “HighAlarm” and “HighHighAlarm”.

- **SubCondition**: Is a sub-state of a multi-state condition. A single state condition has only one sub-state of interest, and thus has only one sub-condition associated with it.

AE Server Cache

The server cache presents a collection that reflects the latest state of the condition event as well as the change-mask flag and the new-state.

Event Notification

To receive event notifications, clients must set up a subscription with the OPC A&E server. The client can then change the state of that subscription or set filter criteria so that only notifications of interest will sent to them via that subscription.

GLOSSARY

C

CLSID
A CLSID is a globally unique identifier that identifies a COM class object. If your server or container allows linking to its embedded objects, you need to register a CLSID for each supported class of objects.

COM

Component **O**bject **M**odel, it is a specification for writing reusable software components. COM is an infrastructure that allows objects to communicate between processes and computers.

D

DCOM
Distributed **C**omponent **O**bject **M**odel, it is a protocol that enables software components to communicate directly over a network in a reliable, secure, and efficient manner.

O

OPC
Ole for **P**rocess **C**ontrol, OPC 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.

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

Offices
- Houston, USA: +1 713 609 9208
- Genova, Italy: +39 34 75 83 93 47
- Tunis, Tunisia: +216 71 861 803

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 us on the Internet:

Online
- www.integrationobjects.com