LG Electronics Institute of Technology

iData EAC Toolkit 4000
Sample Programs Manual

(Version 1.04)

iData EAC Toolkit 4000 Sample Programs Manual

Revision History Page

Description and reference of change	Date	Revision
Initial Preparation	25th July, 2006	1.0

LG Electronics Institute of Technology
16 Woomyeon-Dong, Seocho-Gu, Seoul 137-724, Korea
tel: +82-2-526-4161 fax: +82-2-526-4165

Table of Contents

1. Introduction

iData EAC Toolkit 4000 provides an application program interface to perform common tasks of Enrollment and Identification. This encapsulates calls to different components involved in Enrollment and Identification process. These APIs will allow any external application to integrate and interact with the current IrisAccess system.

This manual explains the various sample applications available with the iData EAC Toolkit 4000 software. It is assumed that the user has read through the User Manual shipped with iData EAC Toolkit 4000 for detailed description of functionality provided by each API.

The sample programs - **Sample_VB, Sample_CSharp and Sample_VC -** provided with iData EAC Toolkit 4000 describe the usage of the API from different programming languages. ***It is strongly recommended to follow the sample programs for your application program.***

2. Pre –Requisites

- Pentium compatible PC must have Net framework v1.1.4222 or higher.
- EAC Software version 3.00

3. Overview

iData EAC Toolkit 4000 connects to IrisServer on the IrisEnroll port. Hence only one instance of IrisEnroll should be configured with the IrisManager with correct values. The steps to configure IrisEnroll with IrisManager are:

1. Execute **IrisManager** application supplied with EAC Software V3.00 and above.
2. Click on "**Creation**" menu and then click on "**IrisEnroll**" menu item.
3. Once the Program Management window opens in "**IrisEnroll**" mode, click on the "**New**" button to register an IrisEnroll on the machine.
4. Fill the **Name**, **IP Address** of the machine where the sample application will run, **Security Id** using which the **IrisEnroll** client will be recognized by the **IrisServer**.

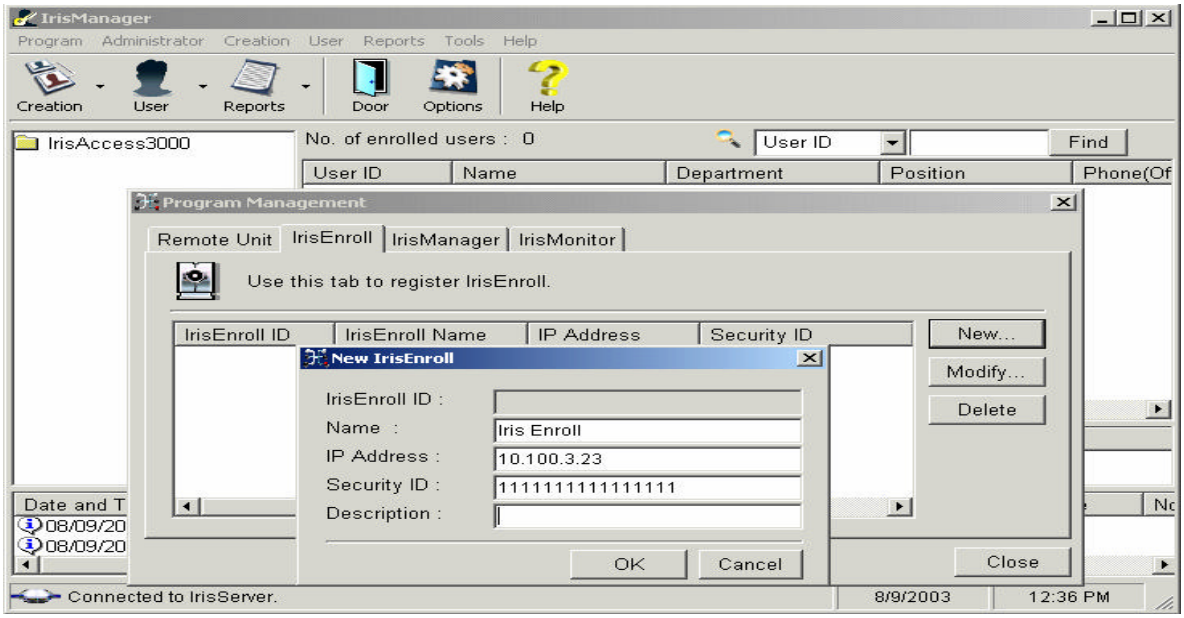

Figure 1 – New registration of IrisEnroll

Once IrisEnroll is configured, sample application can be executed to test APIs. To use EnrollUser API, it is required a RemoteGroup and TimeGroup named "All" in the IrisManager otherwise this API will fail. Configuring at least one remote unit in the IrisManager will create RemoteGroup and TimeGroup with "All" value.

It should be noted that only one IrisEnroll instance could be registered with the IrisManager from a particular IP address. If your machine (IP address) has already been configured for an IrisEnroll client with IrisManager, all sample applications can make use of the same registration details. Supply the same Security Id used for registering the IrisEnroll when connecting to the IrisServer using the sample applications.

The iData EAC Toolkit 4000 consists of two interfaces namely, IEacApi and IEacApiEvents. While IEacApi contains methods to connect to IrisServer and performs enrollment or identification, IEacApiEvents interface contains event handlers for the events raised by the API. The samples documented in the manual not only display the use of each API but also handle the events raised by the API to demonstrate the use of these events.

4. Samples

This section describes the various samples provided with iData EAC Toolkit 4000 and also steps to implement clients in commonly used languages namely C#, VB and VC++. For every language described here, the description consists of language specific steps that should be performed while developing a client application for iData EAC Toolkit 4000. Please refer the samples shipped with iData EAC Toolkit 4000 software for information on how each client is implemented.

Figure 2 illustrates the UI that is displayed on executing Sample_Csharp. UI for Sample_VB and Sample_VC are similar to that of Sample_CSharp. Most of the buttons on the UI represents an API call. Event handlers for each of these contain code for the making a call to the corresponding API.

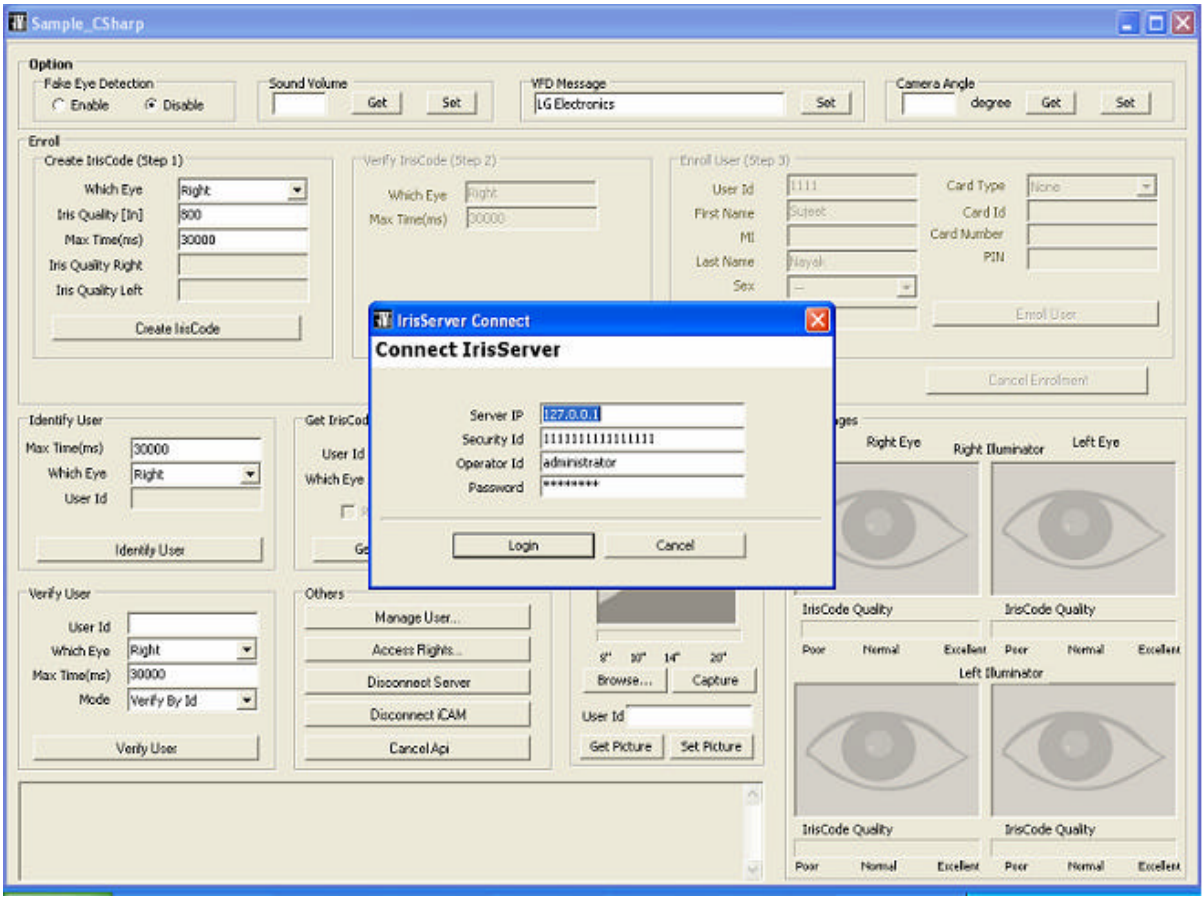

Figure 2 – IrisServer connection screen.

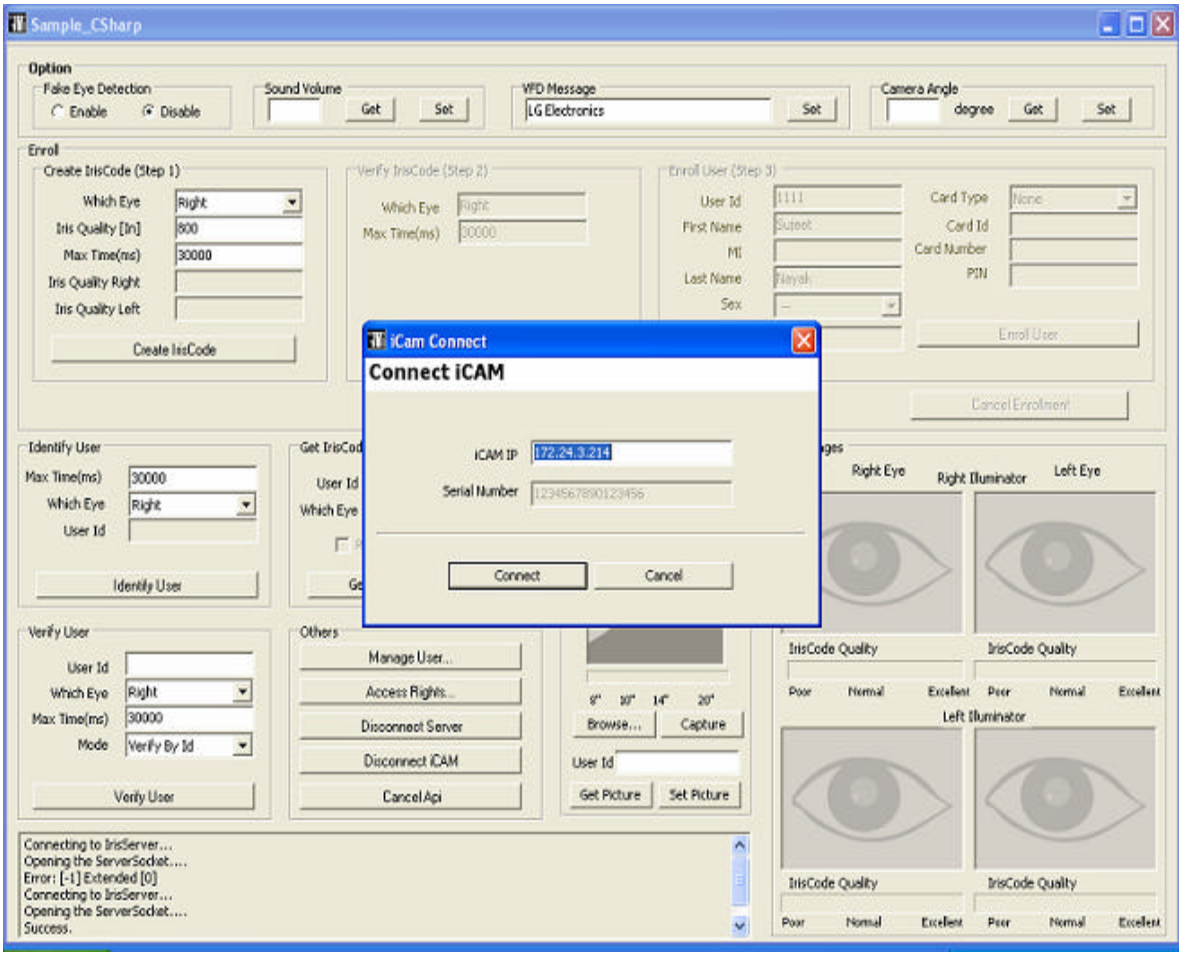

Figure 3 - iCAM connection screen.

User input is basically the parameters that are required for the execution of the API.

All out parameters of an API call are displayed back on the UI with the exception of IrisCode, which gets generated from calls to CreateIrisCode and GetIrisCodeFromServer APIs. IrisCode returned by these APIs is stored in a member variable and passed as a parameter during calls to EnrollUser and VerifyIrisCode API.

Every sample application handles OnGetError, OnGetStatus and OnGetCamStatus, OnGetFaceImage, OnGetIrisImage, OnServerDisconnect events raised by the API. While messages received in the OnGetStatus and OnGetError are displayed in the read only textbox, picture panel on the right displays the image received in OnGetIrisImage event.

Messages received in OnGetStatus is displayed in the format:
MessageReceivedFromAPI

Messages received in OnGetError is displayed in the format:
Error: [ErrorCode] Extended: [Extended Error Code] MessageReceivedFromAPI

The requirement of IrisServer and/or iCAM for calling different APIs available are shown below.

No	Api	IrisServer	Camera
1	AddAccessRightToUser	YES	NO
2	Cancel	NO	YES
3	CaptureRealTimeImages	YES	YES
4	ConnectToCam	YES	NO
5	ConnectToServer	NO	NO
6	CreateIrisCode	YES	YES
7	DeleteAccessRightFromUser	YES	NO
8	DeleteUser	YES	NO
9	DisconnectCam	NO	YES
10	DisconnectServer	YES	NO
11	EnrollUser	YES	YES
12	GetAccessRight	YES	NO
13	GetAccessRightNumber	YES	NO
14	GetCamAngleFunction	NO	YES
15	GetExtendedError	NO	NO
16	GetFacePicture	YES	NO
17	GetFakeEyeDetection	YES	YES
18	GetGroup	YES	NO
19	GetGroupNumber	YES	NO
20	GetIrisCodeFromServer	YES	NO
21	GetVolume	YES	YES
22	GetUserInfo	YES	NO
24	IdentifyUser	YES	YES
24	IsCamConnected	NO	NO
25	IsServerConnected	NO	NO
26	ModifyUser	YES	NO
27	SetCamAngle	YES	YES
28	SetFacePicture	YES	NO
29	SetFakeEyeDetection	YES	YES
30	SetVolume	YES	YES
31	SetVFD	YES	YES
32	StopRealTimeImages	NO	YES
33	VerifyIrisCode	YES	YES
34	VerifyUserByPin	YES	YES
35	VerifyUser	YES	YES

In case where both IrisServer and iCAM are connected, all buttons excluding "Verify", "Enroll" and "Cancel Enrollment" get enabled. Enroll process can be initiated by clicking "Create IrisCode" button which will in turn, disable the buttons which require iCAM. Every step in the Enroll will enable the button of the next step. Clicking on the "Cancel Enrollment" during any enroll operation will exit enroll process and main screen will return to its default state.

To connect IrisServer, pass values associated with the IrisEnroll while configuring the IrisEnroll instance with the IrisManager, as parameters. Since the current IrisServer supports only one

IrisEnroll to be connected from a PC, sample application will fail to run if a sample IrisEnroll application provided with EAC Software V3.00 and above is already connected.

Selecting "Enable" or "Disable" radio buttons in "Fake Eye Detection" can enable or disable fake eye detection.

To set the volume of the messages rendered by the iCAM, enter volume level between 0 and 8 and click "Set" in "Sound Volume". To get the current volume level of iCAM, click "Get" in "Sound Volume".

Vfd message can be set clicking "Set" in "VFD Message".

The steps to perform enrollment are:

1. Select **Right/ Left/Both eye** in CreateIrisCode group box, and click on **CreateIrisCode** button.
2. Verify the IrisCode generated using "**Verify IrisCode"** button in "Verify IrisCode" group.
3. In "Enroll User" group , enter the **User Id** and other mandatory fields of a new user and click on "**Enroll User"** button. It should be noted that a user is enrolled with "All" Remote and Time group access rights. Hence the user can pass through all the remote groups.
4. At any point the user can cancel enrollment process, by clicking on the "**Cancel Enrollment"** button.

To verify a user using his User Id, select "Verify by Id" option "Verify User" in group box and then enter the User Id of the user to be verified. Click on the "Verify User" button.

To verify a user using PIN , select "Verify by PIN" in "Verify User" group and then click the "Verify User"**,** now enter the PIN on the iCAM keypad..

To Identify a user, select the eye in "Identify User" and then click on "Identify User" button.

Click on "Get IrisCode From Server" button after entering the User ID and which eye to retrieve the IrisCode(s) of a user.

Clicking on "Capture" in "Picture", captures the live images within the range of the iCAM and displayed on the UI.

"Cancel Api" button cancels the current operation being performed.

Face picture can be set/retreived from the server by entering the User Id in the text box provided and on clicking "Set Picture" or "Get Picture" buttons. Face picture can be loaded from disk using "Browse…" button.

To delete a user from the IrisServer database, enter the User ID of the user and click "Delete User" button.

"UpdateUserInfo…" allows modifying user information of an already registered user. The following UI gets dispalyed. Clicking on "GetUserInfo" button will get the User details for the specified User ID and displays the User details in the UI shown below:

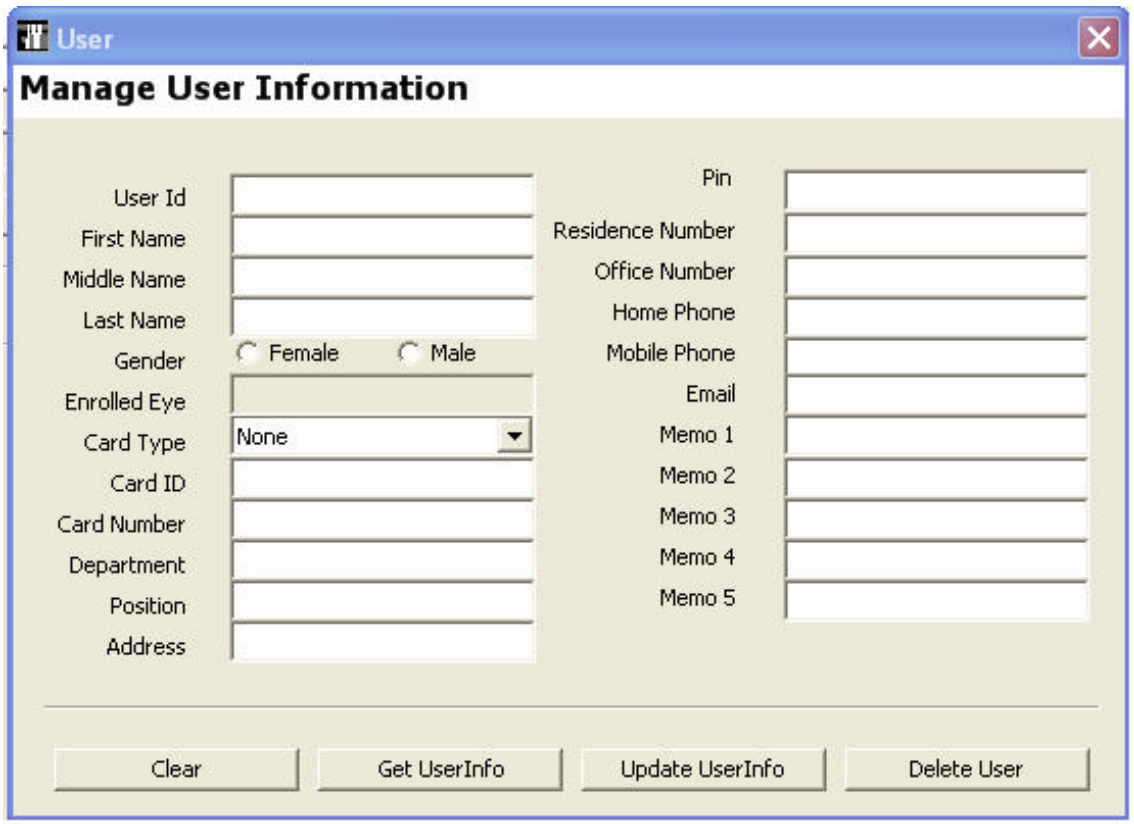

Figure 4- Manage User Information screen.

The Clear button clears the displayed user information from the UI.

"Assign Rights…" button allows viewing of available RemoteGroups and TimeGroups in the system. It also facilitates assignment of TimeGroup and RemoteGroup to an existing user. The following UI gets displayed.

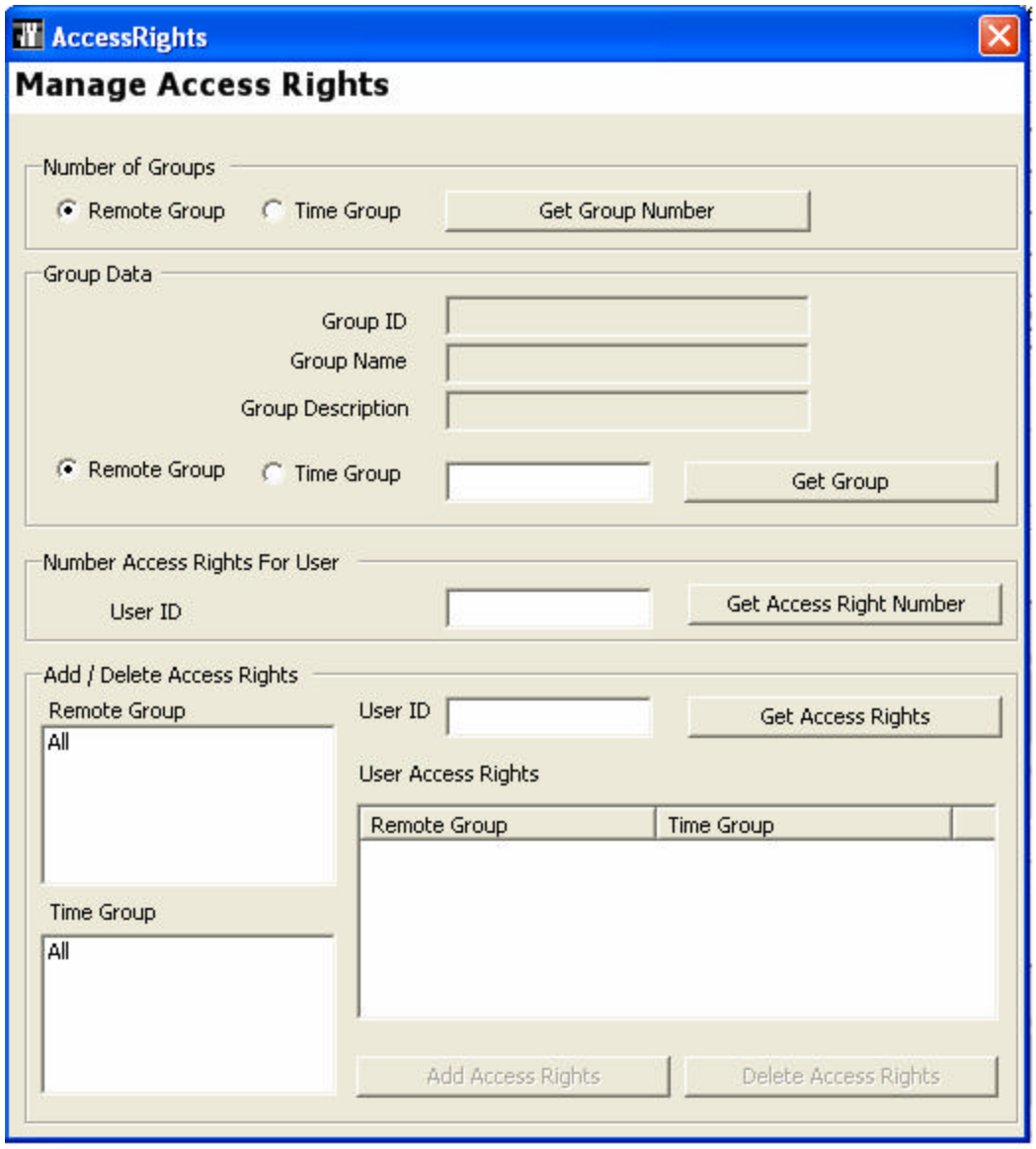

Figure 5 – Manage Access Rights screen

4.1. C# Clients

A C# client does not require any special installation of the iData EAC Toolkit 4000 software. The client should just add a reference to the iDataToolkit4000.dll and include the namespace using "using iDataToolkit4000". Once this is done, iData EAC Toolkit 4000 can be used in the code.

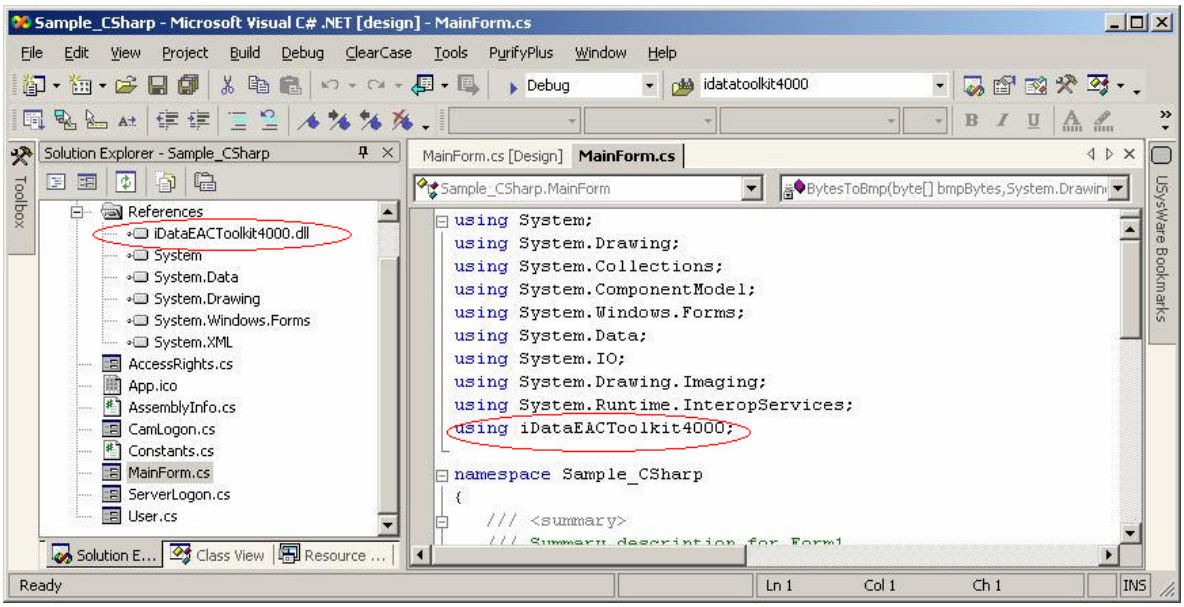

Figure 6 – Add reference to C# project.

Event Handling

To trap the events raised by the API, event handlers should be associated to the object of EacApi class. Figure 7 shows creation of the respective delegates for each of the events that is required and assigning the delegates as event handlers to IrisEnroll variable which is an object of EacApi class.

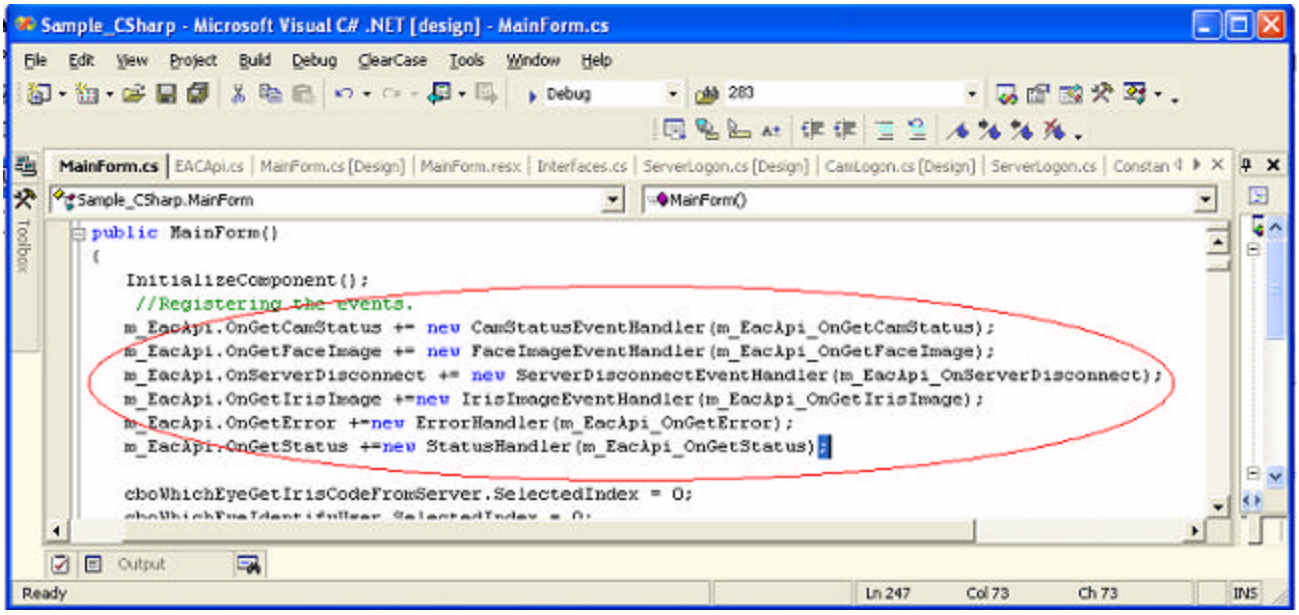

Figure 7 – Event handling in C# application.

4.2. Visual Basic Clients

A VB client is implemented like any other COM Client. Add the reference to the registered iDataEACToolkit4000.dll in the VB client application by selecting the **References** menu item under the **Project** menu of the VB IDE can do this. In the references window that opens, select iDataEACToolkit4000 and click on OK. Figure 8 shows the selection of iDataEACToolkit4000 in the references dialog box.

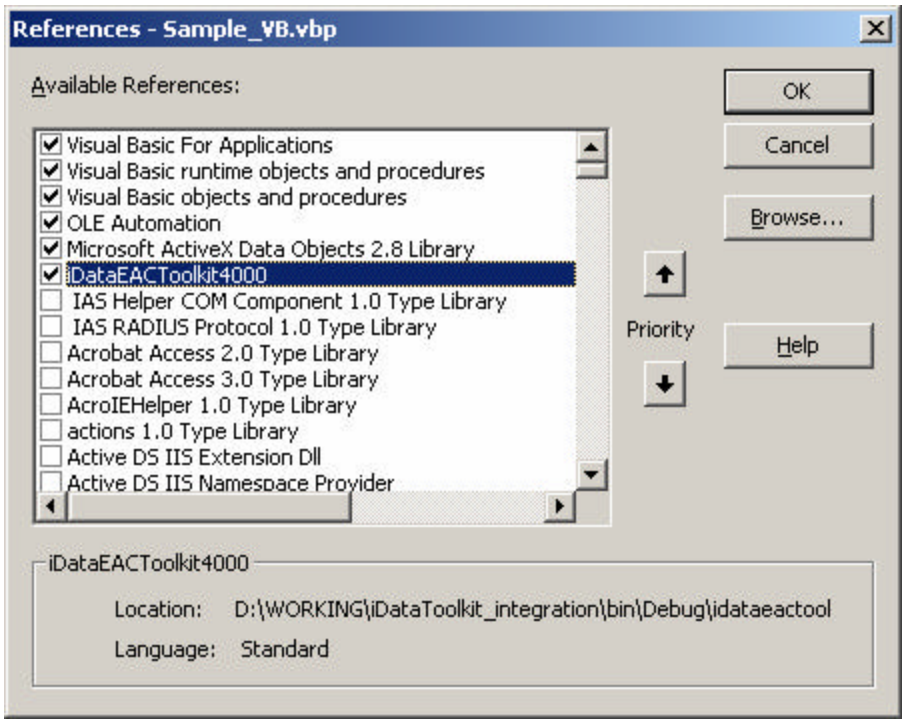

Figure 8 – Add reference to VB project

Event Handling

In order to handle events raised by the API, event handlers should be defined and associated with an object of EacApi. "WithEvents" keyword of VB is used for early binding to the component (in this case iDataEACToolkit4000.EacApi) in order to receive events.

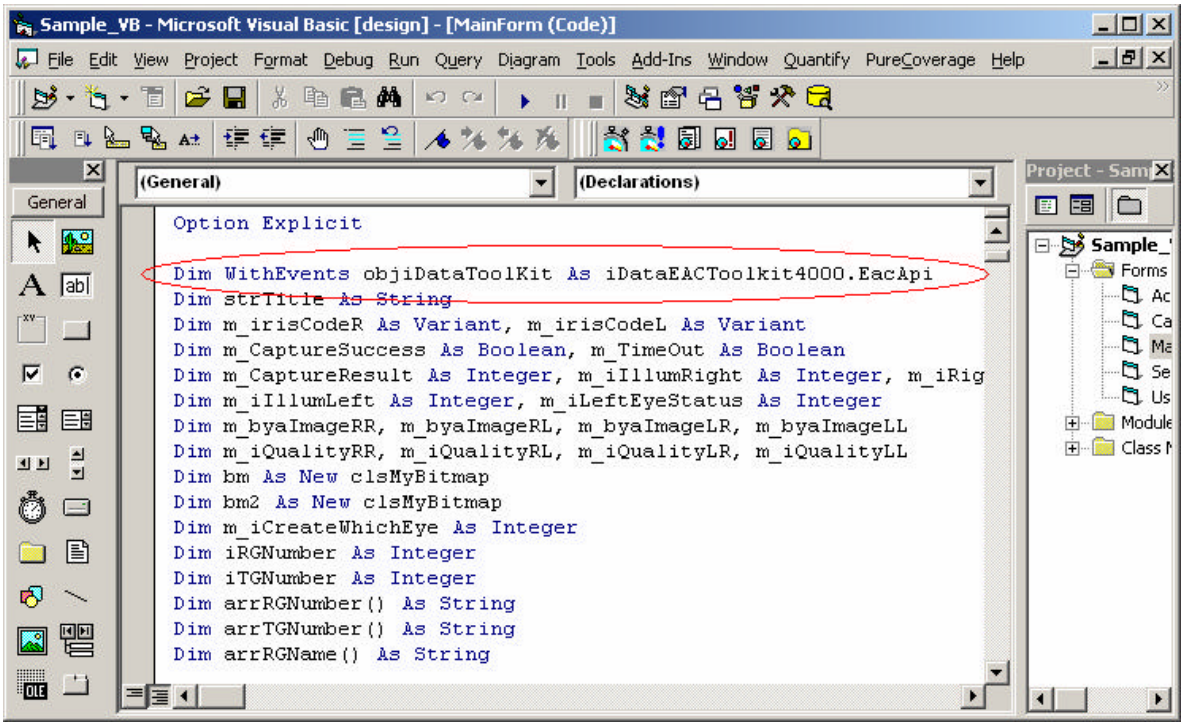

Figure 9 – Event handling in VB application.

Figure 10 shows the creation of an object of EacApi using "WithEvents" keyword to indicate that the client will receive and handle events. To associate handlers with their respective events, event handler naming convention is the same as any other event handlers in VB is followed – objectName_EventName. Figure 11 shows the event handler for the OnGetIrisImage event raised by the API where objiDataToolkit4000 is variable name for the EacApi object created (refer Figure 7) and OnGetIrisImage is the event raised by the API. The data types of method parameters should match the parameters raised by the event.

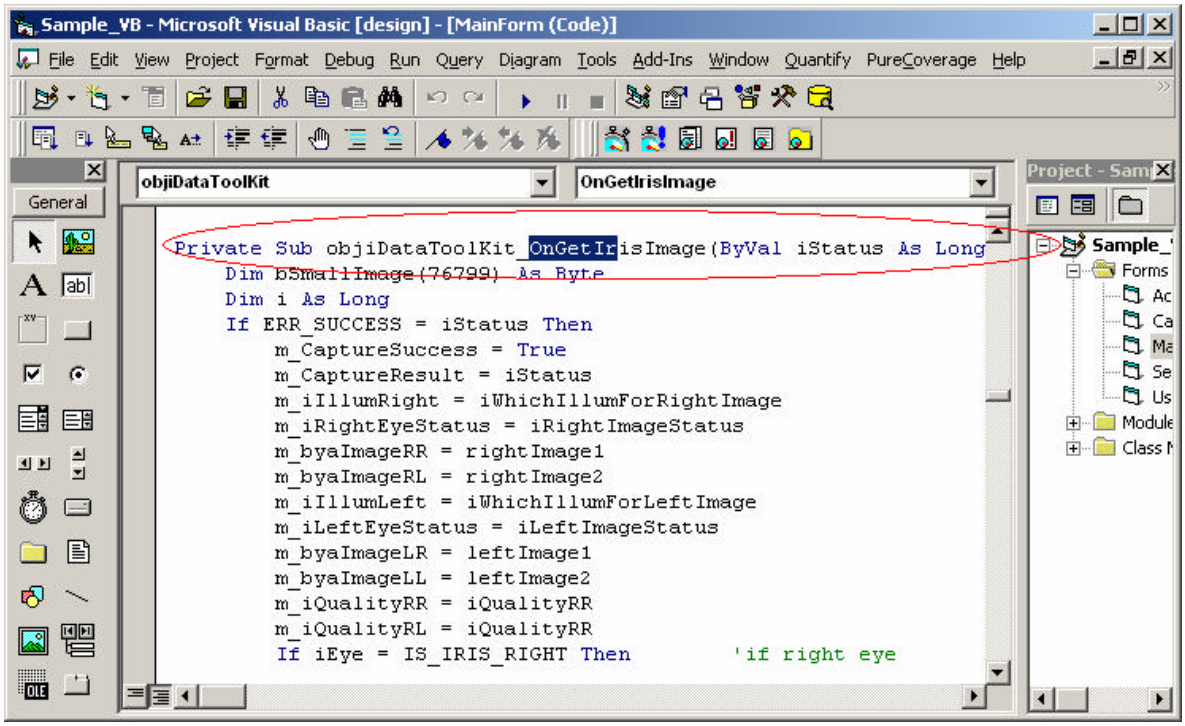

Figure 10 – Event handler for OnGetIrisImage

The signatures of event handler implemented in the VB are as:

OnGetStatus - ByVal strMsg As String
OnGetError - ByVal iErrorCode As Long, ByVal strDescription As String
OnGetImage - ByVal Image As Variant
OnGetFaceImage – ByVal Image as Variant

4.3. Visual C++ Clients

A VC++ client is implemented like any VC++ COM Client. Import the supplied iDataEACToolkit4000.tlb file into the client application using the #import statement. IEacApi interface pointer once instantiated using CoCreateInstance, can be used make calls to the iData EACToolkit4000. Figure 11 #import statement used to import iDataToolkit4000.tlb present within the debug directory of the workspace.

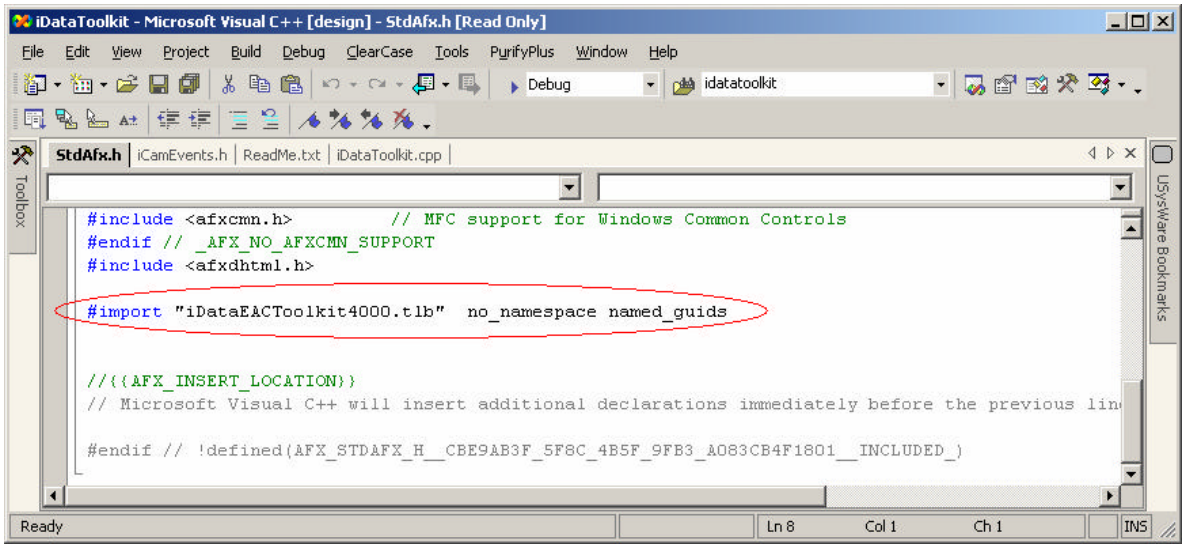

Figure 11- Add reference in VC++ application.

To trap events raised by iData EAC Toolkit 4000, the client application should create a class that implements IEacApiEvents (an outgoing interface in the iData Toolkit) interface methods along with the IUnknown interface methods. IEacApiEvents interface methods that need to be implemented are:

HRESULT STDMETHODCALLTYPE OnGetError (long, long, BSTR);
HRESULT STDMETHODCALLTYPE OnGetStatus (BSTR);
HRESULT STDMETHODCALLTYPE OnGetCamStatus (long, long, long, VARIANT);
HRESULT STDMETHODCALLTYPE OnGetFaceImage (long, VARIANT, long);
HRESULT STDMETHODCALLTYPE OnServerDisconnect ();
HRESULT STDMETHODCALLTYPE OnGetIrisImage (long, long, long, long, long,
VARIANT,long,VARIANT,long, long, long, VARIANT , long, VARIANT,long);

Figure 12 shows a sink called CiCamEvents implementing the IEacApiEvents interface.

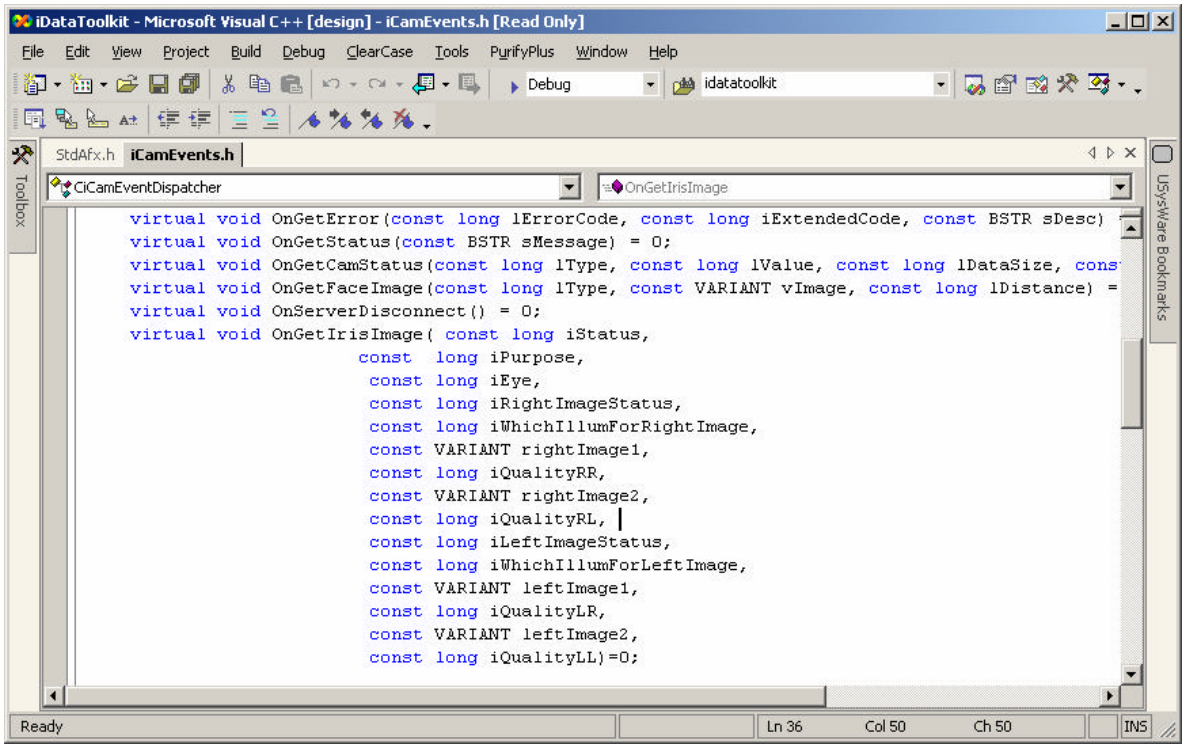

Figure 12 – ATL Event Sink for VC++ application.

Once the sink is implemented, it has to be associated with the iData Toolkit, for handling any event raised by the API. Figure 13 illustrates the code snippet for retrieving the connection point for IEacApiEvents interface and establishing a connection between the connection point and sink object. In the code,. Sink object here is an object of CiCamEvents class. Client application should ensure to invoke the clean up methods like Release, Unadvise.

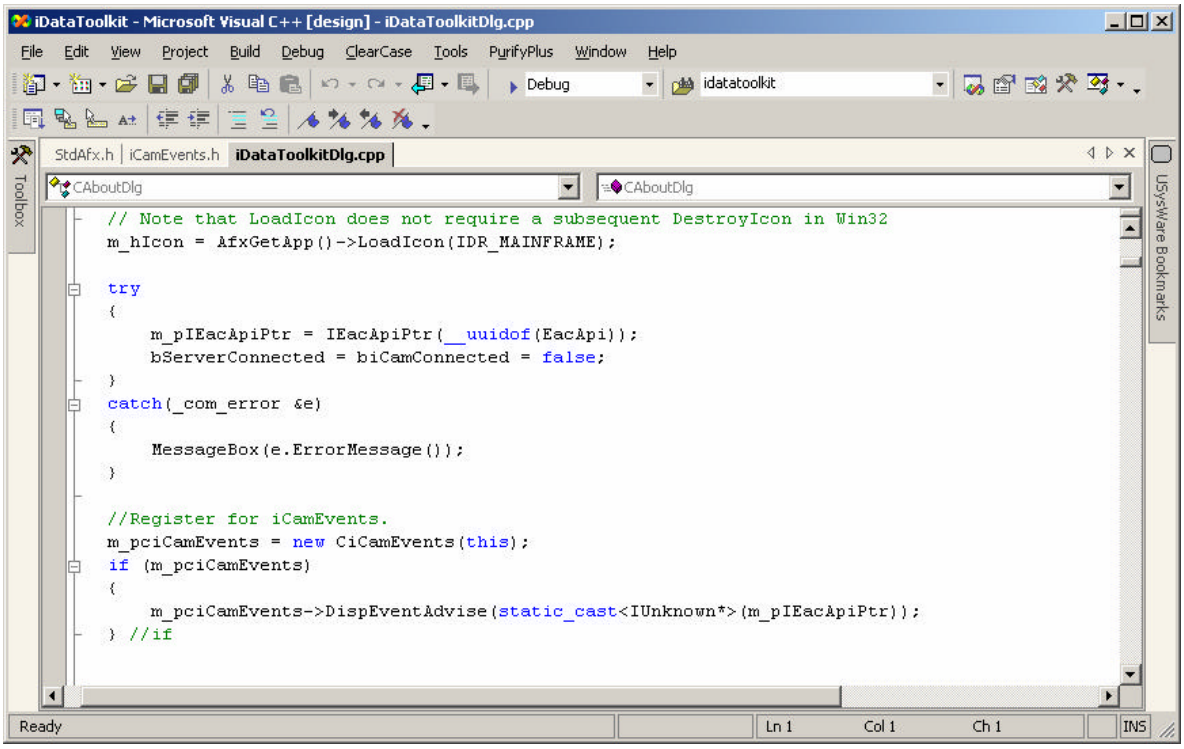

Figure 13 – Setting up event handlers.

5. Reference

- **iData Toolkit User Manual**
- **Code in the sample applications**