Integration Objects'

OPC HDA Client Toolkit for C++

OPC HDA Client Toolkit
Version 1.1 Rev.0

USER GUIDE

OPC Compatibility
OPC Historical Data Access 1.20
OPC Historical Data Access 1.10
OPC Historical Data Access 1.00

Integration Objects' OPCHDA Client Toolkit User's Guide Version 1.1 Rev. 0
Published in July 2016

TABLE OF CONTENT

TABLE OF FIGURES

TABLE OF TABLES

PREFACE

About This User Guide

This guide presents the compatibility of the Integration Objects' OPC HDA Client Toolkit with C++ environment and describes the functions exported by its API (Application Programmer Interface).

Target Audience

This reference manual is intended to C++ developers of OPC Historical Data Access client applications. It assumes a working knowledge of programming with the C++language and of the OPC HDA specifications.

Related documentation

As you use this user's guide, you may also find useful the following specification:

- OPC Historical Data Access Custom Interface Standard Version 1.20
- OPC Common Custom Interface Standard Version 1.0

Customer Support Service

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

INTRODUCTION

1. Overview

Integration Objects' OPC HDA Client Toolkit is a Windows DLL that implements all required and optional OPC Historical Data Access interfaces.

Using this toolkit, C++custom applications will be able to access historical data from any OPC HDA server without having to be concerned with the details of the OPC HDA standard.

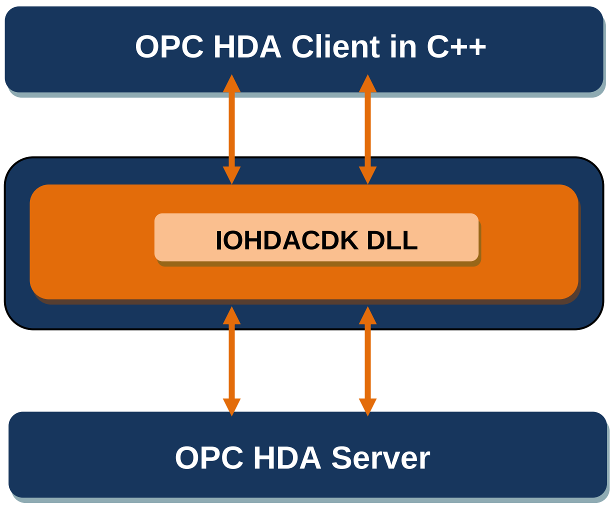


Figure 1: OPC HDA Client Toolkit Overview

2. Supported Interfaces

The supported interfaces by the OPC HDA Client Toolkit are listed in the table below:

Interfaces	V. 1.1/1.2	Implemented
OPCHDAServer		
IOPCCommon	Required	Yes
IOPCHDA_Server	Required	Yes
IOPCHDA_SyncRead	Required	Yes
IOPCHDA_SyncUpdate	Optional	Yes
IOPCHDA_SyncAnnotations	Optional	Yes
IOPCHDA_AsyncRead	Optional	Yes
IOPCHDA_AsyncUpdate	Optional	Yes
IOPCHDA_AsyncAnnotations	Optional	Yes
IOPCHDA_Playback	Optional	Yes
IConnectionPointContainer	Required if any Async interface is supported	Yes
OPCHDABrowser		
IOPCHDA_Browser	Optional	Yes

Table 1: Supported Interfaces

3. Operating Systems Compatibility

The applications generated using the OPC HDA Client Toolkit were tested under the following operating systems:

- Windows Server 2003
- Windows XP
- Windows Server 2008
- Windows 7
- Windows Server 2012
- Windows 10

4. OPC Compatibility

- OPC Historical Data Access 1.20
- OPC Historical Data Access 1.10
- OPC Historical Data Access 1.00

GETTING STARTED

1. Pre-installation Considerations

In order to run properly any OPC HDA Client application that uses the OPC HDA Client Toolkit, you need to install the OPC core components in the target system.

The OPC core components 3.00 consist of all shared OPC modules including the DCOM proxy/stub libraries, the OPC Server Enumerator, .NET wrappers, etc.

You can download the OPC Core Components 3.00 from the OPC Foundation site.

2. Files Distribution

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

Folder	Description
Bin	**IOHDACDK.DLL**: Release Toolkit DLL. **License.dll**
Configuration files	**HDAClientToolkit-CfgFile.ini**: contains configurable parameters for the log capabilities.
Include	**Callbacks.bas**: Includes the VB callbacks implementation. It can be used by the VB developer to implement the VB application interacting with the toolkit. **Wrapper.bas**: Includes the wrappers for the Toolkit exported functions to be used by VB applications.
OPC Sample Projects/VS2010 Sample	This distribution contains: ▪ An advanced VB OPC HDA client implemented using this VB wrapper. ▪ A set of VB samples that simulate the most important wrapped HDA functionalities. ▪ A VB Chart sample. ▪ An OPC HDA client sample implemented using the C++ toolkit.
OPC Sample Demos/VS2010	The VB distribution contains: ▪ **VB-Sample-HDA-Client.exe**: an executable of an OPC HDA VB client for simulation. ▪ **MSChart-Sample.exe:** an example of HDA Chart client based

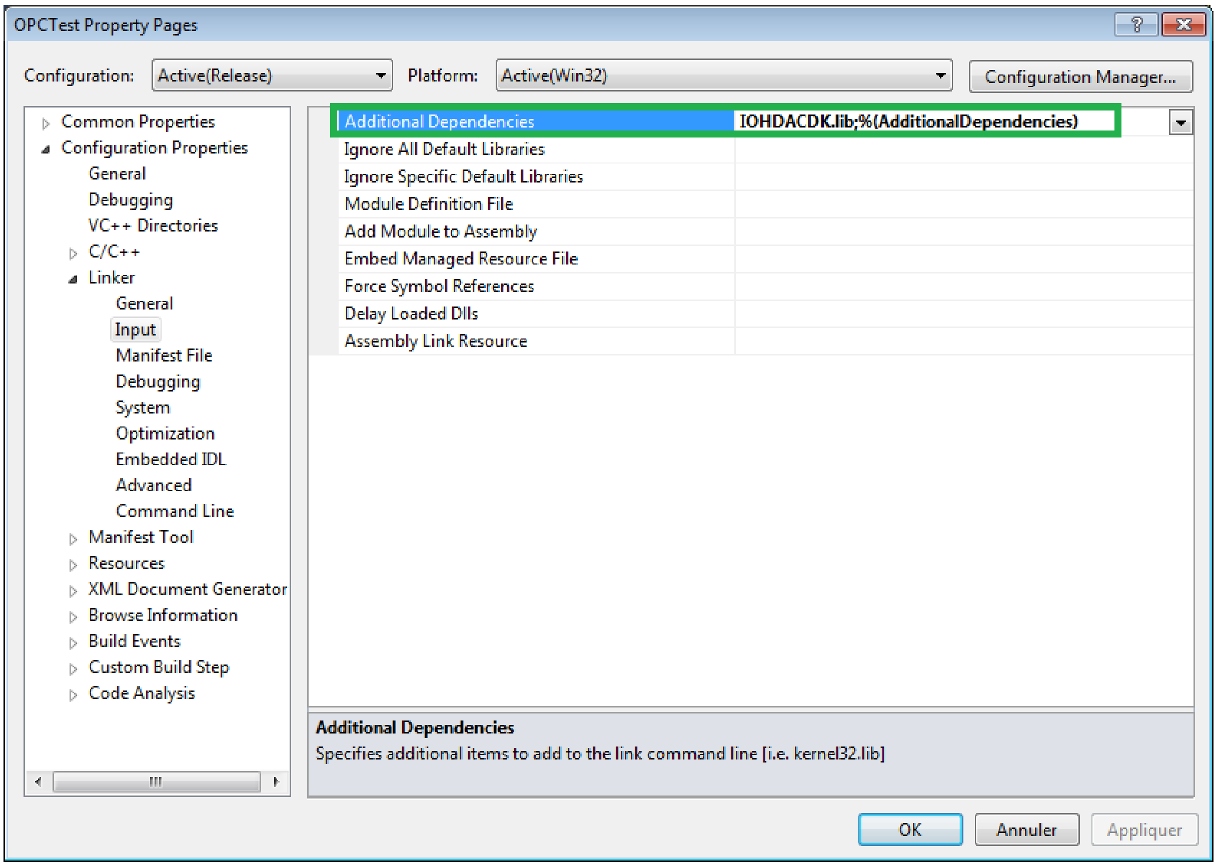
	on OPC HDA Client Toolkit. ▪ A set of executables that simulate the most important wrapped HDA functionalities. ▪ **OPCHDASampleTest.exe**: an executable of an OPC HDA Client example programmed in C++.
Documents/OPC	**OPC Common 1.0 Specification.pdf:** The OPC Common 1.0 specification. **OPC HDA 1.20 Specification.pdf:** The OPC Historical Data Access 1.20 specification.
Documents	**OPC HDA Client Toolkit for VB User's Guide.pdf**: The OPC HDA Client Toolkit for VB user guide **OPCHDA Client Toolkit for C++User's Guide.pdf**: The OPC HDA Client Toolkit for C++ User guide.

Table 2: Distributed Files

3. Compiling and Linking Applications

To successfully create your C++ HDA client application, you have to:

- Start **Visual Studio 2010**
- Create a **New Project➔Visual C++➔Win32 Console Application**
- Add **IOHDACDK.lib** to the current project folder
- Link **IOHDACDK.lib** to the created C++ sample (From the menu, click **Project➔Properties➔Linker➔Input** then Add **IOHDACDK.lib** to the **Additional Dependencies**).


OPCTest Property Pages

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

Additional Dependencies	IOHDACDK.lib;%(AdditionalDependencies)
Ignore All Default Libraries	
Ignore Specific Default Libraries	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded Dlls	
Assembly Link Resource	

Additional Dependencies
Specifies additional items to add to the link command line [i.e. kernel32.lib]

- Add the **IOHDACDK.DLL** delivered with the current package to the current project folder.

OPC HDA SPECIFICATION – TERMS & CONCEPTS

Attribute – An additional qualifier that a particular item may have associated with it. For example, an “Item Value” attribute would probably have the following attributes associated with it: “Data Type” (VT_R4), “Stepped” (0) and “Archiving”(1)., i.e. the “Item Value” returns a 4-byte real number, the value can be displayed as interpolated (sloped line) and data is being archived .

Aggregates – Methods that summarize data values. Common aggregates include Averages, Minimum and Maximum. These aggregates are performed during data retrieval.

Annotations - An operator or user enters comment that is associated with an item, usually at a given instance in time. There does not have to be a value stored at that time.

Bounding Values - Bounding values are required by clients to determine the entry and the exit points when requesting raw data over a time range. If a raw data value exists at the entry or the exit point, it is considered the bounding value even though it is part of the data request. If no raw data value exists at the entry or exit point, the next data point outside of the range is considered the bounding value.

Interpolated Data – Data that is derived from the data in the archive, but for which there is no stored value. This may be linearly derived from two stored data points on either side of the requested timestamp, or it may be extrapolated from the data in the archive by a more complex method.

Item Handles – The Item Handle can be either a client or a server value. It is used by the owner to speed access to the items. Its data type is OPCHandle (DWORD).

It is expected that a client will assign a unique value to the client handle if it intends to use any of the asynchronous functions of the OPC HDA interfaces. However, the server should not make any assumptions about the client handle and the client should not make any assumptions about the server handle. Uniqueness of the item handles are implementation dependent.

Item ID - The character string that is a unique reference to a data item in the server address space.

Modified values – A modified value is a value that has been changed after it was stored in the historian. A lab data entry value is not a modified value, but if a user corrects a lab value, the original one would be considered as a modified value, and would be returned on a request for modified values. All methods on all interfaces are assumed to be based upon the current, or most recent, value for the specified Item at the specified timestamp. Requests for modified values are used to access values that have been superseded.

Properties – Within the Automation interface, properties are attributes of the history server that indicate how it operates.

Raw Data - Data that is stored within the historian. The data may be compressed or may be collected for the item depending on the historian and the storage rules invoked when the item values were saved.

Start Time / End Time – The time bounding a history read request, which define the time domain of the request.

For all requests, a value falling at the end of the time domain is not included in the domain, so that requests made for successive, contiguous time domains will include every value in the archive, exactly once. See the examples section below.

Time Domain - The interval of time covered by a particular request, or by a particular response. In general, if the start time is earlier than the end time, the time domain is considered to begin at the start time and to end just before the end time; if the end time is earlier than the start time, the time domain still begins at the start time and ends just before the end time, with time "running backward" for the particular request and response. In both cases, any value, which falls exactly at the end time of the time domain, is not included in the domain. See the examples section below.

Note that all timestamps which can legally be represented in a FILETIME are valid timestamps, and the server may not return E_INVALIDARG due to the timestamp being outside of the range for which the server has data. Servers are expected to handle out-of-bounds timestamps gracefully, and return the proper error codes and value to clients, such as OPC_S_NODATA or OPCHDA_NOBOUND.

Synchronous Interfaces

Synchronous operations require that the client software wait until the server will fulfill the request and will return the data. Synchronous operations, that may require significant time for the server to fulfill, have corresponding asynchronous operations that may be cancelled.

Asynchronous Interfaces

Asynchronous operations allow a client to send a request to a server without waiting for it to fill the request and return the data. Each operation has an associated transaction ID (created by the client) which is returned with the data during the callback to the client. Asynchronous methods for OPC HDA servers are implemented using IConnectionPoint. This allows the client to establish different callbacks to handle different types of data transfers. While there is some information given on IConnectionPoint later in this document, it is advisable to read the Microsoft documentation.

It is anticipated that some servers will not support all of the asynchronous interfaces specified here. However, all servers must support IConnectPointContainer, and any server which supports any asynchronous interface must support the IOPCHDA_DataCallback ConnectionPoint. If a server does not support a specific interface, it does not need to implement the means to send a callback to the matching callback routine in the client's IOPCHDA_DataCallback object.

The client provides a TransactionID to differentiate one call from another, if it needs to do so. Since the response to an async call can actually arrive before the call completes, the client should store this TransactionID before making the call, so the callback routine will have access to the TransactionID. Conversely, the CancelID is generated by the server, and is used to cancel async requests if the client wishes to do so.

Asynchronous vs.Synchronous Interfaces

Here are two ways for a client application to collect historical data from a server:

- It can perform a synchronous read (simple and reasonably efficient). This may be appropriate for simple clients that are reading relatively small amounts of data and where maximum efficiency is not a concern. A client that operates in this way is willing to block and to wait for the results. When a large amount of data is requested, this could require some time. This method is appropriate for reports or other non-interactive reads, but would be very poor for an interactive display.
- It can ‘subscribe’ to data using the Async methods, which is more complex but very efficient. This is the recommended behavior for interactive clients because it will minimize display lockups. The client would be free to process other interactions while waiting for the data to return.

USING OPC HDA CLIENT TOOLKIT

1. Initialization of the API

After the DLL initialization, the client application has the responsibility to properly initialize the COM library. The Toolkit exports an API function named **io_initialize** that performs the basic initialization functions. **Io_initialize** initializes COM to run in a multi-threaded environment and to use the default security.

```
HRESULT __stdcall  io_initialize()
```

Return Values

Return Code	Description
IO_S_OK	The function was successful
IO_S_FALSE	COM library already initialized
IO_E_COM_CHANGED_MODE	A previous call to CoInitializeEx specified a different concurrency model for the calling thread
IO_E_COM_INVALIDARG	An invalid argument was passed
IO_E_COM_OUTOFMEMORY	The memory could not be allocated
IO_E_COM_UNEXPECTED	An unexpected error occurred
IO_E_COM_UNKNOWN	An unknown error occurred

Table 3: io_initialize - Return Values

Example:

```
int res=io_initialize();
```

The operation is also successful if the return code is IO_E_COM_CHANGED_MODE

2. OPC Server Management

2.1. Connect To Server

```
HRESULT __stdcall OPCHDA_ConnectToHDAServer(constchar *_pszProgID,
                                            constchar *_pszMachineID,
                                            int        *_piServerHandle)
```

This function establishes a connection to the server identified by _pszProgIDand located in the machine _pszMachineID. Passing a NULL string as the _pszMachineIDparameter identifies the local machine.

Parameters

Parameter	Description
*_pszProgID	The server progID
*_pszMachineID	The server node name
_piServerHandle	If the call succeeds, this parameter will contain a unique identifier for the connection. It should be passed in any further call to the server

Table 4: OPCHDA_ConnectToHDAServer- Parameters

Return Values

Return Code	Description
IO_S_OK	Successful connection to server
IO_E_INVALIDARG	CoCreateInstanceEx: failed, E_INVALIDARG error code: invalid NodeName argument
IO_E_NOINTERFACE	CoCreateInstanceEx: failed, can't retrieve the IUnknown interface
IO_RPC_S_SERVER_UNAVAILABLE	The server is unavailable
IO_E_ACCESSDENIED	Access is denied
IO_REGDB_E_CLASSNOTREG	The component specified is not registered.
IO_REGDB_E_READREGDB	Error when reading from the registry
IO_CO_E_DLLNOTFOUND	In-process DLL or handler DLL not found.

IO_CO_E_APPNOTFOUND	EXE not found.
IO_CO_E_ERRORINDLL	EXE has error in image.
IO_CO_E_APPDIDNTREG	EXE was launched, but it did not register the class object.
IO_E_UNEXPECTED	An unexpected error occurred.

Table 5: OPCHDA_ConnectToHDAServer- Return Values

Example:

```
int Shandle,res;
printf("Enter the OPC Server PROGID\n");
char ProgID[200];
fflush(stdin);
gets(ProgID);
string NodeName= "localhost";
char *Node_Name= (char*)NodeName.c_str();
constchar * progID=(constchar *) ProgID;
constchar* NodeN=(constchar*)Node_Name;
res =OPCHDA_ConnectToHDAServer(progID, NodeN, &Shandle);
```

2.2. Disconnect from Server

To disconnect from the server, the client may use a method called **OPCHDA_DisconnectFromHDAServer**, providing the ServerHandle returned in OPCHDA_DisconnectFromHDAServer. If the disconnection succeeds, the returned code will be IO_S_OK. Otherwise, it will be IO_E_FAIL.

```
HRESULT __stdcall OPCHDA_DisconnectFromHDAServer (int _iServerHandle)
```

Parameters

Parameter	Description
_iServerHandle	The handle of the server's connection.

Table 6: OPCHDA_DisconnectFromHDAServer- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_FAIL	The function failed.

Table 7: OPCHDA_DisconnectFromHDAServer- Return Values

Example:

```
int res = OPCHDA_DisconnectFromHDAServer(Shandle);
```

2.3. Connection to the HDA Interfaces

Once the connection to the OPC HDA Server is correctly established, the user must connect to the different HDA interfaces offered by the server to access its functionalities.

To do that, the API offers a connection function for each HDA interface. The user must only give the ServerHandle that represents the connection ID between the API and the server. An error code will be returned to indicate the success or failure of the function call.

```
HRESULT __stdcall OPCHDA_ConnectToHDASynchReadInterface (int _iServerHandle)
HRESULT __stdcall OPCHDA_ConnectToHDASynchUpdateInterface(int_iServerHandle)
HRESULT __stdcall OPCHDA_ConnectToHDASynchAnnotationsInterface(int  _iServerHandle)
HRESULT __stdcall OPCHDA_ConnectToHDAASynchReadInterface(int_iServerHandle)
HRESULT __stdcall OPCHDA_ConnectToHDAASynchUpdateInterface(int_iServerHandle)
HRESULT __stdcall OPCHDA_ConnectToHDAAsynchAnnotationsInterface(int_iServerHandle)
HRESULT __stdcall OPCHDA_ConnectToHDAPlayBackInterface(int_iServerHandle)
```

Parameters

Parameter	Description
iServerHandle	The handle of the server's connection.

Table 8: Connection to the HDA Interfaces - Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_ NOINTERFACE	The server does not implement that interface.
IO_E_ FAIL	Connection failed for unknown reason.

Table 9: Connection to the HDA Interfaces - Return Values

Example:

```
Int res=OPCHDA_ConnectToHDASynchReadInterface(Shandle);
res=OPCHDA_ConnectToHDASynchUpdateInterface(Shandle);
res=OPCHDA_ConnectToHDASynchAnnotationsInterface(Shandle);
res=OPCHDA_ConnectToHDAASynchReadInterface(Shandle);
res=OPCHDA_ConnectToHDAASynchUpdateInterface(Shandle);
res=OPCHDA_ConnectToHDAAsynchAnnotationsInterface(Shandle);
```

The API offers a method to check if an interface is implemented in the server. This can be used before calling the interface's connection method.
The user must specify the ServerHandle and the interface name. A result will be returned by the server to indicate the state of the interface.

Here is the list of all required and optional HDA interfaces.
Any OPC HDA server should support:

- IOPCCommon
- IOPCHDA_Server
- IOPCHDA_SyncRead

And optionally implement:

- IOPCHDA_SyncUpdate
- IOPCHDA_SyncAnnotations
- IOPCHDA_AsyncRead
- IOPCHDA_AsyncUpdate
- IOPCHDA_AsyncAnnotations
- IOPCHDA_Playback

```
HRESULT __stdcall OPCHDA_CheckHDAInterfaceStatus (int _iServerHandle,
                                                 Const char *_pszInterfaceID)
```

Parameters

Parameter	Description
_iServerHandle	The handle of the server's connection.
*_pszInterfaceID	The interface name.

Table 10: OPCHDA_CheckHDAInterfaceStatus - Parameters

Return Values

Return Code	Description
IO_S_OK	The server implements this interface.
IO_E_ NOINTERFACE	The server does not implement that interface.
IO_E_ FAIL	Connection failed for unknown reason.

Table 11: OPCHDA_CheckHDAInterfaceStatus - Return Values

2.4. Subscription to Callbacks Functions

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

According to the OPC HDA specification, the client must provide 10 types of callback function:

- OnReadComplete
- OnDataChange
- OnReadModifiedComplete
- OnReadAttributeComplete
- OnUpdateComplete
- OnReadAnnotations
- OnInsertAnnotations
- OnDataPlayBack
- OnCancelComplete
- OnShutdown

The API offers one function for subscribing to each callback. To activate a callback, the user must provide the server handle and the local function that will be invoked by the server every time it needs to notify the client.

```
HRESULT __stdcall    OPCHDA_initHDAReadCompleteCallBack(int    _iServerHandle,
       Void(*myReadComplete)(int,DWORD,HRESULT,DWORD,OPCHDA_ITEM*,HRESULT*)
)

HRESULT__stdcallOPCHDA_initHDADataChangeCallBack(int_iServerHandle,
       Void(*myDataChange)(int, DWORD,HRESULT,DWORD,OPCHDA_ITEM*,HRESULT*)
)

HRESULT__stdcallOPCHDA_initHDAReadModifiedCompleteCallBack   (int_iServerHandle,
       Void(*myReadModifiedComplete)(int,
       DWORD,HRESULT,DWORD,OPCHDA_MODIFIEDITEM*,HRESULT*)
)

HRESULT__stdcallOPCHDA_initHDAReadAttributeCompleteCallBack(int_iServerHandle,
       Void(*myReadAttributeComplete)(int,DWORD, HRESULT,OPCHANDLE,DWORD,
       OPCHDA_ATTRIBUTE*,HRESULT*)
)

HRESULT__stdcallOPCHDA_initHDACancelCallBack(int_iServerHandle,
       Void(*myCancel) (int, DWORD)
)

HRESULT__stdcallOPCHDA_initHDAUpdateCompleteCallBack(int     _iServerHandle,
       Void(*myUpdateComplete)(int, DWORD,HRESULT,DWORD,OPCHANDLE*,HRESULT*)
)

HRESULT__stdcallOPCHDA_initHDAReadAnnotationsCallBack(int    _iServerHandle,
       Void(*myReadAnnotations)(int, DWORD,HRESULT,DWORD,OPCHDA_ANNOTATION*,HRESULT*)
)

HRESULT__stdcallOPCHDA_initHDAInsertAnnotationsCallBack(int_iServerHandle,
       Void(*myInsertAnnotations) (int, DWORD,HRESULT,DWORD,OPCHANDLE*,HRESULT*)
)

HRESULT__stdcallOPCHDA_initHDADataPlaybackCallBack(int_iServerHandle,
       Void(*myPlayback)(int, DWORD,HRESULT,DWORD,OPCHDA_ITEM**,HRESULT*)
)
```

Parameters

Parameter	Description
myReadComplete	OnReadComplete procedure address
myDataChange	OnDataChange procedure address
myReadAttributeComplete	OnReadAttributeComplete procedure address
myUpdateComplete	OnUpdateComplete procedure address
myInsertAnnotations	OnInsertAnnotations procedure address
myCancel	OnCancelComplete procedure address
myPlayback	OnDataPlayBack procedure address
myReadModifiedComplete	OnReadModifiedCompleteprocedure address
myReadAnnotations	OnReadAnnotations procedure address

Table 12: Callbacks Functions - Parameters

Return Values

Return Code	Description
IO_S_OK	The callback function was defined correctly.
IO_E_ FAIL	The callback subscription failed.

Table 13: Callbacks Functions - Return Values

Example:

```
void myReadComplete(int      SrvHandle,
                    DWORD dwTransactionID,
                    HRESULT hrStatus,
                    DWORD dwNumItems,
                    OPCHDA_ITEM* objItemValues,
                    HRESULT* phrErrors){ }
```

```
int res=OPCHDA_initHDAReadCompleteCallBack(Shandle, myReadComplete);
```

3. OPC HDA Items

3.1. GetHDAItemID

This function provides a way to get fully qualified item identification. This is required since the browsing functions return only the components or tokens that make up an ItemID and do not return the delimiters used to separate those tokens.

```
HRESULT __stdcall OPCHDA_GetHDAItemHandles(int _iServerHandle,
                                             DWORD_dwCount,
                                            LPWSTR*_pszItemID,
                                            OPCHANDLE*_phClient,
                                            OPCHANDLE**_pphServer,
                                            HRESULT**_ppErrors)
```

Parameters

Parameter	Description
_iServerHandle	The handle of the server's connection
dwCount	The number of items to be handled
*_pszItemID	A pointer to the string, which contains the fully qualified ItemID
_phClient	The handle of the client to be associated with the item
*_pphServer	The returned handle of the server used to refer to this item
*_ppErrors	Returned errors

Table 14: OPCHDA_GetHDAItemHandles - Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_ FAIL	The function was unsuccessful.

Table 15: OPCHDA_GetHDAItemHandles- Return Values

Example:

```
printf("Enter the Item ID\n");
char ItemID[200];
fflush(stdin);
gets(ItemID);
wchar_t wtext[200];
mbstowcs(wtext, ItemID, strlen(ItemID)+1);
LPWSTR ptrItem = wtext;
_pszItemID[0]=ptrItem;
int res;
res=OPCHDA_GetHDAItemHandles(Shandle,1,_pszItemID,_phClient,&_pphServer,&_ppErrors);
```

3.2. GetHDAHistorianStatus

This function returns information on the current server status.

```
HRESULT __stdcall OPCHDA_GetHDAHistorianStatus (int _iServerHandle,
              OPCHDA_SERVERSTATUS  *_pwStatus,
              FILETIME            **_pftCurrentTime,
              FILETIME             **_pftStartTime,
              WORD                 *_pwMajorVersion,
              WORD                 *_pwMinorVersion,
              WORD                 *_pwBuildNumber,
              DWORD                *_pdwMaxReturnValues,
              LPWSTR               *_ppszStatusString,
              LPWSTR               *_ppszVendorInfo)
```

Parameters

Parameter	Description
_iServerHandle	The handle of the server's connection.
_pwStatus	The current status of the historian. See values defined below (OPCHDA_UP, OPCHDA_DOWN or OPCHDA_INDETERMINATE)
*_pftCurrentTime	The current time at the historian location
*_pftStartTime	The time when the historian was last started
_pwMajorVersion	The major version identification of the historian
_pwMinorVersion	The minor version identification of the historian
_pwBuildNumber	The build number identification of the historian
_pdwMaxReturnValues	The maximum number of values that can be returned by the server on a per item basis. A value of 0 indicates that the server forces no limit on the number of values it can return.
_ppszStatusString	A string explaining historian status when the pwStatus value is OPCHDA_INDETERMINATE.
_ppszVendorInfo	A vendor specific informational string.

Table 16: OPCHDA_GetHDAHistorianStatus- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_ FAIL	The function was unsuccessful.

Table 17: OPCHDA_GetHDAHistorianStatus- Return Values

Example:

```
OPCHDA_SERVERSTATUS  _pwStatus;
FILETIME *_pftCurrentTime;
FILETIME *_pftStartTime;
WORD      _pwMajorVersion,_pwMinorVersion,_pwBuildNumber;
DWORD     _pdwMaxReturnValues;
LPWSTR _ppszStatusString,_ppszVendorInfo;

int res=OPCHDA_GetHDAHistorianStatus(Shandle,
             &_pwStatus,
             &_pftCurrentTime,
             &_pftStartTime,
             &_pwMajorVersion,
             &_pwMinorVersion,
             &_pwBuildNumber,
             &_pdwMaxReturnValues,
             &_ppszStatusString,
             &_ppszVendorInfo);
```

4. Sync Read Methods

4.1. SynchReadReadHDARaw

This function reads the values, qualities, and timestamps from the historian for the specified time domain for one or more items. When *lngBounds* is TRUE, the bounding values for the time domain are returned. This function is intended for use by clients to collect the actual data to be saved within the historian. The actual data may be compressed or may be all data collected for the item depending on the historian and the storage rules invoked when the item values were saved. The optional bounding values are provided to allow clients to interpolate values for the start and end times when trending the actual data on a display.

```
HRESULT __stdcall OPCHDA_SynchReadReadHDARaw (int _iServerHandle,
             OPCHDA_TIME   *_htStartTime,
             OPCHDA_TIME   *_htEndTime,
             DWORD         _dwNumValues,
             BOOL          _bBounds,
             DWORD         _dwNumItems,
             OPCHANDLE     *_phServer,
             OPCHDA_ITEM   **_ppItemValues,
             HRESULT       **_ppErrors)
```

Parameters

Parameter	Description
_iServerHandle	The handle of the server's connection.
*_htStartTime	The beginning of the history period to be read.
*_htEndTime	The end of the history period to be read.
_dwNumValues	The maximum number of values returned for any item over the time range. If only one time is specified, the time range must extend to return this number of values.
_bBounds	1 if bounding values should be returned (0 otherwise).
_dwNumItems	The number of items to be read
_phServer	The server item handle for the item to be read
*_ppItemValues	The returned structure of item's values
*_ppErrors	Returned errors

Table 18: OPCHDA_SynchReadReadHDARaw- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine success of individual validations
IO_E_MAXEXCEEDED	The maximum number of values requested (dwNumValues) is greater than the server limit of maximum values returned.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.

Table 19: OPCHDA_SynchReadReadHDARaw- Return Values

_ppErrorsReturn Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_BADRIGHTS	Insufficient rights for this operation.

IO_E_INVALIDHANDLE	The handle is invalid.
IO_S_MOREDATA	More data is available in the time range beyond the number of values requested.
IO_S_NODATA	No data was found in the specified time range.
IO_E_FAIL	The item read was unsuccessful.

Table 20: OPCHDA_SynchReadReadHDARaw- _ppErrorsReturn Values

4.2. SynchReadReadHDAProcessed

This function reads aggregate values, qualities, and timestamps from the historian for the specified time domain for one or more items. The time domain is divided into subintervals of duration *ftResampleInterval*. The specified *haAggregate* is calculated for each subinterval beginning with *htStartTime* by using the data within the next *ftResampleInterval*.

This function is intended to provide values calculated with respect to the resample interval. For example, this function can provide hourly statistics such as Maximum, Minimum, Average, etc. for each item during the specified time domain when *ftResampleInterval* is 1 hour.

```
HRESULT __stdcall OPCHDA_SynchReadReadHDAProcessed (int _iServerHandle,
              OPCHDA_TIME         *_htStartTime,
              OPCHDA_TIME         *_htEndTime,
              FILETIME            _ftResampleInterval,
              DWORD               _dwNumItems,
              OPCHANDLE           *_phServer,
              OPCHDA_AGGREGATE    *_phaAggregate,
              OPCHDA_ITEM         **_ppItemValues,
              HRESULT             **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`*_htStartTime`	The beginning of the history period to be read.
`*_htEndTime`	The end of the history period to be read.
`_ftResampleInterval`	Interval between returned values.
`_dwNumItems`	The number of items to be read
`_phServer`	The server item handle for the item to be read
`_phaAggregate`	The calculation to be performed on the raw data to create the values to be returned.
`*_ppItemValues`	The returned structure of item's values
`*_ppErrors`	Returned errors

Table 21: OPCHDA_SynchReadReadHDAProcessed- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine success of individual validations
IO_E_MAXEXCEEDED	The maximum number of values requested (dwNumValues) is greater than the server limit of maximum values returned.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.

Table 22: OPCHDA_SynchReadReadHDAProcessed- Return Values

_ppErrorsReturn Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_NODATA	No data was found in the specified time range.
IO_S_EXTRADATA	There is more data available than was returned.
IO_E_NOT_AVAIL	The requested aggregate is not available from the provided item.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.
IO_E_FAIL	The Item read was unsuccessful.

Table 23:OPCHDA_SynchReadReadHDAProcessed- _ppErrorsReturn Values

4.3. SynchReadReadHDAAtTime

This function reads the values and qualities from the historian for the specified timestamps for one or more items. This function is intended to provide values to correlate with other values with a known timestamp.

```
HRESULT __stdcall OPCHDA_SynchReadReadHDAAtTime (int _iServerHandle,
              DWORD         _dwNumTimeStamps,
              FILETIME      *_pftTimeStamps,
              DWORD         _dwNumItems,
              OPCHANDLE     *_phServer,
              OPCHDA_ITEM   **_ppItemValues,
              HRESULT       **_ppErrors)
```

Parameters

Parameter	Description
_iServerHandle	The handle of the server's connection.
_dwNumTimeStamps	The number of timestamps specified.
*_pftTimeStamps	Array of timestamps for the requested data.
_dwNumItems	The number of items to be read
_phServer	The server item handle for the item to be read
*_ppItemValues	The returned structure of item's values
*_ppErrors	The status of Read At Time call.

Table 24: OPCHDA_SynchReadReadHDAAtTime- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_INVALIDARG	An Invalid parameter was passed.
IO_E_NOTIMPL	This server does not support this function.
IO_E_FAIL	The function was unsuccessful.

Table 25: OPCHDA_SynchReadReadHDAAtTime- Return Values

_ppErrors Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.
IO_S_NODATA	No data was found for the item.
IO_E_FAIL	The item read was unsuccessful.

Table 26: OPCHDA_SynchReadReadHDAAtTime- _ppErrors Return Values

4.4. SynchReadReadHDAModified

This function reads the values, qualities, timestamps, user ID, and timestamp of the modification from the history database for the specified time domain for one or more items. The purpose of this function is to read values from history that have been modified / replaced.

```
HRESULT __stdcall OPCHDA_SynchReadReadHDAModified (int _iServerHandle,
              OPCHDA_TIME          *_htStartTime,
              OPCHDA_TIME          *_htEndTime,
              DWORD                _dwNumValues,
              DWORD                _dwNumItems,
              OPCHANDLE            *_phServer,
              OPCHDA_MODIFIEDITEM **_ppItemValues,
              HRESULT             **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`*_htStartTime`	The beginning of the history period to be read.
`*_htEndTime`	The end of the history period to be read.
`_dwNumValues`	The maximum number of values returned for any item over the time range. If only one time is specified, the time range must extend to return this number of values.
`_dwNumItems`	The number of items to be read
`_phServer`	The server item handle for the item to be read
`*_ppItemValues`	The returned structure of item's values
`*_ppErrors`	The status of Read Modified call.

Table 27: OPCHDA_SynchReadReadHDAModified- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_INVALIDARG	An Invalid parameter was passed.
IO_E_NOTIMPL	This server does not support this function.
IO_E_FAIL	The function was unsuccessful.

Table 28: OPCHDA_SynchReadReadHDAModified- Return Values

_ppErrorsReturn Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_BADRIGHTS	Insufficient rights for this operation..
IO_E_INVALIDHANDLE	The handle is invalid.
IO_S_MOREDATA	More data is available in the time range beyond the number of values requested.
IO_S_NODATA	No data was found in the specified time range.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The item read was unsuccessful.

Table 29: OPCHDA_SynchReadReadHDAModified- _ppErrors Return Values

4.5. SynchReadReadHDAAttribute

This function reads the attribute values and timestamps from the history database for the specified time domain for an item. If the current values for the attributes are desired, htStartTime shall be set to "NOW" and htEndTime shall be NULL time.

This function is intended to be used to retrieve attributes that have changed to correlate the values of these attributes with the values of their data. For example, the recalibration of a sensor may have required the normal maximum and minimum attributes to be changed.

```
HRESULT __stdcall OPCHDA_SynchReadReadHDAAttribute (int _iServerHandle,
             OPCHDA_TIME          *_htStartTime,
             OPCHDA_TIME          *_htEndTime,
             OPCHANDLE            _hServer,
             DWORD                _dwNumAttributes,
             DWORD                *_pdwAttributeIDs,
             OPCHDA_ATTRIBUTE     **_ppAttributeValues,
             HRESULT              **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server’s connection.
`*_htStartTime`	The beginning of the history period to be read.
`*_htEndTime`	The end of the history period to be read.
`_hServer`	The server item handle for the item to be read.
`_dwNumAttributes`	The number of attributes to be read

`*_pdwAttributeIDs`	The attribute IDs to be read
`*_ppAttributeValues`	The returned structure of item’s values
`*_ppErrors`	The status of Read Attributecall.

Table 30: OPCHDA_SynchReadReadHDAAttribute- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_INVALIDHANDLE	The handle is invalid.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.

Table 31: OPCHDA_SynchReadReadHDAAttribute- Return Values

_ppErrorsReturn Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDATTRID	The attribute ID is not valid.
IO_E_FAIL	The item read was unsuccessful.
IO_S_CURRENTVALUE	No history available for attribute.

Table 32: OPCHDA_SynchReadReadHDAAttribute- _ppErrors Return Values

5. Sync Update Methods

5.1. SynchUpdateQueryHDACapabilities

This function specifies the update methods that the server supports.

```
HRESULT __stdcall OPCHDA_SynchUpdateQueryHDACapabilities (int _iServerHandle,
        OPCHDA_UPDATECAPABILITIES *_pCapabilities)
```

Parameters

Parameter	Description
_iServerHandle	The handle of the server's connection.
_pCapabilities	Array of server supported capabilities ID.

Table 33: OPCHDA_SynchUpdateQueryHDACapabilities- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_NOTIMP	This server does not support this function.
IO_E_FAIL	The function was unsuccessful.

Table 34: OPCHDA_SynchUpdateQueryHDACapabilities- Return Values

5.2. SynchUpdateHDAInsert

This function inserts values and qualities into the history database at the specified timestamps for one or more items. If a value exists at the specified timestamp, the new value shall not be inserted; instead, ppErrors shall indicate an error.

This function is intended to insert new values at the specified timestamps; e.g., the insertion of lab data to reflect the time of data collection.

```
HRESULT __stdcall OPCHDA_SynchUpdateHDAInsert  (int   _iServerHandle,
              DWORD         _dwNumItems,
              OPCHANDLE*    _phServer,
              FILETIME      *_pftTimeStamps,
              VARIANT       *_pvDataValues,
              DWORD         *_pdwQualities,
              HRESULT       **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`_dwNumItems`	The number of items to be inserted.
`*_phServer`	The list of server item handles for the items to be inserted.
`*_pftTimeStamps`	Array of the time stamps for the new values.
`*_pvDataValues`	Array of structures which contain the new item values.
`*_pdwQualities`	Array of quality flags of the new values.
`*_ppErrors`	Array of HRESULTs indicating the success of the individual item. The errors correspond to the handles passed in phServer.

Table 35: OPCHDA_SynchUpdateHDAInsert - Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_NOTIMPL	This server does not support this function.
IO_E_FAIL	The function was unsuccessful.

Table 36: OPCHDA_SynchUpdateHDAInsert - Return Values

_ppErrors Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.
IO_E_DATAEXISTS	Unable to insert – data already present..
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The Item update was unsuccessful.

Table 37: OPCHDA_SynchUpdateHDAInsert- _ppErrors Return Values

5.3. SynchUpdateHDAReplace

This function replaces the values and qualities in the history database at the specified timestamps for one or more items. If no value exists at the specified timestamp, the new value shall not be inserted; instead, ppErrors shall indicate an error.

This function is intended to replace existing values at the specified timestamp; e.g., correct lab data that was improperly processed, but inserted into the history database.

```
HRESULT __stdcall OPCHDA_SynchUpdateHDAReplace (int    _iServerHandle,
              DWORD          _dwNumItems,
              OPCHANDLE*     _phServer,
              FILETIME       *_pftTimeStamps,
              VARIANT        *_pvDataValues,
              DWORD          *_pdwQualities,
              HRESULT        **_ppErrors)
```

Parameters

Parameter	Description
_iServerHandle	The handle of the server's connection.
_dwNumItems	The number of items to be replaced.
*_phServer	The list of server item handles for the items to be replaced.
*_pftTimeStamps	Array of the time stamps for the new values.
*_pvDataValues	Array of structures which contain the new item values.
*_pdwQualities	Array of the quality flags of the new values.
*_ppErrors	Array of HRESULTs indicating the success of the individual item. The errors correspond to the handles passed in phServer.

Table 38: OPCHDA_SynchUpdateHDAReplace - Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_NOTIMPL	This server does not support this function.

IO_E_FAIL	The function was unsuccessful.

Table 39: OPCHDA_SynchUpdateHDAReplace - Return Values

_ppErrorsReturn Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.
IO_E_DATAEXISTS	Unable to insert – data already present..
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The Item update was unsuccessful.

Table 40: OPCHDA_SynchUpdateHDAReplace - _ppErrors Return Values

5.4. SynchUpdateHDAInsertReplace

This function inserts or replaces values and qualities in the history database for the specified timestamps for one or more items. If the item has a value at the specified timestamp, the new value and quality will replace the old one. If there is no value at that timestamp, the function will insert the new data. The function runs to completion before returning.

This function is intended to unconditionally insert/replace values and qualities; e.g., correction of values for bad sensors.

```
HRESULT __stdcall OPCHDA_SynchUpdateHDAInsertReplace (int _iServerHandle,
              DWORD         _dwNumItems,
              OPCHANDLE*    _phServer,
              FILETIME      *_pftTimeStamps,
              VARIANT       *_pvDataValues,
              DWORD         *_pdwQualities,
              HRESULT       **_ppErrors)
```

Parameters

Parameter	Description
_iServerHandle	The handle of the server’s connection.
_dwNumItems	The number of items to be edited.
*_phServer	The list of server item handles for the items to be edited.
*_pftTimeStamps	Array of the time stamps for the new values.

`*_pvDataValues`	Array of structures which contain the new item values.
`*_pdwQualities`	Array of the quality flags of the new values.
`*_ppErrors`	Array of HRESULTs indicating the success of the individual item. The errors correspond to the handles passed in phServer.

Table 41: OPCHDA_SynchUpdateHDAInsertReplace - Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_NOTIMPL	This server does not support this function.
IO_E_FAIL	The function was unsuccessful.

Table 42: OPCHDA_SynchUpdateHDAInsertReplace - Return Values

_ppErrorsReturn Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.
IO_E_DATAEXISTS	Unable to insert – data already present.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The Item update was unsuccessful.

Table 43: OPCHDA_SynchUpdateHDAInsertReplace - _ppErrors Return Values

5.5. SynchUpdateHDADeleteRaw

This function deletes the values, qualities, and timestamps from the history database for the specified time domain for one or more items.

This function is intended to be used to delete data that has been accidentally entered into the history database; e.g., deletion of data from a source with incorrect timestamps.

```
HRESULT __stdcall OPCHDA_SynchUpdateHDADeleteRaw (int _iServerHandle,
OPCHDA_TIME          *_htStartTime,
OPCHDA_TIME          *_htEndTime,
DWORD                _dwNumItems,
OPCHANDLE            *_phServer,
HRESULT              **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`*_htStartTime`	The beginning of the history period to be deleted.
`*_htEndTime`	The end of the history period to be deleted.
`_dwNumItems`	The number of items to be deleted
`*_phServer`	The list of server item handles for the items to be deleted.
`*_ppErrors`	Array of HRESULTs indicating the success of the individual item deletes. The errors correspond to the handles passed in phServer. This indicates whether the delete succeeded in removing the specified items.

Table 44: OPCHDA_SynchUpdateHDADeleteRaw- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_NOTIMPL	This server does not support this function.
IO_E_FAIL	The function was unsuccessful.

Table 45: OPCHDA_SynchUpdateHDADeleteRaw- Return Values

_ppErrorsReturn Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.
IO_E_DATAEXISTS	Unable to insert – data already present..

IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The Item update was unsuccessful.

Table 46: OPCHDA_SynchUpdateHDADeleteRaw- _ppErrors Return Values

5.6. SynchUpdateHDADeleteAtTime

This function deletes the values and qualities in the history database for the specified timestamps for one or more items.

This function is intended to be used to delete specific data from the history database; e.g., lab data that is incorrect and cannot be correctly reproduced.

```
HRESULT __stdcall OPCHDA_SynchUpdateHDADeleteAtTime(int_iServerHandle,
              DWORD         _dwNumItems,
              OPCHANDLE     *_phServer,
              FILETIME      *_pftTimeStamps,
              HRESULT       **_ppErrors)
```

Parameters

Parameter	Description
_iServerHandle	The handle of the server's connection.
_dwNumItems	The number of items to be deleted.
*_phServer	The list of server item handles for the items to be deleted.
*_pftTimeStamps	The timestamps for the data to be deleted.
*_ppErrors	Array of HRESULTs indicating the success of the individual item deletes. The errors correspond to the timestamps passed in phServer.

Table 47: OPCHDA_SynchUpdateHDADeleteAtTime - Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_NOTIMPL	This server does not support this function.
IO_E_FAIL	The function was unsuccessful.

Table 48: OPCHDA_SynchUpdateHDADeleteAtTime - Return Values

_ppErrorsReturn Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.
IO_E_DATAEXISTS	Unable to insert – data already present..
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The Item update was unsuccessful.

6. Synch Annotations Methods

6.1. SynchAnnotationsQueryHDACapabilities

This function specifies which update methods the server supports. It is a required method for all servers, which support the OPCHDA SyncAnnotations interface.

```
HRESULT __stdcall OPCHDA_SynchAnnotationsQueryHDACapabilities (int  _iServerHandle,
              OPCHDA_ANNOTATIONCAPABILITIES *_pCapabilities)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`*_pCapabilities`	Array of server supported annotation query capabilities ID

Table 49: OPCHDA_SynchAnnotationsQueryHDACapabilities- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_FAIL	The function was unsuccessful.

Table 50: OPCHDA_SynchAnnotationsQueryHDACapabilities- Return Values

6.2. SynchAnnotationsHDAReadAnnotation

This function reads the annotations from the history database in the specified time domain for the specified item IDs.

This function is intended to read annotations for an item at specified timestamps.

```
HRESULT __stdcall OPCHDA_SynchAnnotationsHDAReadAnnotation (int _iServerHandle,
              OPCHDA_TIME                  *_htStartTime,
              OPCHDA_TIME                  *_htEndTime,
              DWORD                        _dwNumItems,
              OPCHANDLE                    *_phServer,
              OPCHDA_ANNOTATION            **_pAnnotationValues,
              HRESULT                      **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`*_htStartTime`	The beginning of the history period to be read.
`*_htEndTime`	The end of the history period
`_dwNumItems`	The number of items to be read
`*_phServer`	The server item handle for the annotation values to be read
`*_pAnnotationValues`	The returned structure of annotation's values
`*_ppErrors`	Returned errors

Table 51: OPCHDA_SynchAnnotationsHDAReadAnnotation- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_NOTIMPL	This server does not support this function.
IO_E_FAIL	The function was unsuccessful.

Table 52: OPCHDA_SynchAnnotationsHDAReadAnnotation- Return Values

_ppErrors Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.
IO_S_NODATA	No data was found in the specified time range.
IO_E_FAIL	The Item update was unsuccessful.

Table 53: OPCHDA_SynchAnnotationsHDAReadAnnotation- _ppErrors Return Values

6.3. SynchAnnotationsHDAInsertAnnotation

This function is intended to insert annotations into the history database by users in order to document observations for a value at a specified timestamp.

```
HRESULT __stdcall OPCHDA_SynchAnnotationsHDAInsertAnnotation (int   _iServerHandle,
             DWORD                 _dwNumItems,
             OPCHANDLE             *_phServer,
             FILETIME              *_pftTimeStamps,
             OPCHDA_ANNOTATION     *_pAnnotationValues,
             HRESULT               **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`_dwNumItems`	The number of the annotations to be inserted
`*_phServer`	The server item handle for the annotations values to be insert
`*_pftTimeStamps`	Array of timestamps for the requested data
`*_pAnnotationValues`	The structure of annotation's values
`*_ppErrors`	Returned errors

Table 54: OPCHDA_SynchAnnotationsHDAInsertAnnotation- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.

IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_NOTIMPL	This server does not support this function.
IO_E_FAIL	The function was unsuccessful.
IO_E_INVALIDARG	An invalid parameter was passed.

Table 55: OPCHDA_SynchAnnotationsHDAInsertAnnotation- Return Values

_ppErrorsReturn Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.
IO_E_FAIL	The Item update was unsuccessful.

Table 56: OPCHDA_SynchAnnotationsHDAInsertAnnotation- _ppErrors Return Values

7. Asynch Read Methods

7.1. ASynchReadReadHDARaw

This function reads the values, qualities, and timestamps from the history database for the specified time domain for one or more items. When *bBounds* is TRUE, the bounding values for the time domain are returned. This function is intended for use by clients wanting the actual data saved within the historian. The actual data may be compressed or may be all data collected for the item depending on the historian and the storage rules invoked when the item values were saved. The optional bounding values are provided to allow clients to interpolate values for the start and end times when trending the actual data on a display.
The results are returned via the client's OnReadComplete method.

```
HRESULT __stdcall OPCHDA_ASynchReadReadHDARaw(int      _iServerHandle,
             DWORD               _dwTransactionID,
             OPCHDA_TIME         *_htStartTime,
             OPCHDA_TIME         *_htEndTime,
             DWORD               _dwNumValues,
             BOOL                _bBounds,
             DWORD               _dwNumItems,
             OPCHANDLE           *_phServer,
            DWORD                    *_pdwCancelID,
            HRESULT                  **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`_dwTransactionID`	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
`*_htStartTime`	The beginning of the history period to be read.
`*_htEndTime`	The end of the history period to be read.
`_dwNumValues`	The maximum number of values returned for any item over the time range. If only one time is specified, the time range must extend to return this number of values.
`_bBounds`	1 if bounding values should be returned.
`_dwNumItems`	The number of items to be read
`*_phServer`	The server item handle for the item to be read
`*_pdwCancelID`	The server-generated cancelID
`*_ppErrors`	Returned errors

Table 57: OPCHDA_ASynchReadReadHDARaw - Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_FAIL	The function was unsuccessful.
IO_E_MAXEXCEEDED	The maximum number of values requested (dwNumValues) is greater than the server limit of maximum values returned.
IO_E_NOTIMPL	This server does not support this function.
IO_E_INVALIDARG	The function was unsuccessful.

Table 58: OPCHDA_ASynchReadReadHDARaw- Return Values

_ppErrorsReturn Values

Return Code	Description
IO_S_OK	The item was read successfully.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.

Table 59: OPCHDA_ASynchReadReadHDARaw- _ppErrors Return Values

7.2. ASynchReadHDAAdviseRaw

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

This function is intended to be used to update the client software with new data as it becomes available; e.g., update a trend with new data on a periodic basis.

The results are returned via the client's OnDataChange method.

```
HRESULT __stdcall OPCHDA_ASynchReadHDAAdviseRaw (int     _iServerHandle,
             DWORD               _dwTransactionID,
             OPCHDA_TIME         *_htStartTime,
             FILETIME            _ftUpdateInterval,
             DWORD               _dwNumItems,
             OPCHANDLE           *_phServer,
             DWORD               *_pdwCancelID,
             HRESULT             **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`_dwTransactionID`	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
`*_htStartTime`	The beginning of the history period to be read.
`_ftUpdateInterval`	Update interval to send new data
`_dwNumItems`	The number of items to be read
`*_phServer`	The server item handle for the item to be read
`_pdwCancelID`	The server-generated cancelID
`*_ppErrors`	Returned errors

Table 60: OPCHDA_ASynchReadHDAAdviseRaw - Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_NOTIMPL	This server does not support this function.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.
IO_S_UNSUPPORTEDRATE	The requested update interval is not supported by the server.

Table 61: OPCHDA_ASynchReadHDAAdviseRaw - Return Values

_ppErrorsReturn Values

Return Code	Description
IO_S_OK	The item was read successfully.
IO_E_INVALIDARG	An Invalid parameter was passed.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.

Table 62: OPCHDA_ASynchReadHDAAdviseRaw- _ppErrors Return Values

7.3. ASynchReadReadHDAProcessed

This function reads aggregate values, qualities, and timestamps from the history database for the specified time domain for one or more items. The time domain is divided into subintervals of duration *ftResampleInterval*. The specified *haAggregate* is calculated for each subinterval beginning with *htStartTime* by using the data within the next *ftResampleInterval*.

This function is intended to provide values calculated with respect to the resample interval. For example, this function can provide hourly statistics such as Maximum, Minimum, Average, etc. for each item during the specified time domain when *ftResampleInterval* is 1 hour.

The results are returned via the client's OnReadComplete method.

```
HRESULT __stdcall   OPCHDA_ASynchReadReadHDAProcessed (int   _iServerHandle,
             DWORD                _dwTransactionID,
             OPCHDA_TIME          *_htStartTime,
             OPCHDA_TIME          *_htEndTime,
             FILETIME             _ftResampleInterval,
             DWORD                _dwNumItems,
             OPCHANDLE            *_phServer,
             OPCHDA_AGGREGATE     *_phaAggregate,
             DWORD                *_pdwCancelID,
             HRESULT              **_ppErrors)
```

Parameters

Parameter	Description
_iServerHandle	The handle of the server's connection.
_dwTransactionID	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
*_htStartTime	The beginning of the history period to be read.
*_htEndTime	The end of the history period to be read.
_ftResampleInterval	Interval between returned values.
_dwNumItems	The number of items to be read
*_phServer	The server item handle for the item to be read
*_phaAggregate	The calculation to be performed on the raw data to create the values to be returned
_pdwCancelID	The server-generated cancelID
*_ppErrors	Returned errors

Table 63: OPCHDA_ASynchReadReadHDAProcessed - Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_MAXEXCEEDED	The maximum number of values returnable by the server was exceeded.
IO_E_NOTIMPL	This server does not support this function.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.

Table 64: OPCHDA_ASynchReadReadHDAProcessed- Return Values

_ppErrorsReturn Values

Return Code	Description
IO_S_OK	The item was read successfully.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.

Table 65: OPCHDA_ASynchReadReadHDAProcessed- _ppErrors Return Values

7.4. ASynchReadHDAAdviseProcessed

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

This function is intended to be used to update the client software with new data as it becomes available; e.g., update a trend with new data on a periodic basis.

The results are returned via the client's OnDataChange method.

```
HRESULT __stdcall   OPCHDA_ASynchReadHDAAdviseProcessed (int _iServerHandle,
              DWORD                 _dwTransactionID,
              OPCHDA_TIME           *_htStartTime,
              FILETIME              _ftResampleInterval,
              DWORD                 _dwNumItems,
              OPCHANDLE             *_phServer,
              OPCHDA_AGGREGATE      *_phaAggregate,
              DWORD                 _dwNumIntervals,
              DWORD                 *_pdwCancelID,
              HRESULT               **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`_dwTransactionID`	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
`*_htStartTime`	The beginning of the history to be read.
`_ftResampleInterval`	Interval between returned values.
`_dwNumItems`	The number of items to be read

`*_phServer`	The server item handle for the item to be read
`*_phaAggregate`	The calculation to be performed on the raw data to create the values to be returned
`_dwNumIntervals`	Number of resample intervals between updates.
`_pdwCancelID`	The server-generated cancelID
`*_ppErrors`	Returned errors

Table 66: OPCHDA_ASynchReadHDAAdviseProcessed - Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_NOTIMPL	This server does not support this function.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.
IO_S_UNSUPPORTEDRATE	The requested resample interval is not supported by the server.

Table 67: OPCHDA_ASynchReadHDAAdviseProcessed - Return Values

_ppErrorsReturn Values

Return Code	Description
IO_S_OK	The item was read successfully.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.

Table 68: OPCHDA_ASynchReadHDAAdviseProcessed - _ppErrors Return Values

7.5. ASynchReadHDAReadAtTime

This function reads the values and qualities from the history database for the specified timestamps for one or more items. This function is intended to provide values to correlate with other values with a known timestamp. For example, the values of sensors when lab samples were collected

The results are returned via the client's OnReadComplete method.

```
HRESULT __stdcall OPCHDA_ASynchReadHDAReadAtTime (int _iServerHandle,
              DWORD                 _dwTransactionID,
              DWORD                 _dwNumTimeStamps,
              FILETIME              *_pftTimeStamps,
              DWORD                 _dwNumItems,
              OPCHANDLE             *_phServer,
              DWORD                 *_pdwCancelID,
              HRESULT               **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`_dwTransactionID`	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
`_dwNumTimeStamps`	The number of time stamps specified.
`*_pftTimeStamps`	The timestamps for the requested data.
`_dwNumItems`	The number of items to be read
`*_phServer`	The server item handle for the item to be read
`_pdwCancelID`	The server-generated cancelID
`*_ppErrors`	Returned errors

Table 69: OPCHDA_ASynchReadHDAReadAtTime - Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_NOTIMPL	This server does not support this function.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.

Table 70: OPCHDA_ASynchReadHDAReadAtTime - Return Values

_ppErrors Return Values

Return Code	Description
IO_S_OK	The item was read successfully.
IO_E_BADRIGHTS	Insufficient rights for this operation.

IO_E_INVALIDHANDLE	The handle is invalid.

Table 71: OPCHDA_ASynchReadHDAReadAtTime - _ppErrors Return Values

7.6. ASynchReadHDAReadModified

This function reads the values, qualities, timestamps, user ID, and timestamp of the modification from the history database for the specified time domain for one or more items.

The purpose of this function is to read values from history that have been modified/replaced (a value was returned with a quality of OPCHDA_EXTRADATA, indicating that there were other values for that item/timestamp which had been superseded).

The results are returned via the client's OnReadModifiedComplete method.

```
HRESULT __stdcall OPCHDA_ASynchReadHDAReadModified (int      _iServerHandle,
             DWORD                  _dwTransactionID,
             OPCHDA_TIME            *_htStartTime,
             OPCHDA_TIME            *_htEndTime,
             DWORD                  _dwNumValues,
             DWORD                  _dwNumItems,
             OPCHANDLE              *_phServer,
             DWORD                  *_pdwCancelID,
             HRESULT                **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`_dwTransactionID`	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
`*_htStartTime`	The beginning of the history period to be read.
`*_htEndTime`	The end of the history period to be read.
`_dwNumValues`	The maximum number of values returned for any item over the time range.
`_dwNumItems`	The number of items to be read
`*_phServer`	The server item handle for the item to be read
`_pdwCancelID`	The server-generated cancelID
`*_ppErrors`	Returned errors

Table 72: OPCHDA_ASynchReadHDAReadModified- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_NOTIMPL	This server does not support this function.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.

Table 73: OPCHDA_ASynchReadHDAReadModified- Return Values

_ppErrors Return Values

Return Code	Description
IO_S_OK	The item was read successfully.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.

Table 74: OPCHDA_ASynchReadHDAReadModified- _ppErrors Return Values

7.7. ASynchReadReadHDAAttribute

This function reads the attribute values and timestamps from the history database for the specified time domain for an item.

This function is intended to be used to retrieve attributes that have changed to correlate the values of these attributes with the values of their data. For example, the recalibration of a sensor may have required the normal maximum and minimum attributes to be changed.

The results are returned via the client's OnReadAttributeComplete method.

```
HRESULT __stdcall OPCHDA_ASynchReadReadHDAAttribute (int     _iServerHandle,
             DWORD                _dwTransactionID,
             OPCHDA_TIME          *_htStartTime,
             OPCHDA_TIME          *_htEndTime,
             OPCHANDLE            _hServer,
             DWORD                _dwNumAttributes,
             DWORD                *_pdwAttributeIDs,
             DWORD                *_pdwCancelID,
             HRESULT              **_ppErrors)
```

Parameters

Parameter	Description

`_iServerHandle`	The handle of the server's connection.
`_dwTransactionID`	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
`*_htStartTime`	The beginning of the attribute history period to be read.
`*_htEndTime`	The end of the attribute history period to be read.
`_hServer`	The server item handle for the item to be read
`_dwNumAttributes`	The number of attributes to be read.
`*_pdwAttributeIDs`	The list of attribute IDs to be read
`_pdwCancelID`	Place to return a Server generated ID to be used in case the operation needs to be canceled.
`*_ppErrors`	Array of HRESULTs indicating whether the corresponding dwAttributeID was valid.

Table 75: OPCHDA_ASynchReadReadHDAAttribute - Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_INVALIDHANDLE	The handle is invalid.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.

Table 76: OPCHDA_ASynchReadReadHDAAttribute - Return Values

_ppErrors Return Values

Return Code	Description
IO_S_OK	The item was read successfully.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDATTRID	The attribute ID is not valid.
IO_E_FAIL	The attribute read was unsuccessful.
IO_S_CURRENTVALUE	No history available for attribute.

Table 77: OPCHDA_ASynchReadReadHDAAttribute - _ppErrors Return Values

7.8. ASynchReadHDACancel

This function cancels the outstanding operation. The actual implementation is server specific, but the server shall respond via the client's OnCancelComplete function unless a FAILED error code is returned from the call. If a FAILED error code is returned, there will be no callback to the client's OnCancelComplete function.

```
HRESULT __stdcall   OPCHDA_ASynchReadHDACancel (int   _iServerHandle,
                    DWORD  _dwCancelID)
```

Parameters

Parameter	Description
_iServerHandle	The handle of the server's connection.
_dwCancelID	The server-generated cancelID which was returned from the original method call.

Table 78: OPCHDA_ASynchReadHDACancel- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_INVALIDARG	The client has not established a connection via the OPCHDA_CancelComplete ConnectionPoint.
IO_E_FAIL	The function was unsuccessful. The CancelID does not match any outstanding operation on the server.

Table 79: OPCHDA_ASynchReadHDACancel- Return Values

8. Asynch Update Methods

8.1. ASynchUpdateQueryHDACapabilities

This function specifies the update methods that the server supports.

```
HRESULT __stdcall   OPCHDA_ASynchUpdateQueryHDACapabilities  (int   _iServerHandle,
             OPCHDA_UPDATECAPABILITIES *_pCapabilities)
```

Parameters

Parameter	Description
_iServerHandle	The handle of the server's connection.

`*_pCapabilities`	Array of server supported update query capabilities.

Table 80:OPCHDA_ASynchUpdateQueryHDACapabilities- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_FAIL	The function was unsuccessful.

Table 81: OPCHDA_ASynchUpdateQueryHDACapabilities- Return Values

8.2. ASynchUpdateHDAInsert

This function inserts values and qualities into the history database for the specified timestamps for one or more items. If a value exists at the specified timestamp, the new value shall not be inserted; instead, ppErrors shall indicate an error.

This function is intended to insert new values at the specified timestamps; e.g., the insertion of lab data to reflect the time of data collection.

The results are returned via the client's OnUpdateComplete method.

```
HRESULT __stdcall OPCHDA_ASynchUpdateHDAInsert (int    _iServerHandle,
             DWORD                    _dwTransactionID,
             DWORD                    _dwNumItems,
             OPCHANDLE                *_phServer,
             FILETIME                 *_pftTimeStamps,
             VARIANT                  *_pvDataValues,
             DWORD                    *_pdwQualities,
             DWORD                    *_pdwCancelID,
             HRESULT                  **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server’s connection.
`_dwTransactionID`	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
`_dwNumItems`	The number of items to be inserted.
`*_phServer`	The list of server item handles for the items to be inserted.
`*_pftTimeStamps`	Array of the time stamps for the new values.

`*_pvDataValues`	Array of structures which contain the item values.
`*_pdwQualities`	Array of the quality flags of the new values.
`_pdwCancelID`	Place to return a Server generated ID to be used in case the operation needs to be canceled.
`*_ppErrors`	Array of HRESULTs indicating whether the corresponding server handle was valid.

Table 82:OPCHDA_ASynchUpdateHDAInsert- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_NOTIMPL	This server does not support this function.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.

Table 83: OPCHDA_ASynchUpdateHDAInsert - Return Values

_ppErrors Return Values

Return Code	Description
IO_S_OK	The item was inserted successfully.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.

Table 84: OPCHDA_ASynchUpdateHDAInsert -_ppErrors Return Values

8.3. ASynchUpdateHDAReplace

This function replaces values and qualities in the history database at the specified timestamps for one or more items. If no value exists at the specified timestamp, the new value shall not be inserted; instead, ppErrors shall indicate an error.

This function is intended to replace existing values at the specified timestamp; e.g., correct lab data that was improperly processed, but inserted into the history database. The results are returned via the client's OnUpdateComplete method.

```
HRESULT __stdcall OPCHDA_ASynchUpdateHDAReplace (int   _iServerHandle,
              DWORD                 _dwTransactionID,
              DWORD                 _dwNumItems,
              OPCHANDLE             *_phServer,
              FILETIME              *_pftTimeStamps,
              VARIANT               *_pvDataValues,
              DWORD                 *_pdwQualities,
              DWORD                 *_pdwCancelID,
              HRESULT               **_ppErrors)
```

Parameters

Parameter	Description
_iServerHandle	The handle of the server's connection.
_dwTransactionID	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
_dwNumItems	The number of items to be replaced.
*_phServer	The list of server item handles for the items to be replaced.
*_pftTimeStamps	Array of the time stamps for the new values.
*_pvDataValues	Array of structures which contain the item values.
*_pdwQualities	Array of the quality flags of the new values.
_pdwCancelID	Place to return a Server generated ID to be used in case the operation needs to be canceled.
*_ppErrors	Array of HRESULTs indicating whether the corresponding server handle was valid.

Table 85: OPCHDA_ASynchUpdateHDAReplace- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_NOTIMPL	This server does not support this function.

IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.

Table 86: OPCHDA_ASynchUpdateHDAReplace- Return Values

_ppErrors Return Values

Return Code	Description
IO_S_OK	The item was replaced successfully.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.

Table 87: OPCHDA_ASynchUpdateHDAReplace- _ppErrors Return Values

8.4. ASynchUpdateHDAInsertReplace

This function inserts or replaces values and qualities at the specified timestamps for one or more items. If the item has a value at the specified timestamp, the new value and quality shall replace the old one. If there is no value at that timestamp, the function shall insert the new data. This function is intended to unconditionally insert/replace values and qualities; e.g., correction of values for bad sensors. The results are returned via the client's OnUpdateComplete method.

```
HRESULT __stdcall   OPCHDA_ASynchUpdateHDAInsertReplace (int _iServerHandle,
             DWORD                  _dwTransactionID,
             DWORD                  _dwNumItems,
             OPCHANDLE              *_phServer,
             FILETIME               *_pftTimeStamps,
             VARIANT                *_pvDataValues,
             DWORD                  *_pdwQualities,
             DWORD                  *_pdwCancelID,
             HRESULT                **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`_dwTransactionID`	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
`_dwNumItems`	The number of items to be inserted or replaced.
`*_phServer`	The list of server item handles for the items to be inserted or replaced.

`*_pftTimeStamps`	Array of the time stamps for the new values.
`*_pvDataValues`	Array of structures which contain the item values.
`*_pdwQualities`	Array of the quality flags of the new values.
`_pdwCancelID`	Place to return a Server generated ID to be used in case the operation needs to be cancelled.
`*_ppErrors`	Array of HRESULTs indicating whether the corresponding server handle was valid.

Table 88: OPCHDA_ASynchUpdateHDAInsertReplace - Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_NOTIMPL	This server does not support this function.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.

Table 89: OPCHDA_ASynchUpdateHDAInsertReplace- Return Values

_ppErrors Return Values

Return Code	Description
IO_S_OK	The item was read successfully.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.

Table 90: OPCHDA_ASynchUpdateHDAInsertReplace- _ppErrors Return Values

8.5. ASynchUpdateHDADeleteRaw

This function deletes the values, qualities, and timestamps from the history database for the specified time domain for one or more items. This function is intended to be used to delete data that has been accidentally entered into the history database; e.g., deletion of data from a source with incorrect timestamps. The results are returned via the client'sOnUpdateComplete method.

```
HRESULT __stdcall   OPCHDA_ASynchUpdateHDADeleteRaw(int      _iServerHandle,
             DWORD               _dwTransactionID,
             OPCHDA_TIME         *_htStartTime,
             OPCHDA_TIME         *_htEndTime,
             DWORD               _dwNumItems,
             OPCHANDLE           *_phServer,
             DWORD               *_pdwCancelID,
             HRESULT             **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`_dwTransactionID`	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
`*_htStartTime`	The beginning of the history period to be deleted.
`*_htEndTime`	The end of the history period to be deleted.
`_dwNumItems`	The number of items to be deleted.
`*_phServer`	The list of server item handles for the items to be deleted.
`_pdwCancelID`	Place to return a Server generated ID to be used in case the operation needs to be canceled.
`*_ppErrors`	Array of HRESULTs indicating whether the corresponding server handle was valid.

Table 91: OPCHDA_ASynchUpdateHDADeleteRaw - Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_NOTIMPL	This server does not support this function.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.

Table 92: OPCHDA_ASynchUpdateHDADeleteRaw - Return Values

_ppErrors Return Values

Return Code	Description
IO_S_OK	The item was read successfully.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.

Table 93: OPCHDA_ASynchUpdateHDADeleteRaw - _ppErrors Return Values

8.6. ASynchUpdateHDADeleteAtTime

This function deletes the values and qualities in the history database for the specified timestamps for one or more items.

This function is intended to be used to delete specific data from the history database; e.g., lab data that is incorrect and cannot be correctly reproduced. The results are returned via the client's OnUpdateComplete method.

```
HRESULT __stdcall OPCHDA_ASynchUpdateHDADeleteAtTime(int     _iServerHandle,
             DWORD                 _dwTransactionID,
             DWORD                 _dwNumItems,
             OPCHANDLE             *_phServer,
             FILETIME              *_pftTimeStamps,
             DWORD                 *_pdwCancelID,
             HRESULT               **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server’s connection.
`_dwTransactionID`	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
`_dwNumItems`	The number of items to be deleted.
`*_phServer`	The list of server item handles for the items to be deleted.
`*_pftTimeStamps`	The timestamps for the data to be deleted.
`_pdwCancelID`	Place to return a Server generated ID to be used in case the operation needs to be canceled.
`*_ppErrors`	Array of HRESULTs indicating whether the corresponding server handle was valid.

Table 94: OPCHDA_ASynchUpdateHDADeleteAtTime- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_NOTIMPL	This server does not support this function.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.

Table 95: OPCHDA_ASynchUpdateHDADeleteAtTime- Return Values

_ppErrors Return Values

Return Code	Description
IO_S_OK	The item was read successfully.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.

Table 96: OPCHDA_ASynchUpdateHDADeleteAtTime- _ppErrors Return Values

8.7. ASynchUpdateHDACancel

This function cancels the outstanding operation. The actual implementation is server specific, but the server shall respond via the client's OnCancelComplete function.

```
HRESULT __stdcall OPCHDA_ASynchUpdateHDACancel  (int   _iServerHandle,
                    DWORD _dwCancelID)
```

Parameters

Parameter	Description
_iServerHandle	The handle of the server's connection.
_dwCancelID	The server-generated cancelID which was returned from the original method call.

Table 97: OPCHDA_ASynchUpdateHDACancel- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_INVALIDARG	The client has not established a connection via the OPCHDA_CancelComplete ConnectionPoint.
IO_E_FAIL	The function was unsuccessful. The CancelID does not match any outstanding operation on the server.

Table 98: OPCHDA_ASynchUpdateHDACancel- Return Values

9. Asynch Annotations Methods

9.1. ASynchAnnotationsQueryHDACapabilities

This function specifies the annotation functions that the server supports.

```
HRESULT __stdcall OPCHDA_ASynchAnnotationsQueryHDACapabilities (int _iServerHandle,
       OPCHDA_ANNOTATIONCAPABILITIES *_pCapabilities)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`*_pCapabilities`	Array of server supported annotation capabilities

Table 99: OPCHDA_ASynchAnnotationsQueryHDACapabilities- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_FAIL	The function was unsuccessful.

Table 100: OPCHDA_ASynchAnnotationsQueryHDACapabilities- Return Values

9.2. ASynchAnnotationsHDAReadAnnotation

This function reads the annotations from the history database in the specified time domain for the specified item IDs.

This function is intended to read annotations for an item at specified timestamps.

The results are returned via the client's OnReadAnnotations function.

```
HRESULT __stdcall OPCHDA_ASynchAnnotationsHDAReadAnnotation (int _iServerHandle,
              DWORD                _dwTransactionID,
              OPCHDA_TIME          *_htStartTime,
              OPCHDA_TIME          *_htEndTime,
              DWORD                _dwNumItems,
              OPCHANDLE            *_phServer,
              DWORD                *_pdwCancelID,
              HRESULT              **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`_dwTransactionID`	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
`*_htStartTime`	The beginning of the history period to be read.
`*_htEndTime`	The end of the history period to be read.
`_dwNumItems`	The number of items to be read.
`*_phServer`	The server item handle for the annotation item to be read.
`_pdwCancelID`	Place to return a Server generated ID to be used in case the operation needs to be canceled.
`*_ppErrors`	The state of the Read Annotation call.

Table 101: OPCHDA_ASynchAnnotationsHDAReadAnnotation- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.

Table 102: OPCHDA_ASynchAnnotationsHDAReadAnnotation- Return Values

_ppErrors Return Values

Return Code	Description
IO_S_OK	The item was read successfully.

IO_E_INVALIDARG	An Invalid parameter was passed.
IO_OPC_E_BADRIGHTS	Insufficient rights for this operation.
IO_OPC_E_INVALIDHANDLE	The handle is invalid.

Table 103: OPCHDA_ASynchAnnotationsHDAReadAnnotation- _ppErrors Return Values

9.3. ASynchAnnotationsHDAInsertAnnotation

This function is intended to insert annotations into the history database by users in order to document observations for a value at a specified timestamp. The results are returned via the client's OnInsertAnnotations function.

```
HRESULT __stdcall OPCHDA_ASynchAnnotationsHDAInsertAnnotation(int   _iServerHandle,
              DWORD                         _dwTransactionID,
              DWORD                         _dwNumItems,
              OPCHANDLE                     *_phServer,
              FILETIME                      *_pftTimeStamps,
              OPCHDA_ANNOTATION             *_pAnnotationValues,
              DWORD                         *_pdwCancelID,
              HRESULT                       **_ppErrors)
```

Parameters

Parameter	Description
_iServerHandle	The handle of the server's connection.
_dwTransactionID	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
_dwNumItems	The number of items to be read.
*_phServer	The server item handle for the annotation item to be inserted.
*_pftTimeStamps	The time stamp for the annotation to be inserted.
*_pAnnotationValues	The annotation values to be inserted.
*_pdwCancelID	Place to return a Server generated ID to be used in case the operation needs to be canceled.
*_ppErrors	The state of the Insert Annotation call.

Table 104: OPCHDA_ASynchAnnotationsHDAInsertAnnotation- Parameters

Return Values

Return Code	Description

IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_NOTIMPL	This server does not support this function.
IO_E_INVALIDARG	An invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.

Table 105: OPCHDA_ASynchAnnotationsHDAInsertAnnotation- Return Values

_ppErrors Return Values

Return Code	Description
IO_S_OK	The annotation was inserted successfully.
IO_E_INVALIDARG	An Invalid parameter was passed.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.

Table 106: OPCHDA_ASynchAnnotationsHDAInsertAnnotation- _ppErrors Return Values

9.4. ASynchAnnotationsHDACancel

This function cancels the outstanding operation. The actual implementation is server specific, but the server shall respond via the client's OnCancelComplete function.

```
HRESULT __stdcall OPCHDA_ASynchAnnotationsHDACancel (int     _iServerHandle,
                    DWORD  _dwCancelID)
```

Parameters

Parameter	Description
_iServerHandle	The handle of the server’s connection.
_dwCancelID	The server-generated cancelID which was returned from the original method call.

Table 107: OPCHDA_ASynchAnnotationsHDACancel - Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_INVALIDARG	The client has not established a connection via the OPCHDA_CancelComplete ConnectionPoint.

IO_E_FAIL	The function was unsuccessful. The CancelID does not match any outstanding operation on the server.

Table 108: OPCHDA_ASynchAnnotationsHDACancel - Return Values

10. Asynch Playback Methods

10.1. PlaybackHDAReadRawWithUpdate

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

This function is intended to be used to playback raw history data. By controlling the update interval, the data can be displayed on trends in real time, in slower motion, or faster than real time. The results are returned via the client's OnPlayback method.

```
HRESULT __stdcall OPCHDA_PlaybackHDAReadRawWithUpdate(int_iServerHandle,
              DWORD                   _dwTransactionID,
              OPCHDA_TIME             *_htStartTime,
              OPCHDA_TIME             *_htEndTime,
              DWORD                   _dwNumValues,
              FILETIME                _ftUpdateDuration,
              FILETIME                _ftUpdateInterval,
              DWORD                   _dwNumItems,
              OPCHANDLE               *_phServer,
              DWORD                   *_pdwCancelID,
              HRESULT                 **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`_dwTransactionID`	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
`*_htStartTime`	The earliest time of the history to be read.
`*_htEndTime`	The latest time of the history to be read. The end time must be greater than the start time, otherwise INVALID_PARMS is returned.
`_dwNumValues`	The maximum number of values returned for any

	item over the time range.
`_ftUpdateDuration`	The amount of time the update covers in.
`_ftUpdateInterval`	The interval to send data for updates.
`_dwNumItems`	The number of items to be read.
`*_phServer`	The server item handle for the item to be read.
`_pdwCancelID`	Place to return a Server generated ID to be used in case the operation needs to be canceled.
`*_ppErrors`	The state of the Read Raw With Update call.

Table 109: OPCHDA_PlaybackHDAReadRawWithUpdate - Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_MAXEXCEEDED	The maximum number of values returnable by the server was exceeded.
IO_E_INVALIDARG	An Invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.

Table 110: OPCHDA_PlaybackHDAReadRawWithUpdate - Return Values

_ppErrors Return Values

Return Code	Description
IO_S_OK	The Item was read successfully.
IO_S_NODATA	The function found no data to return.
IO_E_INVALIDARG	An Invalid parameter was passed.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.
IO_E_FAIL	The Item read was unsuccessful.

Table 111: OPCHDA_PlaybackHDAReadRawWithUpdate - _ppErrors Return Values

10.2. PlaybackHDAReadProcessedWithUpdate

This operation initially retrieves data from the start time to the end time. After the initial response, it periodically (every *ftUpdateInterval*) responds with *dwNumIntervals* worth of data divided into *ftResampleInterval* sized bins. The time of the last value returned in the initial response is used as the start time for the first update. After that, the time of the last value returned in an update is used as the start time for the next update.

This function is intended to be used to playback processed history data. By controlling the update interval, the data can be displayed on trends in real time, in slower motion, or faster than real time.

The results are returned via the client's OnPlayback function.

```
HRESULT __stdcall OPCHDA_PlaybackHDAReadProcessedWithUpdate (int _iServerHandle,
             DWORD                    _dwTransactionID,
             OPCHDA_TIME              *_htStartTime,
             OPCHDA_TIME              *_htEndTime,
             FILETIME                 _ftResampleInterval,
             DWORD                    _dwNumIntervals,
             FILETIME                 _ftUpdateInterval,
             DWORD                    _dwNumItems,
             OPCHANDLE                *_phServer,
             OPCHDA_AGGREGATE         *_haAggregate,
             DWORD                    *_pdwCancelID,
             HRESULT                  **_ppErrors)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`_dwTransactionID`	An identifier created by the client and passed to the server in this call. The server shall return this identifier along with the results of this call.
`*_htStartTime`	The earliest time of the history to be read.
`*_htEndTime`	The latest time of the history to be read.
`_ftResampleInterval`	Time between return values.
`_dwNumIntervals`	The number of ResampleIntervals in an update.
`_ftUpdateInterval`	The interval to send data for updates.
`_dwNumItems`	The number of items to be read.
`*_phServer`	The server handle of the item to be retrieved.
`*_haAggregate`	The aggregate to be returned.
`_pdwCancelID`	Place to return a Server generated ID to be used in case operation needs to be canceled.
`*_ppErrors`	The state of the Read Processed With Update call.

Table 112: OPCHDA_PlaybackHDAReadProcessedWithUpdate- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_S_FALSE	The function was partially successful. See the ppErrors to determine what happened.
IO_E_MAXEXCEEDED	The maximum number of values returnable by the server was exceeded.
IO_E_NOTIMPL	This server does not support this function.
IO_E_INVALIDARG	An Invalid parameter was passed.
IO_E_FAIL	The function was unsuccessful.

Table 113: OPCHDA_PlaybackHDAReadProcessedWithUpdate- Return Values

_ppErrors Return Values

Return Code	Description
IO_S_OK	The Item was read successfully.
IO_E_BADRIGHTS	Insufficient rights for this operation.
IO_E_INVALIDHANDLE	The handle is invalid.
IO_E_FAIL	The Item read was unsuccessful.

Table 114: OPCHDA_PlaybackHDAReadProcessedWithUpdate- _ppErrors Return Values

10.3. PlaybackHDACancel

This function cancels the outstanding operation. The actual implementation is server specific, but the server shall respond via the client's CancelComplete callback.

```
HRESULT__stdcall OPCHDA_PlaybackHDACancel(int         _iServerHandle,
                                    DWORD  _dwCancelID)
```

Parameters

Parameter	Description
`_iServerHandle`	The handle of the server's connection.
`_dwCancelID`	The server-generated cancelID which was returned from the original method call.

Table 115: OPCHDA_PlaybackHDACancel- Parameters

Return Values

Return Code	Description
IO_S_OK	The function was successful.
IO_E_INVALIDARG	The client has not established a connection via the OPCHDA_CancelComplete ConnectionPoint.
IO_E_FAIL	The function was unsuccessful. The CancelID does not match any outstanding operation on the server.

Table 116: OPCHDA_PlaybackHDACancel- Return Values

TOOLKIT TRACING CAPABILITIES

The toolkit has tracing capabilities. Developer can record the toolkit errors and debugging information (COM and OPC messages) in a log file named "HDAClientToolkit-LogEvent.LOG". If difficulties occur with the toolkit, the log file can be extremely valuable for troubleshooting. Under normal operation, the toolkit logs very little information.

This log file is generated at start-up where the server executable is located. The toolkit incorporates a configuration file "HDAClientToolkit-CfgFile.ini" which includes several logging parameters. All of these parameters have default settings and can be changed at start-up by editing the configuration file.

To change this file:

1. Open HDAClientToolkit-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 ~ 2 Mb (MegaByte)
LogLevel	The higher the log level, the more information is recorded. We recommend using level 0 for a better performance of the server.	0
ArchiveLastLog	TRUE: Old file is copied in an intermediate file with incremental extension, before being overwritten. FALSE: Any pre-existing log file is erased and overwritten at start-up	FALSE
LogFileName	The file to associate to the logevent file created by the toolkit	HDAClientToolkit-LogEvent

Table 117: Log Settings

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

Sample Configuration File

```
[LogSetting]
LogFileMaxSize = 2097152
LogLevel = 0
ArchiveLastLog = False
LogFileName = HDAClientToolkit-LogEvent
```

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 Process Control, OPC is based on Microsoft's OLE/COM technology. OPC 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.

APPENDIX A: DCOM CONFIGURATION (CLIENT SIDE)

In a local configuration, the OPC server and OPC client both run on a single machine. In that case, the installation process does not need any specific settings in the client side.
However, in distributed configuration, these components are executed on two or more machines cooperatively: the OPC client initially resides on a remote machine (client computer) on the network and uses the DCOM mechanism to directly access server. To enable this functionality, some settings are needed even on the client's computer.

You need to register your OPC server on the client's computer by indicating its location on named remote machine.

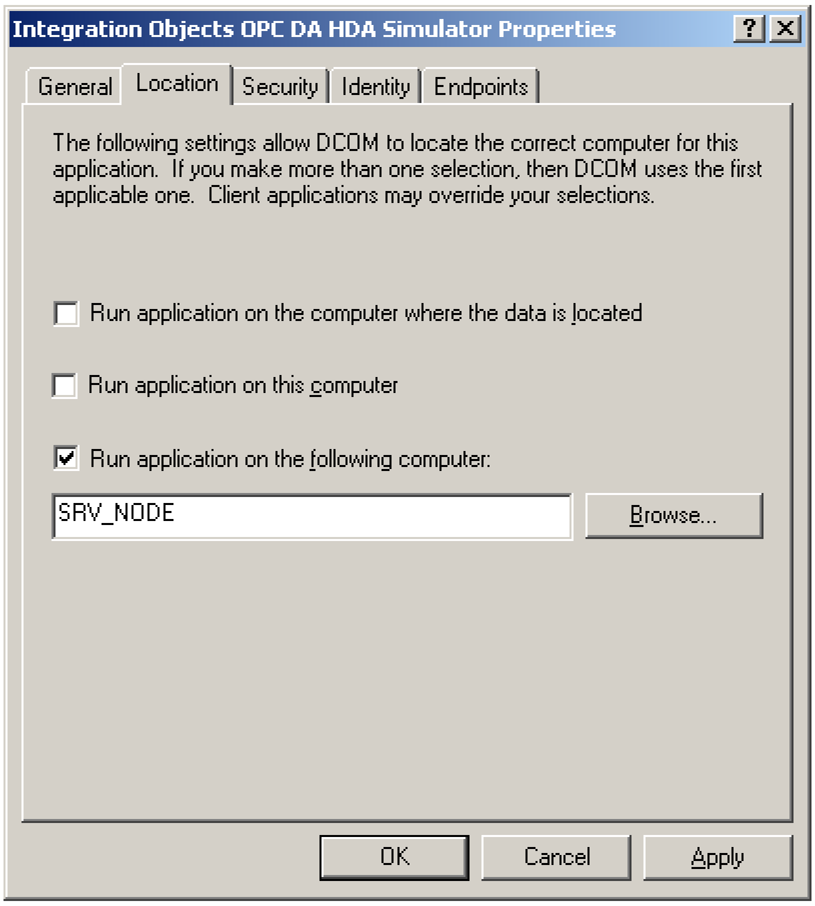
There are different ways of accomplishing this for your OPC server, depending on the client's environment. Here are two methods:

1. Prepare and apply a customized reg file on the client computer (See Microsoft registry documentation for details). We recommend this method only for experienced users with Windows Registry.

2. Alternatively, install and configure your OPC Server on the client's computer. This action -registers the server in the System Registry. This is the simplest way for an automatic registration.

   Then, use the following steps to verify that the OPC server machine is properly delegated:

   a. On the client's machine, run the DCOM Config Utility (Dcomcnfg.exe).
   b. Select your OPC server from the Applications tab and choose Properties.
   c. On the General tab, be sure that there is an entry for Remote Computer and that the remote computer's name is correct.
   d. If the computer name is incorrect, select the Location tab.
   e. Ensure the Run application on the following computer setting is checked. In the Dialog box beneath this selection, type in the correct computer name for your OPC server (see the figure below).


Integration Objects OPC DA HDA Simulator Properties

The following settings allow DCOM to locate the correct computer for this application. If you make more than one selection, then DCOM uses the first applicable one. Client applications may override your selections.

- [ ] Run application on the computer where the data is located
- [ ] Run application on this computer
- [x] Run application on the following computer:

SRV_NODE

You can also use the following steps to verify the remote computer name by using the Windows Registry:

a. Run RegEdit.exe.
b. The remote server name is specified in the following registry key:

HKEY_CLASSES_ROOT\AppID\{The CLSID of the OPC server}\RemoteServerName

APPENDIX B: TIPS FOR CONFIGURING DCOM SERVERS

Here are four tips for configuring DCOM Servers:

1. Users on Trusted Domains need to have an account created for them with matching usernames and passwords on the DCOM server's domain. The purpose of this is to set up a matching SID (Security ID). Trusted Domain group members need to have remote DCOM servers initiated for them by a Primary Domain member.

   A trusted Domain is a setup that allows resources from one domain to access resources in another domain. Trusted Domains typically go in one direction, although they can be bi-directional. The process of one domain trusting another and passing user authentication to another domain is called pass-through authentication.

2. Workgroup machines are individual domains, so you must set up matching SIDs (usernames and passwords) to establish connections between the machines.

3. Always create a Global Group through NT Server's User Manager and add the members for whom you want to provide access to specific DCOM servers. Then, use DCOMCnfg to set the launch permissions to that group. This makes administration easy to manage, even if you have a group that contains everyone.

4. If the client application implements a sink (callback), the server must be able to call the client back t. You must configure the client to accept calls from the server. Just because the client can connect to the server doesn't mean the server can call the client back .

APPENDIX C: DCOM ISSUES

This section addresses some DCOM related problems while using OPC servers:

Problem 1: You have an "Access denied" error on the client machine. The client and server are running on standalone machines (meaning not on the same domain).

Let's assume that the OPC client is running on machine A and the OPC server on machine B.
When OPC client and server are on different computers, you have to give each computer access to the other by giving access permissions. Permission issue is crucial to proper DCOM configurations.
Here the server is running on a standalone machine. So the ONLY user accounts it will trust are those it finds in its own "local" security database. Here is how this can get you into trouble on setting up and OPC client to server connection.
To allow remote client to access the DCOM server, DCOM utility uses Windows Security database. For this reason, you cannot give access to a user account which does not figure in this database.
Here are our propositions to resolve the issue:

1. You can add Machine B into the same domain as Machine A (or in a trusted domain), which is the most safe way to set up correctly the communication between the OPC client and the OPC server.
2. You need to create the EXACT SAME user account name AND password on BOTH machines (for example User1 (login), PWD1 (password)). Once you have that set up, when Machine A comes calling on Machine B with an OPC request and identifies himself as User1 with PWD1 password, Machine B will look in its database, see the same account name, the same password, and same "come on in request from Machine A". When Machine B goes to return its data from the OPC server to the OPC client on machine A, the OPC server will go call Machine A as User1 with a password -- Machine A will look in its database, see that it has that account, and accept the call. This

workaround should resolve the communication problem between the OPC client and server.

Problem 2: You have been running your OPC client on a Windows XP machine. When upgrading the machine to XP Service Pack 2, the OPC client becomes unable to connect to the OPC server.

This is a common problem when using OPC via DCOM with Microsoft Windows XP Service Pack 2.
In fact, when Service Pack 2 is installed with its default configuration settings, OPC communication via DCOM will cease to work.

To resolve this issue, you have to reconfigure your settings for Windows XP firewall and DCOM.

Click on this link Using OPC via DCOM with XP SP2.pdf to download the OPC Foundation document that describes all steps to apply new settings.

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 us on the Internet:

Online

- www.integrationobjects.com