WtHDAClient DLL User's Guide

The WinTECH Software Rapid HDA Client Development DLL, (WtHDAClient), provides an easy to use API for integrating a custom application with data from any OPC Historical Data Access Server. All the details of COM, OPC, and HDA are handled by the DLL, which allows an application to easily obtain data records from a Server, without having to be concerned with the actual implementation of the underlying interfaces. The DLL may be easily integrated with existing applications, or new ones. All required OPC Interfaces are supported for the OPC 1.0 Historical Data Access standard.

Creating a Custom OPC HDA Client using WtHDAClient.DLL

Link WtHDAClient.lib with the Application

WtHDAClient.lib contains the export definitions for the DLL's API. Include this file with the project files for the custom application and include WtHDAClientAPI.h with those modules which will be making calls into the DLL. If programming in Visual Basic, include the module1.bas from the VBA_example, (supplied with the demo), into your project.

DCOM Initialization

It is the Client Application's responsibility to properly initialize DCOM. The load-order of dll's makes it not feasible to perfrom this initialization within the DLLMain procedure, so WtHDAClient.DLL exports an API function, (WtHDAClientCoInit()), that performs the basic initializtion functions. WtHDAClientCoInit () initializes DCOM as multi-threaded and uses default security. Your application may choose to perform DCOM initialization itself rather than using the exported function. In either case, your application MUST call CoUninitialize() when terminating.

Obtaining the List of OPC HDA Servers

WtHDAClient.dll exports two functions that allow the controlling application to obtain a list of available OPC HDA Servers.

int NumberOfHDAServers (LPCSTR MachineName);
BOOL GetHDAServerName (int index, LPSTR Buffer, int BufSize);

The NumberOfHDAServers function scans the Windows Registry, (either on the local machine or the one specified by MachineName), to fill an internal array with the names of all the HDA Servers available.

Once the number of servers has been returned from WtHDAClient.dll, the application can iterate through the list of names by calling:

BOOL GetHDAServerName (int index, LPSTR Buffer, int BufSize);

A user supplied character buffer receives a name from the Server list as identified by a passed index. BufSize identifies the length of the user supplied character buffer and prevents the dll from loading a name that's too long. GetHDAServerName returns TRUE if a valid Server name is returned in Buffer, otherwise the return value is FALSE.

Establishing an OPC Connection

HANDLE ConnectHDA (LPCSTR MachineName, LPCSTR ServerName);

This function returns a valid HANDLE if a connection could be established to the HDA Server defined by MachineName and ServerName. Multiple simultaneous connections to different servers may be established, and the returned HANDLE identifies the connection for future calls to read and write records. Passing a Null String as the MachineName parameter identifies the local machine

void DisconnectHDA (HANDLE hConnect);

When an application terminates, it is responsible for disconnecting from an attached Server. The DisconnectHDA() function will provide a clean shutdown of the connection defined by hConnect.

Accessing data within a Server

Data Items

int NumberOfHDAItems(HANDLE hConnect);
BOOL GetHDAItemName (HANDLE hConnect, int index, char *pBuf, int BufSize);

These two functions operate similarly to those which allow you to obtain the list of Server names. NumberOfHDAItems() returns the total number of unique tags supported by a connected Server. Incrementally calling GetHDAItemName will step through all the available tag names to allow the controlling application to select an item or present a Browse list to the user. NumberOfHDAItems will browse the complete list of item names from the server and present the list in a 'FLAT' format.

HANDLE GetHDAItemHandle (HANDLE hConnect, LPCSTR ItemName, DWORD ClientHandle);
BOOL ReleaseHDAItemHandle (HANDLE hConnect, HANDLE hServer);

These two functions allow the application to obtain and release the server handle for an HDA item tag. The handle is used in subsequent calls to read and write item values.

```
DWORD  ReadHDAItemValues (HANDLE    hConnect,
                          HANDLE    hItem,\
                          BOOL      GetBoundingValues,
                          FILETIME  Start,
                          FILETIME  End,
                          DWORD     MaxValues,
                          FILETIME  *pTimeStamps,
                          VARIANT   *pValues,
                          DWORD     *pQualities);
```

The application can read data items from the HDA server between the specified Start & End times. hConnect specifies the Server. hItem is that returned from GetHDAItemHandle. If GetBoundingValues is TRUE the server will return the item value at or immediately prior to the Start time and the value at or immediately following the End time. Values are returned in the three arrays supplied by the user up to MaxValues. The return value contains the number of items read from the server. If the most significant bit of the return value is set, it indicates that more data was available from the server than would fit in the allocated buffers.

```
DWORD  WriteHDAItemValues (HANDLE   hConnect,
                           HANDLE   hItem,\
                           DWORB    UpdateType,
                           DWORD    NumValues,
                           FILETIME *pTimeStamps,
                           VARIANT  *pValues,
                           DWORD    *pQualities);
```

The application can write data values back to the connected HDA Server if the server supports the optional SyncUpdate interface. The UpdateType parameter refers to literals defined in the opchda.h header file and have values of **INSERT, REPLACE,** or **INSERTREPLACE**, (1,2 & 4 respectively), to determine the method used to update the data. Please refer to the OPC HAD Specification for details.

Attributes

int NumberOfHDAAttributes (HANDLE hConnect);

```
BOOL  GetHDAAttributeDescr (HANDLE    hConnect,
                            int       index,
                            DWORD     *pAttrId,
                            VARTYPE   *pVT,
                            char      *pNameBuf,
                            int       nBufSize,
                            char      *pDescBuf,
                            int        dBufSize);
```

These two functions may be used to obtain the list of HDA Item Attributes from the connected server. NumberOfHDAAttributes returns the number of attributes supported by the server and the description of each attribute may be obtained by calling GetHDAAttributeDescr. This function returns the attribute name, description, Id, and Variant Type in the user supplied buffers. TRUE is returned if the function is successful, otherwise FALSE.

```
DWORD  ReadHDAItemAttributes (HANDLE      hConnect,
                              HANDLE      hItem,
                              DWORD       AttrID,
                              FILETIME    Start,
                              FILETIME    End,
                              DWORD       MaxValues,
                              FILETIME    *pTimeStamps,
                              VARIANT     *pValues);
```

If the connected server supports archiving of item attribute values, the client application may retrieve historical data using the ReadHDAItemAttributes function. Start & End specify the time period of interest. The user supplied buffers, (pTimeStamps & pValues), will be filled with data from the serevr corresponding to the selected item and Attribute identification. The number of item attributes actually read will be returned. If the most significant bit of the return value is set, it indicates that more data was available than would fit in the allocated buffer space, (defined by MaxValues).

Aggregates

int NumberOfHDAAggregates (HANDLE hConnect);

```
BOOL  GetHDAAggregateDescr (HANDLE    hConnect,
                            int       index,
                            DWORD     *pAggrId,
                            char      *pNameBuf,
                            int       nBufSize,
                            char      *pDescBuf,
                            int       dBufSize);
```

These two functions may be used to obtain the list of HDA Item Aggregates from the connected server. NumberOfHDAAggregates returns the number of aggregates supported by the server and the description of each aggregate may be obtained by calling GetHDAAggregateDescr. This function returns the aggregate name, description, and Id in the user supplied buffers. TRUE is returned if the function is successful, otherwise FALSE.

```
DWORD  ReadHDAProcessedItemValues (HANDLE        hConnect,
                                   HANDLE        hItem,
                                   DWORD         AggregateID,
                                   FILETIME      Start,
                                   FILETIME      End,
                                   FILETIME      ReSampleInterval,
                                   DWORD         MaxValues,
                                   FILETIME      *pTimeStamps,
                                   VARIANT       *pValues,
                                   DWORD         *pQualities);
```

The client application may obtain calculated values from the server by calling ReadHDAProcessedItemValues with the selected item and aggregate identifications. Start and End times specify the time period of interest and ReSampleInterval determines the individual time periods over which to perform the calculations. Values returned from the server will be stored in the user-supplied buffers, (pTimeStamps, pValues, & pQualities), up to the maximum number of values specified by MaxValues. The return value for this function returns the actual number of values returned. If the most significant bit of the return value is set, it indicates that more data was available than would fit in the allocated buffer space, (defined by MaxValues).

Server Status

```
BOOL GetHDASvrStatus (HANDLE        hConnect,
                      WORD          *pStatus,
                      FILETIME      *pCurrentTime,
                      FILETIME      *pStartTime,
                      WORD          *pwMajorVersion,
                      WORD          *pwMinorVersion,
                      WORD          *pwBuildNumber,
                      DWORD         *pdwMaxReturnValues,
                      char          *pStatusString,
                      int           StatusBufSize,
                      char          *pVendorInfo,
                      int           VendorInfoBufSize);
```

GetHDASvrStatus returns status values from the connected server. Values returned from the server are stored in the locations provided by the user. This function returns TRUE if successful.

Client Application Callbacks

WtHDAClient.DLL uses callbacks to notify the controlling application of events occurring with the associated connection. The application must explicitly enable each callback by providing the address of an appropriate handler function.

BOOL EnableHDAErrorNotification (HDAERRORPROC lpCallback);

void CALLBACK EXPORT HDAErrorMsgCallback (DWORD hResult, char *pMsg);

The error notification callback is used to pass control to the application if an error is encountered within the WtHDAClient dll. If this callback is not enabled, the DLL will display an error dialog box that must be acknowledged by the user.

BOOL EnableHDAShutdownNotification (HANDLE hConnect, HDASHUTDOWNPROC lpCallback);

void CALLBACK EXPORT HDAShutdownCallback (HANDLE hConnect);

The shutdown notification callback is used to pass control to the application if an attached server requests a disconnect. The application should respond to this notification by releasing all server resources and performing a clean disconnect.

BOOL EnableHDAEventMsgs (HDAEVENTMSGPROC lpCallback);

void CALLBACK EXPORT HDAEventsCallback (char *pMsg);

The event notification callback is used to pass control to the application during normal program execution to provide some feedback of activity taking place on a connection. These messages may be useful for debugging a connection and include text descriptions of each HDA Interface as it is executed.

HDAClientTst MFC Example

The HDAClientTst application was designed to demonstrate how an OPC HDA Client application may be assembled using the WtHDAClient dll. HDAClientTst was built using MSDEV Version 6.0. All project and resource files are contained in the distribution file HDAClientTst.zip.

This simple application uses the standard MFC framework, (Application/Document/View), architecture generated by the Microsoft development platform for a multiple-document interface. In this example, only one HDA Server connection is allowed at a time. Menu options allow you to select a server, establish a connection, browse the list of available items, and read historical data. Attribute and Aggregate values may also be obtained from a connected HDA Server.