Iris ID

iCAM TD100 SDK User Manual

Version 2.05.x

April, 2012

iCAM TD100 SDK User Manual

Iris ID Systems, Inc., 7 Clarke Drive, Cranbury, New Jersey 08512, USA.

Document No. IRISIDTD100-01-0205-0412

Last Update: April, 2012

Table of Contents

1. Introduction

Since 1997, Iris ID Systems, Inc., formerly a division of LG Electronics U.S.A. has been the key developer and driver of the commercialization of iris recognition technology. IrisAccess®, now in a fourth generation, is the world's leading deployed iris recognition platform. Found on 6 continents, in thousands of locations, authenticating the identities of millions and millions of persons, more people in more places authenticate with Iris ID IrisAccess® than with all other iris recognition products combined. Through our expertise and Advanced Identity Authentication, Iris ID (formerly LG IRIS) helps add security, convenience, privacy, and productivity to the enterprise operation you wish to improve.

Traditional Notions of Establishing Identity

Historically, identity or authentication conventions were based on things one possessed (a key, a passport, or identity credential), or something one knew (a password, the answer to a question, or a PIN.) This possession or knowledge was generally all that was required to confirm identity or confer privileges. However, these conventions could be compromised - as possession of a token or the requisite knowledge by the wrong individual could, and still does, lead to security breaches.

Biometric Appeal of Iris Recognition

Of all the biometric technologies used for human authentication today, it is generally conceded that iris recognition is the most accurate. Coupling this high confidence authentication with factors like outlier group size, speed, usage/human factors, platform versatility and flexibility for use in identification or verification modes - as well as addressing issues like database size/management and privacy concerns - iris recognition has also shown to be exceedingly versatile and suited for large population applications.

Benefits:

1. The smallest outlier population of all biometrics - Few people cannot use the technology, as most individuals have at least one eye. In a few instances even blind persons have used iris recognition successfully, as the technology is iris pattern-dependent, not sight dependent.
2. Iris pattern and structure exhibit long-term stability. Structural formation in the human iris is fixed from about one year in age and remains constant (barring trauma, certain rare diseases, or possible change from special some ophthalmologic surgical procedures) over time. So, once an individual is enrolled, re-enrollment requirements are infrequent. With other biometric technologies, changes in voice timbre, weight, hairstyle, finger or hand size, cuts or even the effect of manual labor can trigger the need for re-enrollment.
3. Ideal for Handling Large Databases. Iris recognition is the only biometric authentication technology designed to work in the 1-n or exhaustive search mode. This makes it ideal for handling applications requiring management of large user groups, such as a National Documentation application might require. Large databases are accommodated without degradation in authentication accuracy. Iris ID IrisAccess® platforms integrate well with large database back ends like Microsoft SQL and Oracle 9i.
4. Unmatched Search Speed in the one to many search mode is unmatched by any other technology, and is limited not by database size, but by hardware selected for server management. In a UK Government-commissioned study, Iris ID's IrisAccess® platform searched records nearly 20 times faster than the next fastest technology. Iris ID has developed a high speed matching engine, IrisAccelerator™, designed to deliver 10 million+ matches per second.
5. Versatile for the One to Many, One to One, Wiegand and Token Environments. While initially designed to work in one-to-many search mode, iris recognition works well in 1-1 matching, or verification mode, making the technology ideal for use in multifactor authentication environments

where PINs, or tokens like proximity (prox) or smartcards are used. In a token environment, many privacy issues related to biometric database management are moot, as the user retains control of biometric data – a small template of 512 bytes per iris.

6. Safety and Security Measures in Place. Iris recognition involves nothing more than taking a digital picture of the iris pattern (from video), and recreating an encrypted digital template of that pattern. 512-byte iris templates are encrypted and cannot be re-engineered or reconstituted to produce any sort of visual image. Iris recognition therefore affords high level defense against identity theft, a rapidly growing crime. The imaging process involves no lasers or bright lights and authentication is essentially non-contact.
7. Convenient, Intuitive User Interface. Using the technology is an almost intuitive experience, requiring relatively little cooperation from subjects. Proximity sensors activate the equipment, which incorporates mirror-assisted alignment functionality. Audio auto-positioning prompts, automated image capture, and visual and audio authentication decision-cueing completes the process.

1.1 Product Overview

Biometric iris recognition is exemplified in the iCAM TD100. A Hand-held tethered device – Iris ID has developed a series of algorithms that provide the capability to capture iris images while either the person or the device is in motion. The "iris in motion" capability helps to realize new horizons in market applications for the technology.

High Speed - Dual Iris Capture

The iCAM TD100 includes an optical system specifically designed and optimized to operate in perfect unison with the integrated high speed multi-sensor iris imager array. The iCAM TD100 automatically processes and outputs high quality ISO standards compliant iris images of a subject in less than one second as the device or subject approaches the optimum capture distance.

Product Highlights

- High Speed Dual Iris Capture
- Compact and Lightweight
- Single Motion Automatic Iris and Face Capture
- Intuitive Operator Guidance System
- Standards Compliant Hardware and Software

Iris Image Capture Process

Fully automatic dual iris image capture and quality analysis routines are available as a part of the iCAM TD100 SDK set for the field application of the iCAM TD100. An illustration of the iris capture GUI screen is shown below. The iData SDK run-time license for iris enrollment and quality assessment is available for use with the iCAM TD100 SDK module subsystem. Iris and face capture are performed by the operator extending their arm from the face capture distance to the iris capture distance as illustrated below.

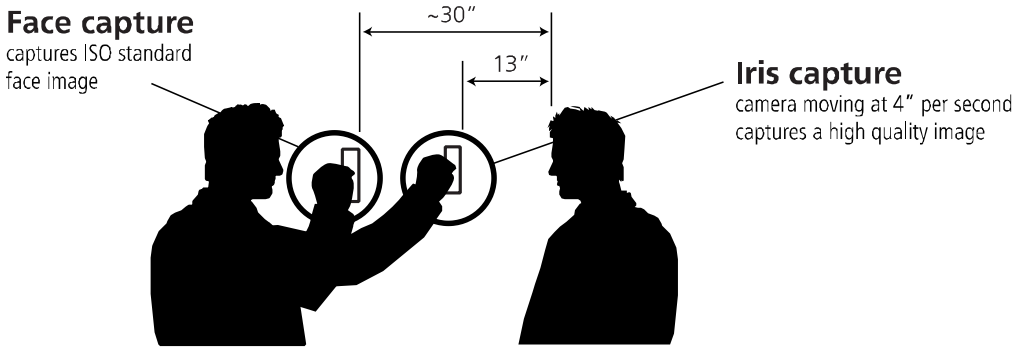


Fig. 1 Operating Range of Iris Capture and Face Capture

Fig. 2 Iris Image Capture Process

Face Image Capture

* *Note: <u>The face capture API function is included in the SDK.</u>*

- The integrated framing function provides feedback for the capture of a properly formatted ISO/ICAO face image.
- Manual face capture with auto focus is also possible through the camera calls in the iCAM TD100 SDK sample application.
- An application developer can also use host based face finding to trigger the face capture automatically from the host processor.
- Face capture can be initiated through API or via the shutter button on the iCAM TD100.
- Sample illustrations of face capture modes are shown in Fig. 3.

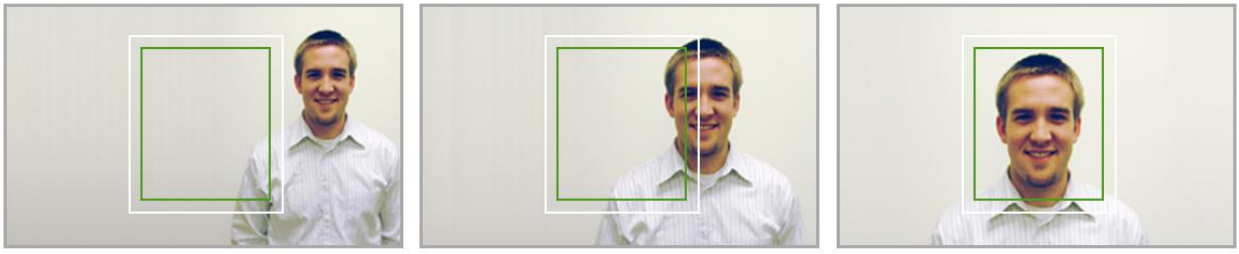


Fig. 3 Face Image Capture Process

iCAM TD100 SDK

The iCAM TD100 SDK is an API to enable functions of the iCAM TD100 device. The development environment and functionality of the SDK for the iCAM TD100 module closely mimic those of the widely deployed iData SDK's in terms of device control. Application developers familiar with the IrisAccess® API's in the iData SDK will find integration to be similar to previous implementations. The sample application provided with the iCAM TD100 SDK is provided in C++ and C#.

The iCAM TD100 SDK will only provide iris enrollment and matching if the iData SDK is installed in addition to the iCAM TD100 SDK.

**Note: A separate product, the iData SDK is used for matching and for image processing. The TD100 SDK sample applications are designed to show the usage of the iCAM TD100 and allow for demonstration of available functions in the TD100 SDK with the ability for image capture (that can be saved and viewed from the PC).*

Purpose and Audience for this Guide

Read this document before attempting to install, configure, expand, run, or modify the product that has been provided from Iris ID Systems, Inc.

This guide is intended to be used as a reference for iCAM TD100 with use of the SDK software and its accessories. This document includes background on the product technology, product purpose, initial installation instruction, and configurations, SDK software setup, additional recommended accessories, Troubleshooting-frequently asked questions, as well as a several of sample application models to assist in setup/configuration and use of this device. It is the most complete reference available once you setup and begin using your iCAM TD100 with the available SDK system.

Reference Materials

In addition to this guide, your box should contain an "iCAM TD100 Quick-Start Operations Guide" designed to streamline the initial setup of your iCAM TD100 product.

** Note: Additional reference, amendments and updated documentation material may become available directly from the http://www.irisid.com website. Check the site for updated information, frequently asked questions, and tips to be used with your product.*

2. Package Contents of iCAM TD100

About the Software

The iCAM TD100 SDK provides a device driver and API for use with the iCAM TD100 camera unit.

Installation CD

An installation CD is included in the iCAM TD100 package. You will find device and SDK documentation as well as a Windows software installation package which contains the iCAM TD100 USB device driver and sample applications.

AC Adapter

The iCAM TD100 requires power from the included AC adapter. Most USB ports cannot supply enough power to the iCAM TD100. The iCAM TD100 requires slightly less than one amp @ 5VDC to operate its infrared illuminators.

2.1 Preparing for Installation

**Note:* Verify that all contents provided by Iris ID and the equipment not provided by Iris ID meets the below equipment specification before performing any part of the installation process.

Items Required for Use of This Product

Required items provided by Iris ID:

- iCAM TD100 SDK software CD
- USB + Power cable
- Power Adapter & Power cord
- iData SDK – Iris matching & quality assessment SDK (optional)

Required Equipment (not provided by Iris ID):

- Windows based PC (Windows XP (32-bit) / Windows 7 (32-bit) / Windows Vista (32-bit) / Windows 7 (64-bit))

***Note:** *64-bit* versions of *Windows Vista are not compatible with the latest version iData TD100 SDK software.*

Minimum Computer requirements:

- Microsoft Windows XP (32-bit), Windows Vista (32-bit), Windows 7 (32-bit), or Windows 7 (64-bit) Operating System
- 512MB RAM (or higher)
- x86 Processor, 2.0 GHz (or higher)
- 2GB available HDD space or above
- CD-ROM Drive (for software & driver installation)
- Microsoft .NET Framework ver. 3.5
- Mouse, SVGA Monitor, Keyboard

- Dedicated USB 2.0 port

*****Note:*** *An available and dedicated USB 2.0 compliant port is needed to properly use the iCAM TD100.*

3. iCAM TD100 Hardware Installation

Please refer to the iCAM TD100 Quick-Start Operations Guide (included with all iCAM TD100 camera units – also available from the iCAM TD100 SDK Software CD).

4. Initial Configuration of iCAM TD100

4.1 Software & USB driver installation of iCAM TD100 for x86 Users (32-bit) only for Win XP, Win Vista & Win 7 (32-bit OS versions)

***Note:** For the latest software version of iCAM TD100 SDK, visit* http://www.irisid.com/software.
 *Attention: DO NOT CONNECT THE USB CABLE OF THE iCAM TD100 DEVICE TO THE COMPUTER BEFORE INSTALLING THE SOFTWARE.

Installing the Software (For Win XP, Win Vista and Win 7 (32-bit) x86 users only)

The following steps provide information on how to install the software application with a Windows XP (32-bit), Win Vista (32-bit) or Windows 7 (32-bit) Operating System. The driver for the iCAM TD100 is automatically installed during the installation process. If using a Windows 7 x64 (64-bit) OS, skip forward to section 4.2 of this manual.

1. Insert iCAM TD100 SDK disc into CD-ROM drive

2. Open the iCAM TD100 SDK _x86 Setup folder from the CD-ROM Drive

3. Open the Setup folder

3. Select the Setup.exe Setup launcher icon

4. Select >Next at the Welcome to InstallShield Wizard for iCAM TD100 SDK

5. Select Radio button for License agreement and press >Next

6. Select destination folder to install and press >Next

7. Select >Next when the "Start copying files - Review settings before copying files" screen appears. (Wait until software installs. This process may take several minutes.)

8. Select >Finish to complete installation of software successfully.

Note: Your drivers are digitially signed by Microsoft. However, in the event that a Windows Security warning dialogue box appears as shown in Fig. 1, select **Install this driver software anyway to continue the driver installation process.*

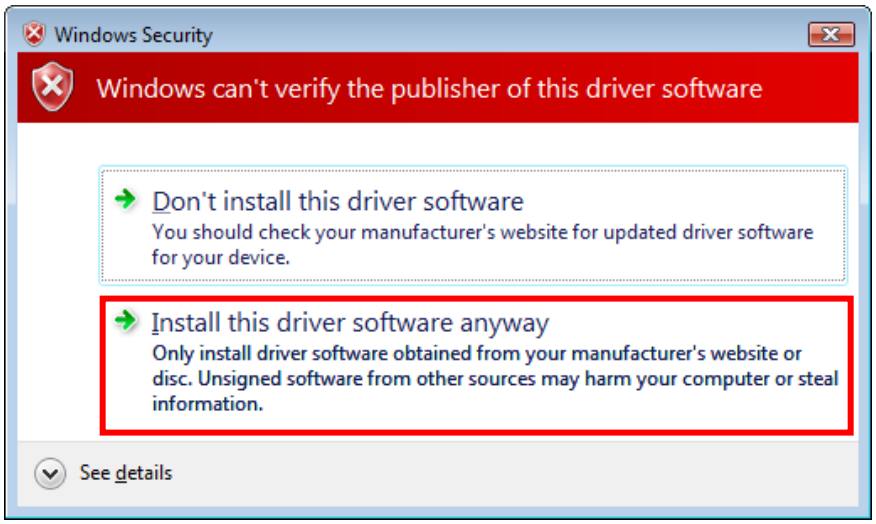


Fig. 1

**Additional Note: After the installation has completed successfully, the program and required driver should now be properly installed when using either Windows XP (32-bit), or Windows 7 (32-bit).*

Once the driver and software have been properly installed on your Windows 7 or Windows XP PC, you are now ready to plug-in your powered USB cable connection from the iCAM TD100 to the computers USB 2.0 (or above) port. A message balloon indicating that new hardware has been properly installed may appear when the USB cable is inserted. Access the installed software by going to your programs screen from the start button on the PC (Fig. 2). If using Windows 7 x64 (64-Bit), follow the steps in section 4.2 for device installation (for Windows 7 64-bit OS only).

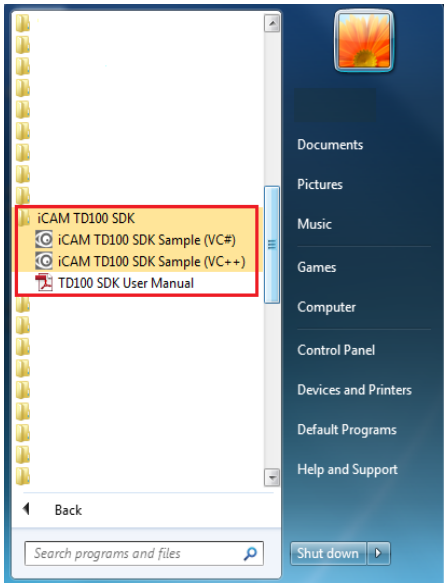


Fig. 2

4.2 Software & USB driver installation of iCAM TD100 for Win 7 x64 Users (64-bit) only.

***Note:** For the latest software & driver version of iCAM TD100 SDK, visit http://www.irisid.com/software.*
*Attention: DO NOT CONNECT THE USB CABLE OF THE iCAM TD100 DEVICE TO THE COMPUTER BEFORE INSTALLING THE SOFTWARE.

Installing the Software For Win 7 x64 (64-bit) users only

The following steps provide information on how to install the software application with a Windows 7 x86 (64-bit) Operating System.

1. Insert iCAM TD100 SDK disc into CD-ROM drive

2. Open the iCAM TD100 SDK _x64 Setup folder from the CD-ROM Drive

3. Open the Setup folder

4. Select >Next at the Welcome to InstallShield Wizard for iCAM TD100 SDK

5. Select Radio button for License agreement and press >Next

6. Select destination folder to install and press >Next

7. Select >Next when the "Start copying files - Review settings before copying files" screen appears. (Wait until software installs. This process may take several minutes.)

8. Select >Finish to complete installation of software successfully.

Note:* Your drivers are digitially signed by Microsoft. However, in the event that a Windows Security warning dialogue box appears as shown in Fig. 3, select **Install this driver software anyway to continue the driver installation process.

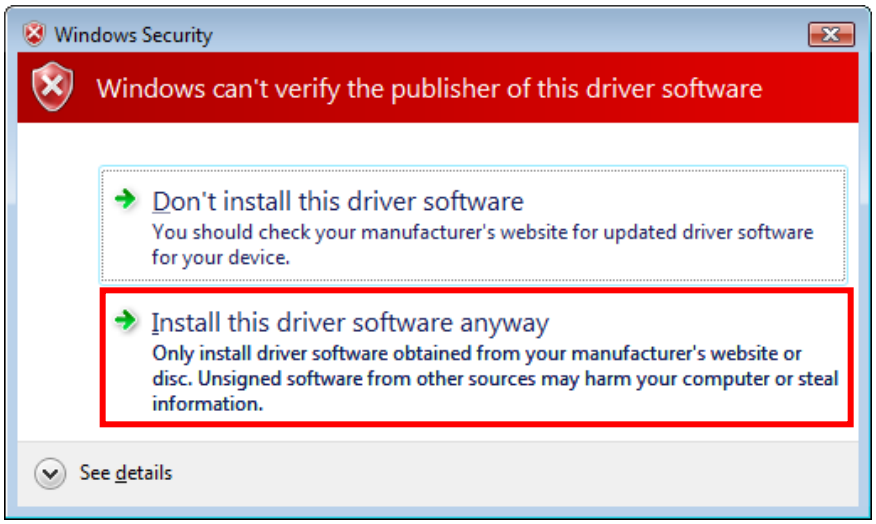


Fig. 3

Additional Note: After the installation has completed successfully, the program and required driver should now be properly installed when using Windows Windows 7 (64-bit). Once the driver and software have been properly installed on your Windows 7, you are now ready to plug-in your powered USB cable connection from the iCAM TD100 to the computer. A message balloon indicating that new hardware has been properly installed may appear when the USB cable is inserted. Access the installed software by going to your programs screen from the start button on the PC (Fig. 4). If using Windows XP or Windows 7 x86 (32-Bit), follow the steps in section 4.1 for device installation (for Windows XP and Windows 7 32-bit OS only).

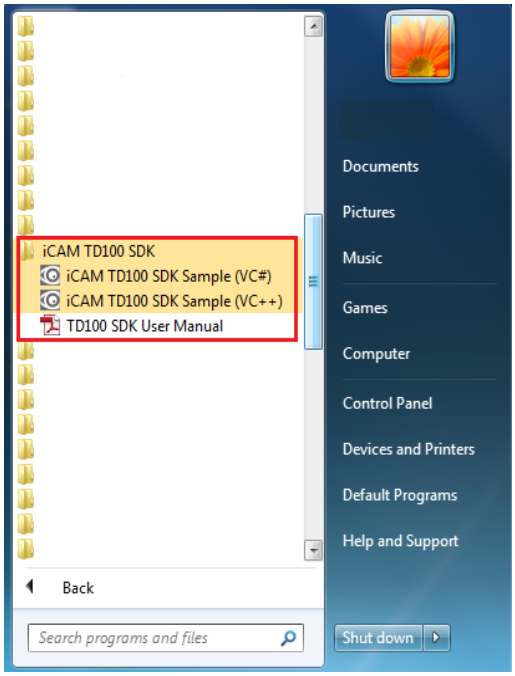


Fig. 4

5. The USB Device Drivers of iCAM TD100

5.1 Verify the success of the USB Device Driver installation of iCAM TD100

The proper USB device driver is automatically installed in the computer when a successful installaltion of the softwae is performed.

After installing iCAM TD100 SDK, test to verify that the software driver has been installed successfully on the PC.

Follow the below steps to perform a verification test of the iCAM TD100 Driver:

1. Connect the iCAM TD100 to an available USB 2.0 port.
2. To verify proper installation of your device driver, view the contents of the Device Manager.
3. Locate the Universal Serial Controllers section *while the iCAM TD100 is connected* to determine if the driver is installed successfully as shown in Fig. 1.

**Note:* It may take several seconds to initialize the driver (as is common) when the iCAM TD100 is connected to the PC through a USB 2.0 port.

*****Note:*** *If the Device Manager shows "Unknown Device" instead of "iCAM TD100 – Iris Capture Device" under "Universal Serial Bus controllers", then refer to Section 8.2 "Resolving the "USB Device Not Recognized" issue" in this manual. To install drivers manually, refer to section 8.3.*

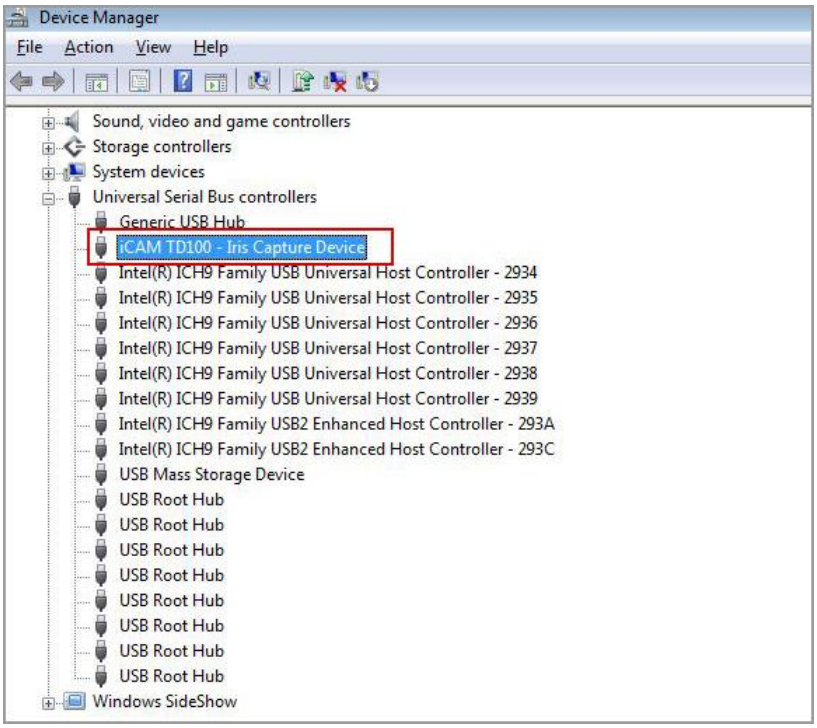


Fig. 1

6. iCAM TD100 SDK Sample Application

6.1 DLL's used to control the iCAM TD100 for image capture for SDK

IAiCAMTD100Control.dll - Used to control the iCAM TD100 for iris image capture. In order to create an iris template or match with iris images, the following .DLL file is needed:

IAiCamIris.dll - Not included in the iCAM TD100 SDK. (Please contact Iris ID for information on receiving this .DLL as needed.)

6.2 Details of the iCAM TD100 SDK Sample Application

The iCAM TD100 SDK sample application is located in **\Program files\Iris ID\iCAM TD100 SDK\Samples** by default. This program consists of two parts; 1) Enrollment images such as iris, face and scene through an iCAM TD100, and 2) Recognition iris image capture. Saved images of either enrollment or recognition images can be viewed from the PC as well. This sample application serves as a good example to better understand how to manage information related to iris images and user's data.

If you enroll or identify a user with the sample application, it will save all data or images to **C:\users\[USER ID]\Document\iCAM TD100 SDK**

**Note:* If you want to clean up enrollment information in the sample program, please remove all of the files in C:\users\[USER ID]\Document\iCAM TD100 SDK.

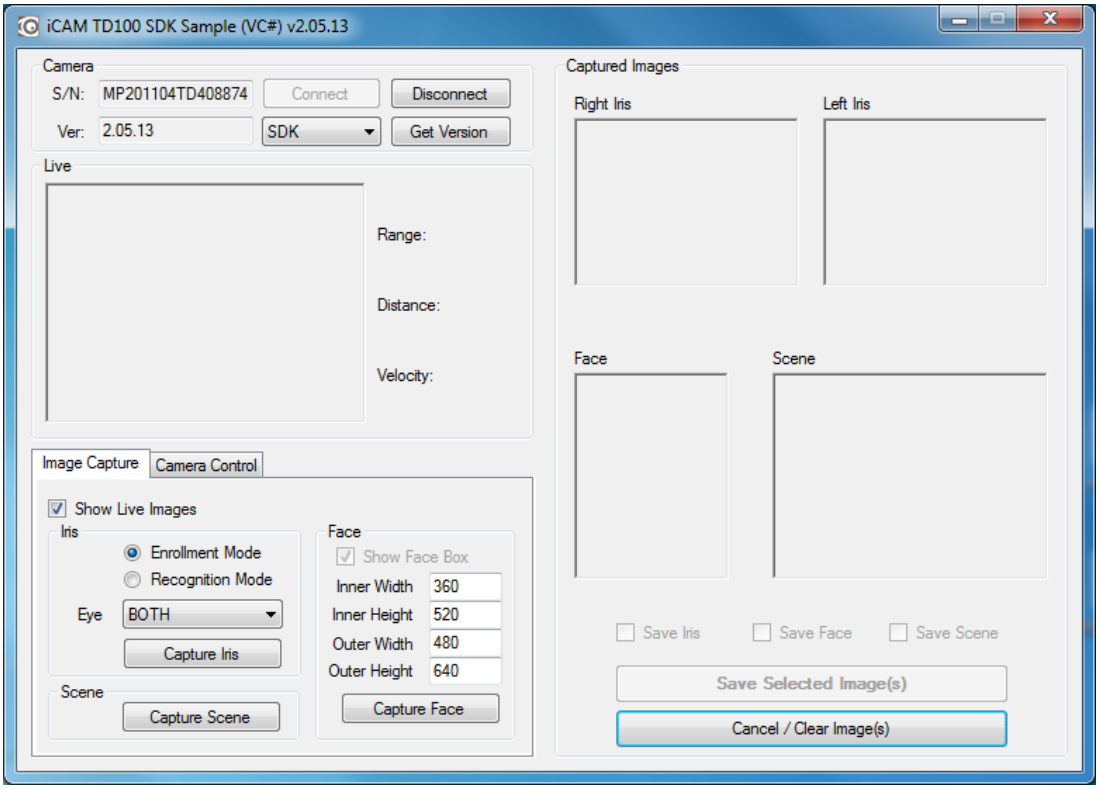


Fig. 1 Sample SDK application preview image

6.3 Opening a Sample Application

1. Connect to iCAM TD100

   Ensure that the iCAM TD100 is connected to the USB port of the computer and that the driver has been installed. The blue LED should be display on the iCAM TD100 (near the shutter button as well as the front face of the iCAM).

2. Press > **Connect** button (as shown in Fig. 1)

Upon successful connection, the serial number field "S/N" will be populated with the iCAM TD100's serial number. Many of the available dialog buttons will become active and the live images will be shown on the LCD Screen (as shown in figure 2).

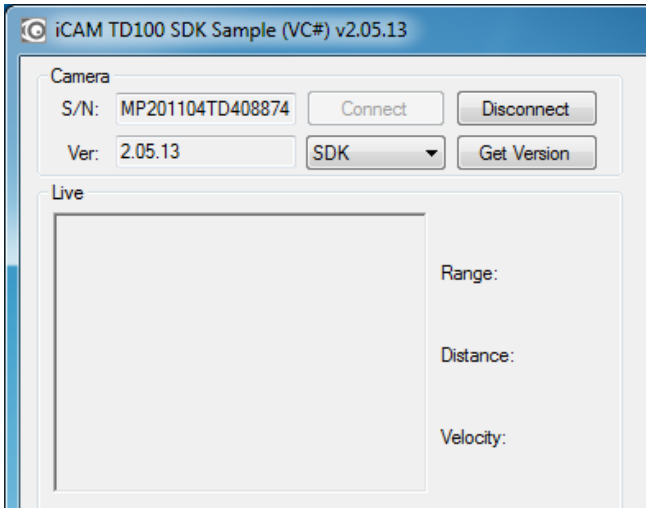


Fig. 2 S/N displayed and live image shown after device connected

The iCAM TD100 is now successfully connected and the sample can properly be used.

6.4 Using the iCAM TD100 SDK Sample Application

Enabling & Disabling Live Images:

The continuous stream of live images displayed on the dialog can be enabled or disabled at any time by checking/un-checking the "Show Live Images" radio box.

Capturing Face/Scene Images:

Face and Scene images can be previewed by pressing the "Capture Face"/ "Capture Scene" button (and then saved as needed for part of enrollment). The continuous stream of live images can be viewed in the Live Image area of the dialog. In the case of face image preview, face guide boxes will be visible by default. Selecting/unselecting the "Show Face Box" check box will either display or hide the face guide boxes. The size of face guide boxes can be modified - Inner Width/Outer Width/Inner Height/Outer Height text boxes. To see the change in size of guide boxes, it is necessary to unselect and then reselect "Show Face Box" check box.

**Note: By default, when the sample application is connected to the TD100, the iCAM will display an image on the device LCD display, but not on the 'Live' field box within the application.*

Enrollment & Recognition

Use the *Eye* Selection dropdown box to specify how the iCAM TD100 with attempt to take an image. This can be specified for *enrollment* and *recognition* modes. The available options for selection are Both Eyes, Left, or Right. Additionally, when performing recognition, the option for "Either eye" is also available for use (using recognition mode selection only).

***Note:** If the user has only one eye, select the eye selection for the eye that needs to be enrolled for automatic image to be taken of that user when using iris capture. For recognition in such instances, the setting of "Either Eye" or "Left"/"Right" eye can be used to properly capture an image automatically from the iCAM TD100.*

The enrollment and recognition functionality can be tested and images can be saved to view from **C:\users\[USER ID]\Document\iCAM TD100 SDK**.

Steps to Enroll User:

1. Select > "Image Capture" tab (if not already selected).

2. Select > "Enrollment Mode" radio button (if not already selected)

3. Select the "Eye-Selection" type that will be used for enrollment. (If only one eye will be enrolled, select the correct eye to enroll (Left/Right).

4. Press > "Capture Iris" button to start iris capture process.

5. On the LCD screen as well as on the Live Area of the dialog, two boxes can be seen. A white face box will appear static, where as a yellow tinted box will be of dynamic size. Its size depends on the distance of users face from the iCAM TD100. When the user is within the appropriate distance from the camera, a green box becomes visible.

6. The user can move closer toward or further away from the camera and position themselves such that the green box is visible, and the position of their eyes are inside that of the green box.

7. At this point in time the iCAM TD100 will automatically capture iris images that will be displayed on the "Captured Images" area of screen.

***Note:** Iris Capture mode also allows for a manual event capture – to perform this process, hold down the physical shutter button on the TD100 camera unit when the user is in proper range. (This procedure can be performed if the user has a single eye, or if the automatic capture is not taking an image.*

8. Select the "Save Iris" check-box on the screen to save these images to the computer, (or select "Save Selected Images" button at bottom of the screen to save all images selected). A message box will be displayed indicating that the images have been saved. (The location of the saves files are: C:\users\[USER ID]\Document\iCAM TD100 SDK\Enrollment).

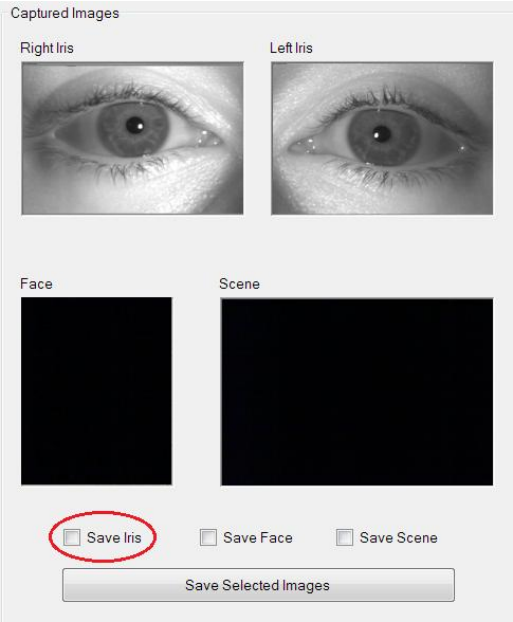


Fig. 3

OR

9. Press the > "Capture Face" button to capture a face image. Either press the same button to "Take Picture" or manually press the shutter button on the TD100 device.

10. Select the "Save Face" check box to save this image to the computer (or select "Save Selected Image(s)" button at bottom of the screen to save all images selected). A message box will be displayed indicating that the images have been saved (The location of the saved files is C:\users\[USER ID]\Document\iCAM TD100 SDK\Face).

11. Press the > "Capture Scene" button to capture a scene image. Either press the same button to "Take scene picture" or manually press the shutter button on the TD100 device.

12. Select the "Save Scene" check box to save this image to the computer (or select "Save Selected Image(s)" button at bottom of the screen to save all images selected). A message box will be displayed indicating that the images have been saved (The location of the saved files is C:\users\[USER ID]\Document\iCAM TD100 SDK\Scene).

13. To complete the save process press the "Save Selected Image(s)" button to save the images (iris Left/Right, Face, and/or Scene), or press the "Cancel / Clear Image(s)" button to cancel the save process operation.

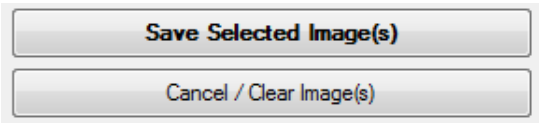


Fig. 4

**__Note__: The expression "Enroll User" refers to the exercise of capturing Iris images in an enrollment mode with our sample application(s). The image that is captured can be saved but will not be stored as a template for purposes of actual identification from the saved enrollment image(s).*

Step to Recognize User

1. Select > "Image Capture" tab (if not already selected).

2. Select > "Recognition Mode" radio button (if not already selected).

3. Select the "Eye-Selection" type that will be used for recognition. (If only one eye will be enrolled, select the correct eye to recognize (Left/Right). Either eye mode can also be used if needed).

4. Press the "Capture Iris" button to start the iris capture process.

5. On the LCD screen as well as on the Live Area of the dialog, two boxes can be seen. A white face box will appear static, where as a yellow tinted box will be of dynamic size. Its size depends on the distance of users face from the iCAM TD100. When the user is within the appropriate distance from the camera, a green box becomes visible.

6. The user can move closer toward or further away from the camera and position themselves such that the green box is visible, and the position of their eyes are inside that of the green box.

7. At this point in time the iCAM TD100 will automatically capture iris images that will be displayed on the "Captured Images" area of screen.

***Note:** Iris Capture mode also allows for a manual event capture – to perform this process, hold down the physical shutter button on the TD100 camera unit when the user is in proper range.*

8. Select the "Save Iris" check box on the right side of the screen to save these images to the recognition images to the computer. A message box will be displayed indicating that the images have been saved (The location of the saved files is C:\users\[USER ID]\Document\iCAM TD100 SDK\Recognition).

**Note: The expression "Recognize User" refers to the exercise of capturing Iris images in a recognition mode with our sample application(s). The image that is captured can be saved but will not be stored as a template for purposes of actual identification from the saved enrollment/recognition image(s).*

Camera Control Options

Additional options for camera control are available in the "Camera Control" section on the sample application located from the tab called Camera Control. These settings allow for the following to be tested:

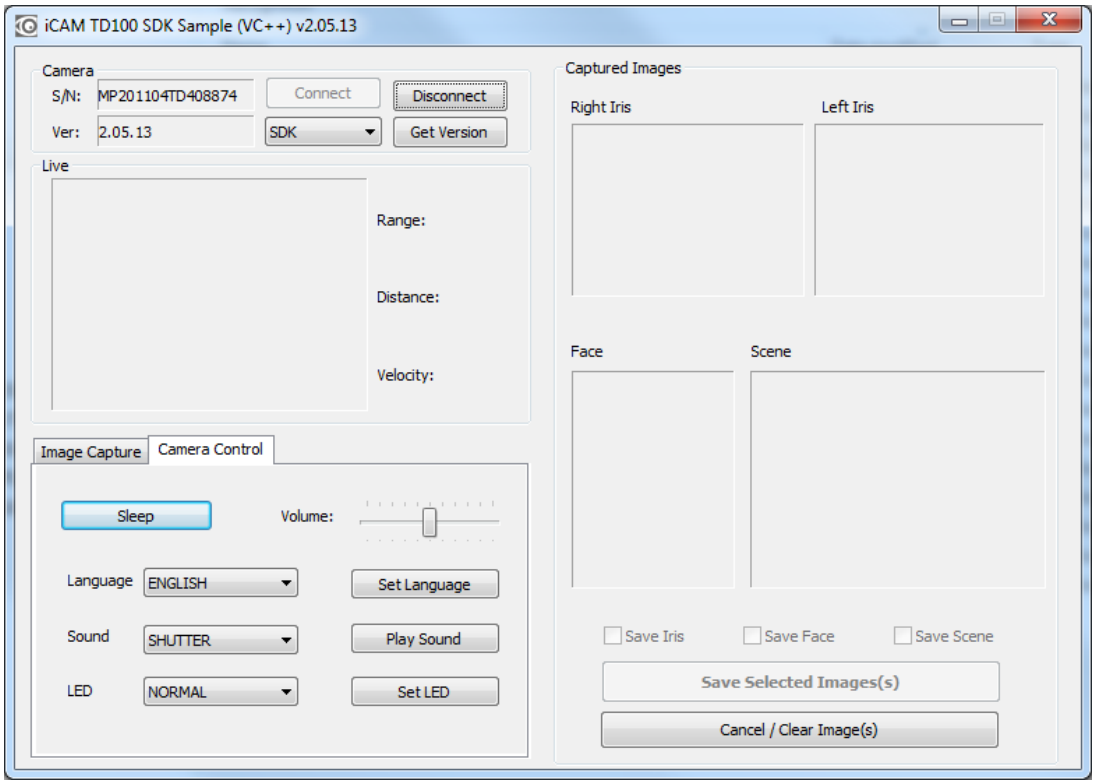


Fig. 5

1. **Sleep mode** – places the camera into a sleep mode state. Press the button again to "wake-up" the camera unit.
2. **Volume modification** – Selectable for lower to higher volume setting – the lowest volume setting will act as mute.
3. **Language type** – 6 languages available directly through this sample application.
    a. English
    b. Arabic
    c. Korean
    d. Spanish
    e. Mandarin
    f. Cantonese
4. **Set Language** – After language type has been chosen, press the "Set language" button to apply the language change.
5. **Sound** – Allows for the sounds to be heard from the camera unit. Selectable sounds can be played and will reflect the language type that has been set in the sample application.
6. **Play Sound** – Executes the command to play a sound. Dependant on the language selected and set and actual sound file that is to be played.
7. **LED** – Selectable dropdown options available to show the LED result action of several functions within the camera.
8. **Set LED** – will display on the TD100 the LED light indication selected from the LED option selection.

7. iCAM TD100 SDK

7.1 Software Development Compatibility

VS2008 SP1 and above can be used to compile the sample applications. If VS2010 is used for the VCPP sample application, please change the $TargetName property for the output file name.

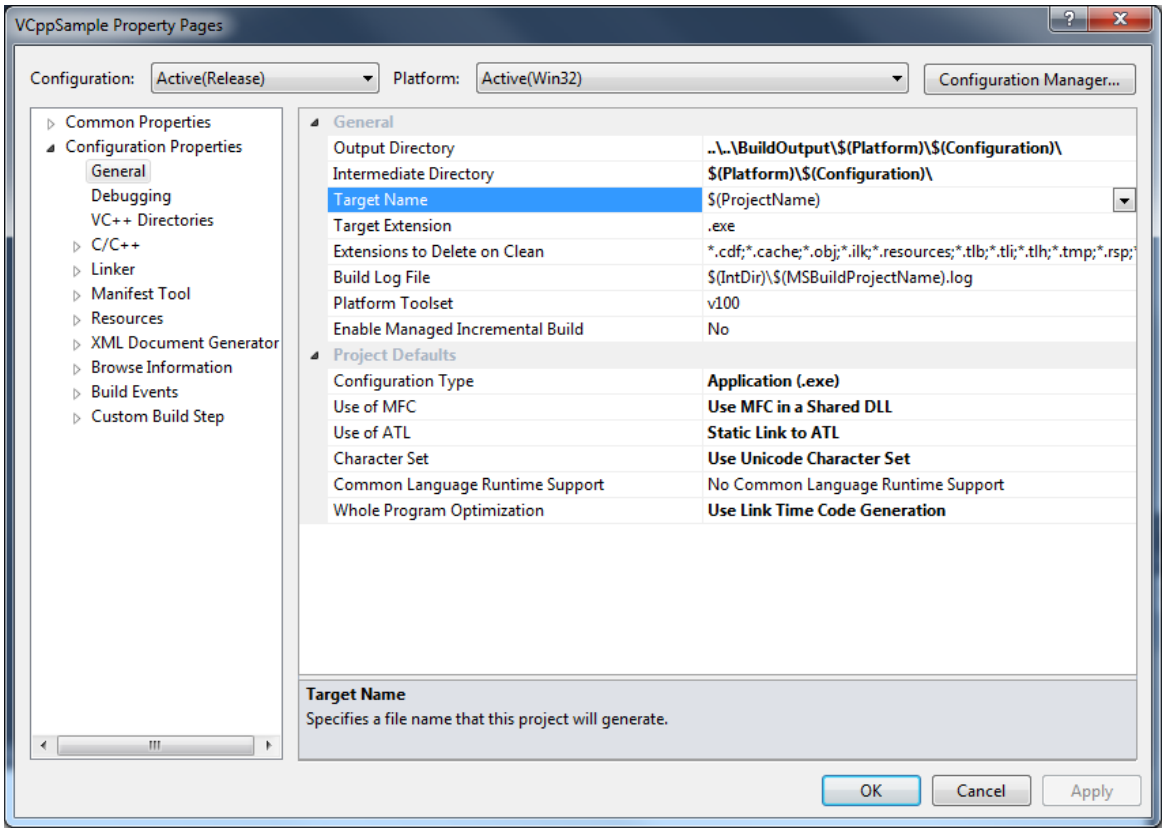


Fig. 1

7.2 Software Architecture

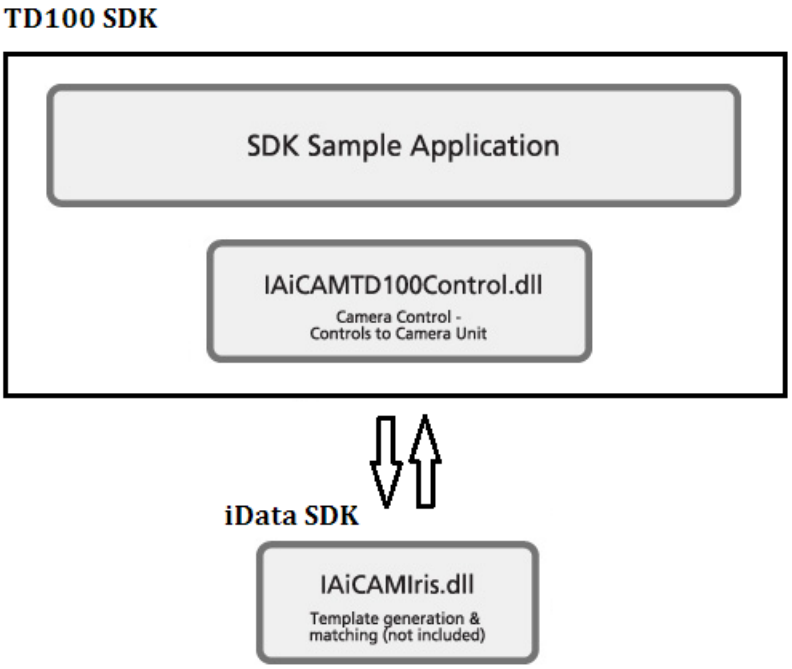


Fig. 2 Software architecture of iris recognition application using iCAM TD100 SDK

The iCAM TD100 needs the IAiCamTD100Control.dll and IAiCamIris.dll for a complete implementation of a host based iris recognition application.

- IAiCamTD100Control.dll will get standards compliant iris images from an iCAM TD100 camera.
- IAiCamIris.dll is used for creating and verifying iris templates and small scale local host matching functions when writing your application. (This is not required for standard usage of the sample applications provided.)
- For large scale 1:N matching applications, the IrisAccelerator™ is used see: http://www.irisid.com/irisaccelerator

**Note:* The entry level iCAM TD100 SDK does not include IAiCamIris.dll (matching and quality assessment functions). However the sample program code in the iCAM TD100 SDK does provide detail on how to call IAiCamIris.dll for iris quality assessment and matching functions. If you need IAiCamIris.dll functionality, please contact Iris ID Systems, Inc. at: sales@irisid.com or call +1-609-819-4747 and select option 2.

7.3 Device Control Library

IAiCamTD100Control.dll

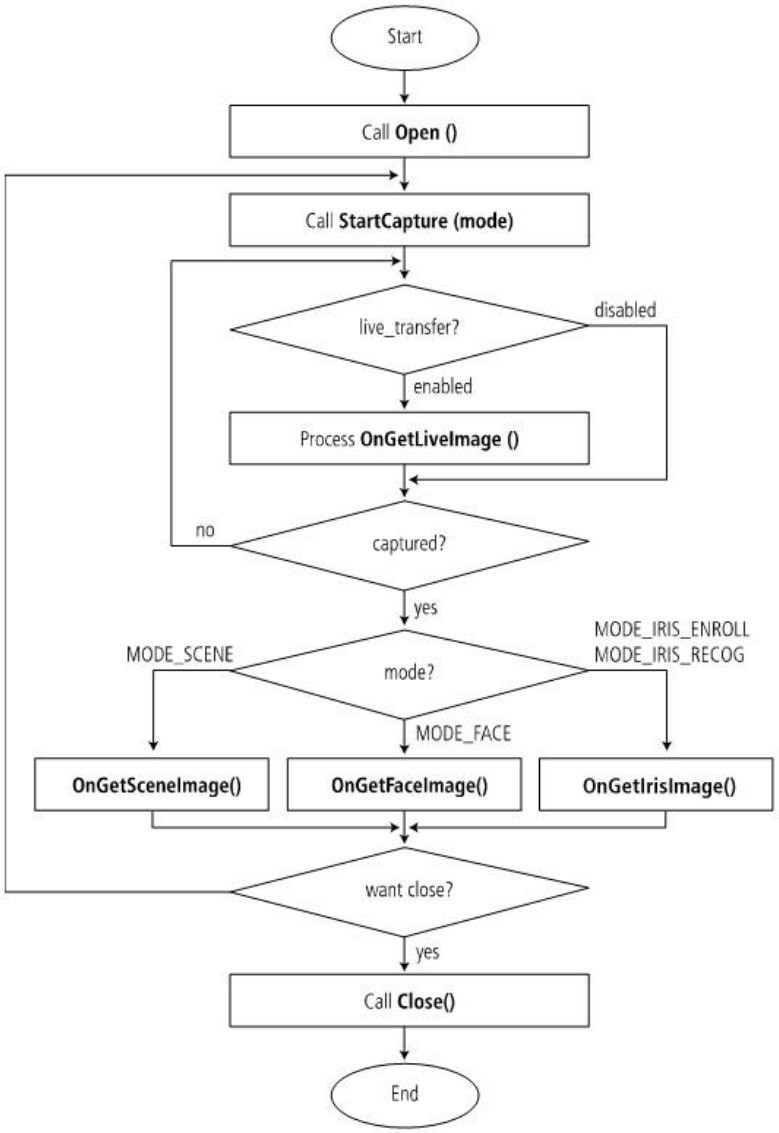
Description: IAiCamTD100Control.dll is an ActiveX .DLL used to control the iCAM TD100 camera through a USB port. Only one camera can be connected to a host PC at a time.

Environment:

- Operating Systems Supported: Windows® XP, Windows® Vista (32-bit), Windows® 7 (32-bit) with .NET Framework 2.0 and Microsoft 2008 redistributable package
- Tool: Microsoft® Visual Studio 2008 Service Pack 1 (SP1)
- Hardware: x86 compatible PC, 2.0 GHz (or higher)
- Sample code supplied by Iris ID : C++, C#

General Procedure for Capturing Images

The below diagram is an overview of the general procedure for capturing images. The diagram steps through the capture process from start to end mapped out in a logical view (for reference purposes only).



Call StartCapture () with mode:
- MODE_SCENE
- MODE_FACE
- MODE_IRIS_ENROLL
- MODE_IRIS_RECOG

To change this option, before calling StartCapture(), call SetLive() with either following parameter:
- LIVE_TRANSFER_DISABLE
- LIVE_TRANSFER_ENABLE

Get live image stream and display them.

By calling PressButton(), clicking the shutter on device, or automatically in Iris capture mode, final image(s) are captured.

According to the mode in StartCapture(), appropriate event will be invoked with the captured final image(s).

Get captured image(s), and store or process them.

By calling close(), the camera will be disconnected. Otherwise application may repeat the operation.

7.4 Application Programming Interfaces

<table>
<tr><th colspan="4">Open ()</th></tr>
<tr><td rowspan="3">Syntax</td><td colspan="3">Open (bstrSerialNumber)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>bstrSerialNumber</td><td>[out] BSTR*</td><td>Serial Number of iCAM TD100 camera</td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command initializes a USB port and connects to an iCAM TD100. When successfully connected to a camera, it returns the Serial Number of the camera unit. Before calling the function, the camera unit should be connected to the PC.</td></tr>
<tr><td>Examples – C#</td><td colspan="3">

```
class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 string serialNumber;

 public event EventHandler OnGetSceneImage;
 public event EventHandler OnGetFaceImage;
 public event EventHandler OnGetLiveImage;
 public event EventHandler OnGetIrisImage;
 public event EventHandler OnGetStatus;
 public event EventHandler OnGetIrisImageInfo;
 public event EventHandler OnGetError;

 public void Open()
 {
  int result;

  if (this.iCam != null)
   throw new Exception("Camera opened already");

  // Initialize on the camera
  this.iCam = new IAiCamTD100ControlClass();

  // Delegate
  this.iCam.OnGetStatus += iCam_OnGetStatus;
  this.iCam.OnGetLiveIrisInfo += iCam_OnGetLiveIrisInfo;
  this.iCam.OnGetLiveImage += iCam_OnGetLiveImage;
  this.iCam.OnGetIrisImage += iCam_OnGetIrisImage;
  this.iCam.OnGetFaceImage += iCam_OnGetFaceImage;
  this.iCam.OnGetSceneImage += iCam_OnGetSceneImage;

  result = this.iCam.Open(out this.serialNumber);
  if (result != (int)CamError.None)
  {
   this.iCam = null;
   throw new Exception("Cannot connect to the camera");
  }

  result = this.iCam.SetLED((int)CamLED.Normal);
  if (result != (int)CamError.None)
  {
   this.iCam = null;
   throw new Exception("Cannot control indicator");
  }
 }
}
```

</td></tr>
</table>

Examples – Visual C++

```
class CCamTD100 : CCamTD100Event
{
  // For IAiCAMTD100Control.dll
  IIAiCamTD100ControlPtr m_pCam;

  // Window Handler to send a message
  HWND m_hTargetWnd;

  // The serial number of the camera
  BSTR m_bstrSerialNumber;
};

long CCamTD100::Open(HWND hWnd)
{
  long lResult;

  // Initialize variables
  lResult = CAM_ERR_NONE;

  try
  {
    // Check input parameters
    if (hWnd == NULL)
      return CAM_ERR_PARAMETER;

    // Check if it is open
    if (m_pCam == NULL)
    {
      // Initialize the COM
      m_pCam = IIAiCamTD100ControlPtr(__uuidof(IAiCamTD100Control));

      // Link the event class
      DispEventAdvise(static_cast<IUnknown*>(m_pCam));
    }
    else
      return CAM_ERR_ALREADY_OPEN;

    m_hTargetWnd = hWnd;

    // Open the camera
    lResult = m_pCam->Open(&m_bstrSerialNumber);
    if (lResult != CAM_ERR_NONE)
    {
      Close();

      return lResult;
    }

    // Set the LED
    lResult = m_pCam->SetLED(LED_NORMAL);
    if (lResult != CAM_ERR_NONE)
    {
      Close();

      return lResult;
    }
  }
  catch (_com_error& error)
  {
    TRACE("%s\n", error.Description());

    return CAM_ERR_UNKNOWN;
  }

  return CAM_ERR_NONE;
}
```

<table>
<tr><th colspan="4">Close ()</th></tr>
<tr><td rowspan="3">Syntax</td><td colspan="3">Close ()</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>None</td><td></td><td></td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command closes the USB connection from an iCAM TD100 camera.</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>
class IAiCamTD100
{
  IAiCamTD100ControlClass iCam;

  public void Close()
  {
    if (this.iCam == null)
      return;

    this.iCam.SetLED((int)CamLED.Normal);

    this.iCam.Close();

    this.iCam = null;
  }
}
</pre></td></tr>
<tr><td>Examples –
Visual C++</td><td colspan="3"><pre>
class CCamTD100 : CCamTD100Event
{
  // For IAiCAMTD100Control.dll
  IIAiCamTD100ControlPtr m_pCam;
};

void CCamTD100::Close(void)
{
  if (m_pCam == NULL)
    return;

  m_pCam->SetLED(LED_NORMAL);

  m_pCam->Close();

  DispEventUnadvise(static_cast<IUnknown*>(m_pCam));

  m_pCam.Release();
}
</pre></td></tr>
</table>

<table>
<tr><th colspan="4">SetLED ()</th></tr>
<tr><td rowspan="3">Syntax</td><td colspan="3">SetLED (lStatus)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>lStatus</td><td>[in] LONG</td><td>Six pre-defined indicator colors and patterns defined.
- LED_NORMAL: Turn on blue
- LED_SUCCESS: Turn on green for a second
- LED_FAILURE: Turn on red for a second
- LED_BUSY: Blinking blue
- LED_ ERROR: Blinking red
- LED_TURN_OFF: Turn off</td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command turns on/off the three-color LED of the camera unit based on “lStatus” argument.</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>class IAiCamTD100
{
  IAiCamTD100ControlClass iCam;

  public void SetLED(CamLED type)
  {
    int result;

    result = this.iCam.SetLED((int)type);
    if (result != (int)CamError.None)
      throw new Exception("Cannot set LED");
  }
}</pre></td></tr>
<tr><td>Examples –
Visual C++</td><td colspan="3"><pre>class CCamTD100 : CCamTD100Event
{
  // For IAiCAMTD100Control.dll
  IIAiCamTD100ControlPtr m_pCam;
};

long CCamTD100::SetLED(CamLED type)
{
  // Check if it is open
  if (m_pCam == NULL)
    return CAM_ERR_CLOSED;

  return m_pCam->SetLED(type);
}</pre></td></tr>
</table>

<table>
<tr><th colspan="4">GetSavedLanguages ()</th></tr>
<tr><td rowspan="4">Syntax</td><td colspan="3">GetSavedLanguages (plNumberOfLanguages, bArrayOfLanguageIDs)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>plNumberOfLanguages</td><td>[out] LONG*</td><td>Number of languages stored in iCAM.</td></tr>
<tr><td>bArrayOfLanguageIDs</td><td>[out] VARIANT*</td><td>Byte array of language IDs that are stored in the iCAM.</td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command gets the list of language IDs saved in iCAM and the number of languages stored.</td></tr>
<tr><td>Note</td><td colspan="3">For the full list of definition of language IDs, refer the Definition section.</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>
class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 public void GetSavedLanguages(out int number, out byte[] languageIDs)
 {
  object IDs;
  int result = iCam.GetSavedLanguages(out number, out IDs);

  if (result != (int)CamError.None)
   throw new Exception("Cannot get saved languages.");

  languageIDs = (byte[])IDs;
 }
}
</pre></td></tr>
<tr><td>Examples – Visual C++</td><td colspan="3"><pre>
class CCamTD100 : CCamTD100Event
{
 // For IAiCAMTD100Control.dll
 IIAiCamTD100ControlPtr m_pCam;
};

long CCamTD100::GetSavedLanguages(long *number, VARIANT* vLangIds)
{
 // Check if it is open
 if (m_pCam == NULL)
  return CAM_ERR_CLOSED;

 return m_pCam->GetSavedLanguages(number, vLangIds);
}
</pre></td></tr>
</table>

<table>
<tr><th colspan="4">SetLanguage ()</th></tr>
<tr><td rowspan="3">Syntax</td><td colspan="3">SetLanguage (lIndex)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>lIndex</td><td>[in] LONG</td><td>Language index that is stored in the iCAM and to be played.</td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command sets the language of voice prompts in the iCAM TD100. Language selection will be saved in non-volatile memory of the camera, therefore remembers the setting even if the camera is disconnected from power or the host PC.</td></tr>
<tr><td>Note</td><td colspan="3">The parameter value, lIndex can be obtained by calling GetSavedLanguages(), of which parameter bArrayOfLanguageIDs contains stored language IDs with specific index. The index of desirable language ID will be the value of the parameter lIndex. For further detail, refer the Sample Applications. The range of lIndex for user-defined language IDs is between 200 and 255.</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>
class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 public int Language
 {
  get
  {
   int result;
   int value;

   result = this.iCam.GetLanguage(out value);
   if (result != (int)CamError.None)
    throw new Exception("Cannot get language");

   return value;
  }
  set { this.iCam.SetLanguage(value); }
 }
}
</pre></td></tr>
<tr><td>Examples – Visual C++</td><td colspan="3"><pre>
class CCamTD100 : CCamTD100Event
{
 // For IAiCAMTD100Control.dll
 IIAiCamTD100ControlPtr m_pCam;
};

long CCamTD100::SetLanguage(int language_index)
{
 // Check if it is open
 if (m_pCam == NULL)
  return CAM_ERR_CLOSED;

 return m_pCam->SetLanguage(language_index);
}
</pre></td></tr>
</table>

<table>
<tr><th colspan="4">GetLanguage ()</th></tr>
<tr><td rowspan="3">Syntax</td><td colspan="3">GetLanguage (plIndex)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>plIndex</td><td>[out] LONG*</td><td>Language index that is stored in the iCAM and has been set to be played.</td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command gets language setting of a camera.</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>
class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 public int Language
 {
  get
  {
   int result;
   int value;

   result = this.iCam.GetLanguage(out value);
   if (result != (int)CamError.None)
    throw new Exception("Cannot get language");

   return value;
  }
  set { this.iCam.SetLanguage(value); }
 }
}
</pre></td></tr>
<tr><td>Examples – Visual C++</td><td colspan="3"><pre>
class CCamTD100 : CCamTD100Event
{
 // For IAiCAMTD100Control.dll
 IIAiCamTD100ControlPtr m_pCam;
};

long CCamTD100::GetLanguage(long *language_index)
{
 long lResult;

 // Initialize variables
 lResult = CAM_ERR_NONE;

 // Check if it is open
 if (m_pCam == NULL)
  return CAM_ERR_CLOSED;

 // Check input parameters
 if (language_index == NULL)
  return CAM_ERR_PARAMETER;

 *language_index = 0;

 lResult = m_pCam->GetLanguage((long*)language_index);

 return lResult;
}
</pre></td></tr>
</table>

<table>
<tr><th colspan="4">PlayMessage ()</th></tr>
<tr><td rowspan="3">Syntax</td><td colspan="3">PlayMessage (lIndex)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>lIndex</td><td>[in] LONG</td><td>Ten pre-defined voice messages defined.
- SOUND_PRESENT:
- SOUND_COME_CLOSER:
- SOUND_MOVE_BACK:
- SOUND_TIMEOUT:
- SOUND_TRY_AGAIN:
- SOUND_CAPTURED:
- SOUND_IDENTIFIED:
- SOUND_NOT_IDENTIFIED:
- SOUND_GRANTED:
- SOUND_DENIED:</td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command plays a message through the built-in speaker of iCAM TD100. Messages will be played according to the language setting of the camera.</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 public void PlaySound(int index)
 {
  int result;

  result = this.iCam.PlayMessage(index);
  if (result != (int)CamError.None)
   throw new Exception("Cannot play sound");
 }
}</pre></td></tr>
<tr><td>Examples – Visual C++</td><td colspan="3"><pre>class CCamTD100 : CCamTD100Event
{
 // For IAiCAMTD100Control.dll
 IIAiCamTD100ControlPtr m_pCam;
};

long CCamTD100::PlaySound(CamSound index)
{
 // Check if it is open
 if (m_pCam == NULL)
  return CAM_ERR_CLOSED;

 return m_pCam->PlayMessage(index);
}</pre></td></tr>
</table>

<table>
<tr><th colspan="4">SetSoundVolume ()</th></tr>
<tr><td rowspan="3">Syntax</td><td colspan="3">SetSoundVolume (lVolume)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>lVolume</td><td>[in] LONG</td><td>The value of sound volume of the iCAM TD100. (0 <= volume <= 10). 0 mutes the sound.</td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command sets camera speaker volume to a specified value. The speaker volume setting will be stored in non-volatile memory in the camera, therefore remembering the setting even if the camera is disconnected from power or the host PC.</td></tr>
<tr><td>Note</td><td colspan="3">The value of sound volume will be 5 after rebooting.</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>
class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 public int Volume
 {
  get
  {
   int result;
   int value;

   result = this.iCam.GetSoundVolume(out value);
   if (result != (int)CamError.None)
    throw new Exception("Cannot get sound volume");

   return value;
  }
  set { this.iCam.SetSoundVolume(value); }
 }
}
</pre></td></tr>
<tr><td>Examples – Visual C++</td><td colspan="3"><pre>
class CCamTD100 : CCamTD100Event
{
 // For IAiCAMTD100Control.dll
 IIAiCamTD100ControlPtr m_pCam;
};

long CCamTD100::SetVolume(long lVolume)
{
 // Check if it is open
 if (m_pCam == NULL)
  return CAM_ERR_CLOSED;

 return m_pCam->SetSoundVolume(lVolume);
}
</pre></td></tr>
</table>

<table>
<tr><th colspan="4">GetSoundVolume ()</th></tr>
<tr><td rowspan="3">Syntax</td><td colspan="3">GetSoundVolume (plVolume)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>plVolume</td><td>[out] LONG*</td><td>The value of sound volume of the iCAM TD100. (0 <= volume <= 10). 0 means the sound is turned off</td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command gets the current setting of a camera speaker volume.</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>
class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 public int Volume
 {
  get
  {
   int result;
   int value;

   result = this.iCam.GetSoundVolume(out value);
   if (result != (int)CamError.None)
    throw new Exception("Cannot get sound volume");

   return value;
  }
  set { this.iCam.SetSoundVolume(value); }
 }
}
</pre></td></tr>
<tr><td>Examples – Visual C++</td><td colspan="3"><pre>
class CCamTD100 : CCamTD100Event
{
 // For IAiCAMTD100Control.dll
 IIAiCamTD100ControlPtr m_pCam;
};

long CCamTD100::GetVolume(void)
{
 long lResult;
 long lVolume;

 // Initialize variables;
 lResult = CAM_ERR_NONE;
 lVolume = 0;

 // Check if it is open
 if (m_pCam == NULL)
  return 0;

 lResult = m_pCam->GetSoundVolume(&lVolume);
 if (lResult != CAM_ERR_NONE)
  return 0;

 return lVolume;
}
</pre></td></tr>
</table>

<table>
<tr><th colspan="4">SetFocusPosition ()</th></tr>
<tr><td rowspan="4">Syntax</td><td colspan="3">SetFocusPosition (lOperatingMode, lFocusValue)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>lOperatingMode</td><td>[in] LONG</td><td>- MODE_FACE: 2
- MODE_SCENE: 1</td></tr>
<tr><td>lFocusValue</td><td>[in] LONG</td><td>The focus position can vary between 0 ~ 15.</td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command sets color focus position at face and scene mode.</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>class IAiCamTD100
{
  IAiCamTD100ControlClass iCam;

  public void SetFocusMode(int mainMode, int focusValue)
  {
    int result = iCam.SetFocusPosition(mainMode, focusValue);
    if (result != (int)CamError.None)
      throw new Exception("Cannot set focus position.");
  }
}</pre></td></tr>
<tr><td>Examples –
Visual C++</td><td colspan="3"><pre>class CCamTD100 : CCamTD100Event
{
  // For IAiCAMTD100Control.dll
  IIAiCamTD100ControlPtr m_pCam;
};

long CCamTD100::SetFocusMode(long lMainMode, long lFocusValue)
{
  // Check if it is open
  if (m_pCam == NULL)
    return CAM_ERR_CLOSED;

  return m_pCam->SetFocusPosition(lMainMode, lFocusValue);
}</pre></td></tr>
</table>

<table>
<tr><th colspan="4">GetFocusPosition ()</th></tr>
<tr><td rowspan="3">Syntax</td><td colspan="3">GetFocusPosition (lFocusValue)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>lFocusValue</td><td>[out] LONG*</td><td>Value for current focus position.
It varies from 0 to 15.</td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command gets color focus position at face and scene mode.</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>
class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 public int GetFocusPosition
 {
  get
  {
   int result;
   int value;

   result = this.iCam.GetFocusPosition(out value);
   if (result != (int)CamError.None)
    throw new Exception("Cannot get focus position");

   return value;
  }
 }
}
</pre></td></tr>
<tr><td>Examples –
Visual C++</td><td colspan="3"><pre>
class CCamTD100 : CCamTD100Event
{
 // For IAiCAMTD100Control.dll
 IIAiCamTD100ControlPtr m_pCam;
};

long CCamTD100::GetFocusPosition(void)
{
 long lResult;
 long lFocusValue;

 // Initialize variables;
 lResult = CAM_ERR_NONE;
 lFocusValue = 0;

 // Check if it is open
 if (m_pCam == NULL)
  return 0;

 lResult = m_pCam->GetFocusPosition(&lFocusValue);
 if (lResult != CAM_ERR_NONE)
  return 0;

 return lFocusValue;
}
</pre></td></tr>
</table>

<table>
<tr><th colspan="4">SetLive ()</th></tr>
<tr><td rowspan="3">Syntax</td><td colspan="3">SetLive (lDisplayMode)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>lDisplayMode</td><td>[in] LONG</td><td>- LIVE_TRANSFER_DISABLE: 0
- LIVE_TRANSFER_ENABLE: 1</td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command is used to enable or disable streaming live images. By default, streaming live images are enabled. Only if streaming live images are enabled, the streaming live images will be fed through OnGetLiveImage().</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 public void SetLive(int type)
 {
  int result;

  result = this.iCam.SetLive(type);
  if (result != (int)CamError.None)
   throw new Exception("Cannot set Live Image");
 }
}</pre></td></tr>
<tr><td>Examples – Visual C++</td><td colspan="3"><pre>class CCamTD100 : CCamTD100Event
{
 // For IAiCAMTD100Control.dll
 IIAiCamTD100ControlPtr m_pCam;
};

long CCamTD100::SetLive(long lDisplay)
{
 // Check if it is open
 if (m_pCam == NULL)
  return CAM_ERR_CLOSED;

 return m_pCam->SetLive(lDisplay);
}</pre></td></tr>
</table>

<table>
<tr><th colspan="4">ShowLCDFaceGuide ()</th></tr>
<tr><td rowspan="3">Syntax</td><td colspan="3">ShowLCDFaceGuide (lDisplay)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>lDisplay</td><td>[in] LONG</td><td>- FACE_BOX_NONE: 0
- FACE_BOX_DISPLAY: 1</td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command is used to enable or disable displaying face guide boxes.</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 public void DisplayLCDFaceBox(int display)
 {
  int result = iCam.ShowLCDFaceGuide(display);
  if (result != (int)CamError.None)
   throw new Exception("Cannot set face box for LCD display.");
 }
}</pre></td></tr>
<tr><td>Examples –
Visual C++</td><td colspan="3"><pre>class CCamTD100 : CCamTD100Event
{
 // For IAiCAMTD100Control.dll
 IIAiCamTD100ControlPtr m_pCam;
};

long CCamTD100::ShowLCDFaceGuide(long lDisplay)
{
 // Check if it is open
 if (m_pCam == NULL)
  return CAM_ERR_CLOSED;

 return m_pCam->ShowLCDFaceGuide(lDisplay);
}</pre></td></tr>
</table>

<table>
<tr><th colspan="4">SetLCDFaceGuideBounds ()</th></tr>
<tr><td rowspan="6">Syntax</td><td colspan="3">SetLCDFaceGuideBounds (lInnerWidth, lInnerHeight, lOuterWidth, lOuterHeight)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>lInnerWidth</td><td>[in] LONG</td><td>width of inner box</td></tr>
<tr><td>lInnerHeight</td><td>[in] LONG</td><td>height of inner box</td></tr>
<tr><td>lOuterWidth</td><td>[in] LONG</td><td>width of outer box</td></tr>
<tr><td>lOuterHeight</td><td>[in] LONG</td><td>height of outer box</td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command is used to set the sizes of face guide boxes.</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>
class IAiCamTD100
{
  IAiCamTD100ControlClass iCam;

  public void SetLCDFaceBox(int inBoxW, int inBoxH, int outBoxW, int outBoxH)
  {
    int result = this.iCam.SetLCDFaceGuideBounds(inBoxW, inBoxH, outBoxW, outBoxH);
    if (result != (int)CamError.None)
      throw new Exception("Cannot set the sizes of face boxes for LCD display.");
  }
}
</pre></td></tr>
<tr><td>Examples –
Visual C++</td><td colspan="3"><pre>
class CCamTD100 : CCamTD100Event
{
  // For IAiCAMTD100Control.dll
  IIAiCamTD100ControlPtr m_pCam;
};

long CCamTD100::SetLCDFaceGuideBounds(
    int iInnerBoxWidth, int iInnerBoxHeight, int iOuterBoxWidth, int iOuterBoxHeight)
{
  // Check if it is open
  if (m_pCam == NULL)
    return CAM_ERR_CLOSED;

    return m_pCam->SetLCDFaceGuideBounds(
        iInnerBoxWidth, iInnerBoxHeight, iOuterBoxWidth, iOuterBoxHeight);
}
</pre></td></tr>
</table>

<table>
<tr><th colspan="4">SetLCDFaceGuidePos ()</th></tr>
<tr><td rowspan="4">Syntax</td><td colspan="3">SetLCDFaceGuidePos (lPositionX, lPositionY)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>lPositionX</td><td>[in] LONG</td><td>X coordinate of guide box center.</td></tr>
<tr><td>lPositionY</td><td>[in] LONG</td><td>Y coordinate of guide box center.</td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command is used to set the position of a guide box on the LCD screen of the iCAM.</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>
class IAiCamTD100
{
  IAiCamTD100ControlClass iCam;

  public void SetLCDFaceGuidePos(int positionX, int positionY)
  {
    int result = iCam.SetLCDFaceGuidePos(positionX, positionY);
    if (result != (int)CamError.None)
      throw new Exception("Cannot set the position of face guide boxes.");
  }
}
</pre></td></tr>
<tr><td>Examples –
Visual C++</td><td colspan="3"><pre>
class CCamTD100 : CCamTD100Event
{
  // For IAiCAMTD100Control.dll
  IIAiCamTD100ControlPtr m_pCam;
};

long CCamTD100::SetLCDFaceGuidePos(int lPositionX, int lPositionY)
{
  // Check if it is open
  if (m_pCam == NULL)
    return CAM_ERR_CLOSED;

  return m_pCam->SetLCDFaceGuidePos(lPositionX, lPositionY);
}
</pre></td></tr>
</table>

<table>
<tr><th colspan="4">StartCapture ()</th></tr>
<tr><td>Syntax</td><td colspan="3">StartCapture (lMode)</td></tr>
<tr><td></td><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td></td><td>lMode</td><td>[in] LONG</td><td>Four pre-defined modes defined.
- MODE_SCENE: To capture an scene image
- MODE_FACE: To capture an face image
- MODE_IRIS_ENROLL_BOTH: To capture right and left iris images for enrollment
- MODE_IRIS_RECOG_BOTH: To capture right and left iris images for recognition
- MODE_IRIS_ENROLL_RIGHT: To capture right iris image for enrollment
- MODE_IRIS_ENROLL_LEFT: To capture left iris image for enrollment
- MODE_IRIS_RECOG_RIGHT: To capture right iris image for recognition
- MODE_IRIS_RECOG_LEFT: To capture left iris image for recognition
- MODE_IRIS_RECOG_EITHER: To capture any iris image for recognition</td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command configures a camera to capture one of the image types described below. When this command is issued, and the option of the streaming live images are enabled, live “preview” image events will return continuously through OnGetLiveImage(). These images can be used to display live images on the user application screen of the host PC. By default, MODE_SCENE is set when Open() is called. According to the mode, final image(s) will be delivered through OnGetSceneImage(), OnGetFaceImage(), or OnGetIrisImage(), respectively.</td></tr>
<tr><td>Detail</td><td colspan="3">• MODE_SCENE (1): In this mode, the LCD of the iCAM TD100 will display live video. At the same time, the host PC user application may receive the identical live image feed through OnGetLiveImage(). The live images are a 400(W)x300(H) color bmp. When proper framing is achieved, an operator can capture a still image by either pressing the shutter button (located on the top of the camera) or by calling PressButton(). A 1600(W)x1200(H) still color bmp image will return through OnGetSceneImage().
• MODE_FACE (2): In this mode, the LCD display will show live images along with a green and white rectangular box. This box approximates a preview of the final face image size/framing for an</td></tr>
</table>

<table>
<tr><td></td><td>operator to properly adjust the camera to frame a quality face picture (as needed). Similar to MODE_SCENE, a user application can get the same live images through OnGetLiveImage(). The face mode live images are a 400(W)x300(H) color bmp. To capture a still image, an operator must press the shutter button or call the PressButton() function. A 1600(W)x1200(H) still color bmp image will return through OnGetFaceImage() after a shutter event.
• MODE_IRIS_ENROLL (3) / MODE_IRIS_RECOG (4) / MODE_IRIS_ENROLL_RIGHT (13) / MODE_IRIS_ENROLL_LEFT (23) / MODE_IRIS_RECOG_RIGHT (14) / MODE_IRIS_RECOG_LEFT (24) / MODE_IRIS_RECOG_EITHER (34): In these modes, the LCD display will show a live video stream with two rectangular boxes. The white box is a reference guide, which gives operators the guidance of where to align a subject's eyes. The red box is a distance indicator- out of range. A user application will get a 320(W)x240(H) color bmp live images through OnGetLiveImage(). When the red box is overlapping with the white box, the color will turn to green, indicating that the subject is in the optimum operating range. iCAM TD100 will automatically capture left and right iris images when the subjects are in the optimal operating range. Captured iris images will return through OnGetIrisImage().</td></tr>
<tr><td>Note</td><td>The captured image by MODE_FACE (2) is a landscape image of 1600(W)x1200(H) color bmp. In order to get a proper portrait face image, application may crop it by 480(W)x640(H) as the center aligned. Typically "resize" operation is not needed to get a proper face image.
MODE_IRIS_ENROLL (3) / MODE_IRIS_RECOG (4) / MODE_IRIS_ENROLL_RIGHT (13) / MODE_IRIS_ENROLL_LEFT (23) / MODE_IRIS_RECOG_RIGHT (14) / MODE_IRIS_RECOG_LEFT (24) / MODE_IRIS_RECOG_EITHER (34) will automatically capture iris images without a button press event. This is a normal way of capturing iris images. In addition, an operator can manually capture iris images by sending a button press event to a camera either by pressing a capture button or calling the PressButton() from the host PC application.</td></tr>
<tr><td>Examples – C#</td><td><pre>class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 public void SetMode(CamMode mode)
 {
  int result;

  if ((mode == CamMode.IrisForEnroll) || (mode == CamMode.IrisForRecog))
   result = this.iCam.SetLED((int)CamLED.Busy);
  else
   result = this.iCam.SetLED((int)CamLED.Normal);

  if (result != (int)CamError.None)
   throw new Exception("Cannot control indicator");

  result = this.iCam.StartCapture((int)mode);
  if (result != (int)CamError.None)
   throw new Exception("Cannot start capture");
 }
}</pre></td></tr>
<tr><td>Examples –</td><td></td></tr>
</table>

Visual C++

```
class CCamTD100 : CCamTD100Event
{
  // For IAiCAMTD100Control.dll
  IIAiCamTD100ControlPtr m_pCam;
};

long CCamTD100::SetMode(CamMode mode)
{
  long lResult;

  // Initialize variables
  lResult = CAM_ERR_NONE;

  // Check if it is open
  if (m_pCam == NULL)
    return CAM_ERR_CLOSED;

  // Set the LED
  if ((mode == MODE_IRIS_ENROLL) || (mode == MODE_IRIS_RECOG))
    lResult = m_pCam->SetLED(LED_BUSY);
  else
    lResult = m_pCam->SetLED(LED_NORMAL);

  if (lResult != CAM_ERR_NONE)
    return lResult;

  // Set the mode
  lResult = m_pCam->StartCapture(mode);

  return lResult;
}
```

<table>
<tr><th colspan="4">PressButton ()</th></tr>
<tr><td rowspan="3">Syntax</td><td colspan="3">PressButton ()</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>None</td><td></td><td></td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command triggers a button event to the iCAM TD100 camera. In response, a user application will get iris images through OnGetIrisImage() in all IRIS modes and face and scene images through OnGetFaceImage() in MODE_FACE and OnGetScene() in MODE_SCENE, respectively. This button event is effective only after StartCapture() is called.</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>
class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 public void CaptureImage()
 {
  int result;

  result = this.iCam.SetLED((int)CamLED.Busy);
  if (result != (int)CamError.None)
   throw new Exception("Cannot control indicator");

  result = this.iCam.PressButton();
  if (result != (int)CamError.None)
   throw new Exception("Cannot capture images");
 }
}
</pre></td></tr>
<tr><td>Examples – Visual C++</td><td colspan="3"><pre>
class CCamTD100 : CCamTD100Event
{
 // For IAiCAMTD100Control.dll
 IIAiCamTD100ControlPtr m_pCam;
};

long CCamTD100::CaptureImage(void)
{
 long lResult;

 // Initialize variables
 lResult = CAM_ERR_NONE;

 // Check if it is open
 if (m_pCam == NULL)
  return CAM_ERR_CLOSED;

 // Set the LED
 lResult = m_pCam->SetLED(LED_BUSY);
 if (lResult != CAM_ERR_NONE)
  return lResult;

 // Capture images
 lResult = m_pCam->PressButton();

 return lResult;
}
</pre></td></tr>
</table>

<table>
<tr><th colspan="4">Sleep ()</th></tr>
<tr><td rowspan="3">Syntax</td><td colspan="3">Sleep ()</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>None</td><td></td><td></td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command sets the iCAM TD100 in sleep mode to conserve power.</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>
class IAiCamTD100
{
  IAiCamTD100ControlClass iCam;

  public void Sleep()
  {
    int result;

    result = this.iCam.Sleep();
    if (result != (int)CamError.None)
      throw new Exception("Cannot sleep the camera");
  }
}
</pre></td></tr>
<tr><td>Examples –
Visual C++</td><td colspan="3"><pre>
class CCamTD100 : CCamTD100Event
{
  // For IAiCAMTD100Control.dll
  IIAiCamTD100ControlPtr m_pCam;
};

long CCamTD100::Sleep(void)
{
  // Check if it is open
  if (m_pCam == NULL)
    return CAM_ERR_CLOSED;

  return m_pCam->Sleep();
}
</pre></td></tr>
</table>

<table>
<tr><th colspan="4">Wakeup ()</th></tr>
<tr><td rowspan="3">Syntax</td><td colspan="3">Wakeup ()</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>None</td><td></td><td></td></tr>
<tr><td>Return Type</td><td>LONG</td><td colspan="2">0 for success. Please refer to IAiCAMTD100Control error code description.</td></tr>
<tr><td>Purpose</td><td colspan="3">This command wakes up a sleeping camera. You can also wake up the camera by pressing the shutter button.</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>
class IAiCamTD100
{
  IAiCamTD100ControlClass iCam;

  public void Wakeup()
  {
    int result;

    result = this.iCam.Wakeup();
    if (result != (int)CamError.None)
      throw new Exception("Cannot wakeup the camera");
  }
}
</pre></td></tr>
<tr><td>Examples – Visual C++</td><td colspan="3"><pre>
class CCamTD100 : CCamTD100Event
{
  // For IAiCAMTD100Control.dll
  IIAiCamTD100ControlPtr m_pCam;
};

long CCamTD100::Wakeup(void)
{
  // Check if it is open
  if (m_pCam == NULL)
    return CAM_ERR_CLOSED;

  return m_pCam->Wakeup();
}
</pre></td></tr>
</table>

7.5 Application Event Functions

<table>
<tr><th colspan="4">OnGetStatus ()</th></tr>
<tr><td>Syntax</td><td colspan="3">OnGetStatus (type)</td></tr>
<tr><td></td><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td></td><td>type</td><td>[in] LONG</td><td>Five pre-defined status defined.
- STAT_USB_DISCONNECTED: When the camera is disconnected
- STAT_IRIS_IMAGE_CAPTURE_FAIL: When it cannot capture iris images after PressButton() or camera button click in Iris mode
- STAT_CAMERA_WAKE_UP: When the sleeping camera is waked up by button press
- STAT_ERROR: When the camera has an error
- STAT_ERROR_TRANSACTION: When the camera is unable to capture image(s) or unable to process internally captured image(s)</td></tr>
<tr><td>Return Type</td><td colspan="3">None</td></tr>
<tr><td>Purpose</td><td colspan="3">This command gets the status of the camera when the status has changed.</td></tr>
<tr><td>Note</td><td colspan="3">When STAT_ERROR_TRANSACTION is delivered, the normal operation may be recovered by calling StartCapture() again,</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>
class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 public event EventHandler OnGetSceneImage;
 public event EventHandler OnGetFaceImage;
 public event EventHandler OnGetLiveImage;
 public event EventHandler OnGetIrisImage;
 public event EventHandler OnGetStatus;
 public event EventHandler OnGetIrisImageInfo;
 public event EventHandler OnGetError;

 public void Open()
 {
  int result;

  if (this.iCam != null)
   throw new Exception("Camera opened already");

  // Initialize on the camera
  this.iCam = new IAiCamTD100ControlClass();

  // Delegate
  this.iCam.OnGetStatus += iCam_OnGetStatus;
  this.iCam.OnGetLiveIrisInfo += iCam_OnGetLiveIrisInfo;
</pre></td></tr>
</table>

```
    this.iCam.OnGetLiveImage += iCam_OnGetLiveImage;
    this.iCam.OnGetIrisImage += iCam_OnGetIrisImage;
    this.iCam.OnGetFaceImage += iCam_OnGetFaceImage;
    this.iCam.OnGetSceneImage += iCam_OnGetSceneImage;

    result = this.iCam.Open(out this.serialNumber);
    if (result != (int)CamError.None)
    {
      this.iCam = null;
      throw new Exception("Cannot connect to the camera");
    }

    result = this.iCam.SetLED((int)CamLED.Normal);
    if (result != (int)CamError.None)
    {
      this.iCam = null;
      throw new Exception("Cannot control indicator");
    }
  }

  void iCam_OnGetStatus(int status, int param)
  {
    CamMessage eventMsg;
    eventMsg = new CamMessage();
    eventMsg.camStatus = (CamStatus)status;
    eventMsg.Distance = param;
    OnGetStatus(this, eventMsg);
  }
}
```

Examples – Visual C++

```
class CCamTD100 : CCamTD100Event
{
  // For IAiCAMTD100Control.dll
  IIAiCamTD100ControlPtr m_pCam;
};

void CCamTD100::OnGetStatus(LONG lStatus, LONG lParam)
{
  CCamMessage* pMessage;

  // Initialize variables;
  pMessage = NULL;

  // Check if it is open
  if (m_pCam == NULL)
    return;

  pMessage = new CCamMessage;

  pMessage->m_camStatus = (CamStatus)lStatus;

  pMessage->m_lDistance = lParam;

  ::PostMessage(m_hTargetWnd, WM_GET_STATUS, (WPARAM)pMessage, 0);
}
```

<table>
<tr><th colspan="4">OnGetLiveIrisInfo ()</th></tr>
<tr><td rowspan="6">Syntax</td><td colspan="3">OnGetLiveIrisInfo (lDistance, lRange, lVelocity, lNone)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>lDistance</td><td>[in] LONG</td><td>Distance from camera (when user is out of range, value is -1)</td></tr>
<tr><td>lRange</td><td>[in] LONG</td><td>Four pre-defined status defined.
- INF_RANGE_OUT: 1
- INF_RANGE_NEAR: 2
- INF_RANGE_FAR: 3
- INF_RANGE_OPERATING: 4</td></tr>
<tr><td>lVelocity</td><td>[in] LONG</td><td>Velocity of movement of user. (All values are positive and when user is out of range OR user is in range but not moving, value is 0)</td></tr>
<tr><td>lNone</td><td>[in] LONG</td><td>Not used as of now</td></tr>
<tr><td>Return Type</td><td>None</td><td colspan="2"></td></tr>
<tr><td>Purpose</td><td colspan="3">This command gets the current information of distance, range, and velocity of detected object.</td></tr>
<tr><td>Note</td><td colspan="3"></td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>
class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 public event EventHandler OnGetSceneImage;
 public event EventHandler OnGetFaceImage;
 public event EventHandler OnGetLiveImage;
 public event EventHandler OnGetIrisImage;
   public event EventHandler OnGetStatus;
 public event EventHandler OnGetIrisImageInfo;
 public event EventHandler OnGetError;

 public void Open()
 {
  int result;

  if (this.iCam != null)
   throw new Exception("Camera opened already");

  // Initialize on the camera
  this.iCam = new IAiCamTD100ControlClass();

  // Delegate
  this.iCam.OnGetStatus += iCam_OnGetStatus;
  this.iCam.OnGetLiveIrisInfo += iCam_OnGetLiveIrisInfo;
  this.iCam.OnGetLiveImage += iCam_OnGetLiveImage;
  this.iCam.OnGetIrisImage += iCam_OnGetIrisImage;
  this.iCam.OnGetFaceImage += iCam_OnGetFaceImage;
  this.iCam.OnGetSceneImage += iCam_OnGetSceneImage;

  result = this.iCam.Open(out this.serialNumber);
  if (result != (int)CamError.None)
  {
</pre></td></tr>
</table>

	this.iCam = null;
	throw new Exception("Cannot connect to the camera");
	}
	
	result = this.iCam.SetLED((int)CamLED.Normal);
	if (result != (int)CamError.None)
	{
	this.iCam = null;
	throw new Exception("Cannot control indicator");
	}
	}
	
	void iCam_OnGetLiveIrisInfo(int lDistance, int lRange, int lVelocity, int lNone)
	{
	CamMessage msg = new CamMessage();
	msg.Distance = lDistance;
	msg.Range = (CamRange) lRange;
	msg.Velocity = lVelocity;
	OnGetIrisImageInfo(this, msg);
	}
	}

Examples –
Visual C++
	class CCamTD100 : CCamTD100Event
	{
	// For IAiCAMTD100Control.dll
	IIAiCamTD100ControlPtr m_pCam;
	};
	
	void CCamTD100::OnGetLiveIrisInfo (
	long lDistance, long lRange, long lVelocity, long lNone)
	{
	CCamMessage* pMessage;
	
	// Initialize variables;
	pMessage = NULL;
	
	// Check if it is open
	if (m_pCam == NULL)
	return;
	
	pMessage->m_lDistance = lDistance;
	pMessage->m_range = (CamRange)lRange;
	pMessage->m_lVelocity = lVelocity;
	
	pMessage->m_camStatus = (CamStatus)lStatus;
	
	::PostMessage(m_hTargetWnd, WM_GET_LIVE_IRIS_INFO, (WPARAM)pMessage, 0);
	}

<table>
<tr><th colspan="4">OnGetIrisImage ()</th></tr>
<tr><td rowspan="4">Syntax</td><td colspan="3">OnGetIrisImage (vRightRawImage, vLeftRawImage)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>vRightRawImage</td><td>[in] VARIANT</td><td>- Right iris image
- 8-bit gray-scale 640x480 RAW format
- 307,200 bytes</td></tr>
<tr><td>vLeftRawImage</td><td>[in] VARIANT</td><td>- Left iris image
- 8-bit gray-scale 640x480 RAW format
- 307,200 bytes</td></tr>
<tr><td>Return Type</td><td>None</td><td colspan="2"></td></tr>
<tr><td>Purpose</td><td colspan="3">After calling StartCapture() function with MODE_IRIS_ENROLL / MODE_IRIS_RECOG / MODE_IRIS_ENROLL_RIGHT / MODE_IRIS_ENROLL_LEFT / MODE_IRIS_RECOG_RIGHT / MODE_IRIS_RECOG_LEFT / MODE_IRIS_RECOG_EITHER, iris images will be automatically captured and returned through this API function when a user is properly positioned within the operating range of camera. Manual image capture is also possible by calling PressButton() or pressing the shutter button. In this case if the captured images do not include iris, PressButton() returns an error value instead of giving iris images through OnGetIrisImage().</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>
class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 public event EventHandler OnGetSceneImage;
 public event EventHandler OnGetFaceImage;
 public event EventHandler OnGetLiveImage;
 public event EventHandler OnGetIrisImage;
 public event EventHandler OnGetStatus;
 public event EventHandler OnGetIrisImageInfo;
 public event EventHandler OnGetError;

 public void Open()
 {
  int result;

  if (this.iCam != null)
   throw new Exception("Camera opened already");

  // Initialize on the camera
  this.iCam = new IAiCamTD100ControlClass();

  // Delegate
  this.iCam.OnGetStatus += iCam_OnGetStatus;
  this.iCam.OnGetLiveIrisInfo += iCam_OnGetLiveIrisInfo;
  this.iCam.OnGetLiveImage += iCam_OnGetLiveImage;
  this.iCam.OnGetIrisImage += iCam_OnGetIrisImage;
  this.iCam.OnGetFaceImage += iCam_OnGetFaceImage;
  this.iCam.OnGetSceneImage += iCam_OnGetSceneImage;

  result = this.iCam.Open(out this.serialNumber);
</pre></td></tr>
</table>

```
    if (result != (int)CamError.None)
    {
      this.iCam = null;
      throw new Exception("Cannot connect to the camera");
    }

    result = this.iCam.SetLED((int)CamLED.Normal);
    if (result != (int)CamError.None)
    {
      this.iCam = null;
      throw new Exception("Cannot control indicator");
    }
  }

  void iCam_OnGetIrisImage(object rightRawImage, object leftRawImage)
  {
    int result;

    CamMessage message;

    message = new CamMessage();

    result = this.iCam.SetLED((int)CamLED.Normal);
    if (result != (int)CamError.None)
    {
      message.strMessage = "Cannot set LED";
      OnGetError(this, message);

      return;
    }

    message.rightIrisImage = rightRawImage;
    message.leftIrisImage = leftRawImage;

    OnGetIrisImage(this, message);
  }
}
```

Examples – Visual C++

```
class CCamTD100 : CCamTD100Event
{
  // For IAiCAMTD100Control.dll
  IIAiCamTD100ControlPtr m_pCam;
};

void CCamTD100::OnGetIrisImage(VARIANT rightRawImage, VARIANT leftRawImage)
{
  CCamMessage* pMessage;

  // Initialize variables;
  pMessage = NULL;

  // check if it is open
  if (m_pCam == NULL)
    return;

  pMessage = new CCamMessage;

  // Set the LED from LED_BUSY to LED_NORMAL
  if (SetLED(CCamTD100::LED_NORMAL) != CAM_ERR_NONE)
  {
    pMessage->m_strMessage = "Cannot set LED";

    ::PostMessage(m_hTargetWnd, WM_GET_ERROR, (WPARAM)pMessage, 0);

    return;
```

	}
	
	// Convert VARIANT type images to byte arrays
	pMessage->m_pRightIrisImage = CVariant::ConvertToArray(&rightRawImage);
	pMessage->m_pLeftIrisImage = CVariant::ConvertToArray(&leftRawImage);
	
	::PostMessage(m_hTargetWnd, WM_GET_IRIS_IMAGE, (WPARAM)pMessage, 0);
	}

<table>
<tr><th colspan="4">OnGetFaceImage ()</th></tr>
<tr><td rowspan="3">Syntax</td><td colspan="3">OnGetFaceImage (vBmpImage)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>vBmpImage</td><td>[in] VARIANT</td><td>- Face image

- Color 1600x1200 BMP image</td></tr>
<tr><td>Return Type</td><td>None</td><td colspan="2"></td></tr>
<tr><td>Purpose</td><td colspan="3">This command gets a face image as PressButton() is called after StartCapture() with MODE_FACE.</td></tr>
<tr><td>Note</td><td colspan="3">Application may crop by 480(W)X640(H).</td></tr>
<tr><td>Examples – C#</td><td colspan="3">

```
class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 public event EventHandler OnGetSceneImage;
 public event EventHandler OnGetFaceImage;
 public event EventHandler OnGetLiveImage;
 public event EventHandler OnGetIrisImage;
 public event EventHandler OnGetStatus;
 public event EventHandler OnGetIrisImageInfo;
 public event EventHandler OnGetError;

 public void Open()
 {
  int result;

  if (this.iCam != null)
   throw new Exception("Camera opened already");

  // Initialize on the camera
  this.iCam = new IAiCamTD100ControlClass();

  // Delegate
  this.iCam.OnGetStatus += iCam_OnGetStatus;
  this.iCam.OnGetLiveIrisInfo += iCam_OnGetLiveIrisInfo;
  this.iCam.OnGetLiveImage += iCam_OnGetLiveImage;
  this.iCam.OnGetIrisImage += iCam_OnGetIrisImage;
  this.iCam.OnGetFaceImage += iCam_OnGetFaceImage;
  this.iCam.OnGetSceneImage += iCam_OnGetSceneImage;

  result = this.iCam.Open(out this.serialNumber);
  if (result != (int)CamError.None)
  {
   this.iCam = null;
   throw new Exception("Cannot connect to the camera");
  }

  result = this.iCam.SetLED((int)CamLED.Normal);
  if (result != (int)CamError.None)
  {
   this.iCam = null;
   throw new Exception("Cannot control indicator");
  }
 }

 void iCam_OnGetFaceImage(object bmpImage)
 {
  int result;
```

</td></tr>
</table>

```
    CamMessage message;

    message = new CamMessage();

    result = this.iCam.SetLED((int)CamLED.Normal);
    if (result != (int)CamError.None)
    {
      message.strMessage = "Cannot control indicator";
      OnGetError(this, message);

      return;
    }

    message.faceImage = bmpImage;

    OnGetFaceImage(this, message);
  }
}
```

Examples – Visual C++

```
class CCamTD100 : CCamTD100Event
{
  // For IAiCAMTD100Control.dll
  IIAiCamTD100ControlPtr m_pCam;
};

void CCamTD100::OnGetFaceImage(VARIANT bmpImage)
{
  CCamMessage* pMessage;

  // Initialize variables;
  pMessage = NULL;

  // Check if it is open
  if (m_pCam == NULL)
    return;

  pMessage = new CCamMessage;

  // Set the LED from LED_BUSY to LED_NORMAL
  if (SetLED(CCamTD100::LED_NORMAL) != CAM_ERR_NONE)
  {
    pMessage->m_strMessage = "Cannot set LED";

    ::PostMessage(m_hTargetWnd, WM_GET_ERROR, (WPARAM)pMessage, 0);

    return;
  }

  // Convert VARIANT type image to byte array
  pMessage->m_pFaceImage = CVariant::ConvertToArray(&bmpImage);

  ::PostMessage(m_hTargetWnd, WM_GET_FACE_IMAGE, (WPARAM)pMessage, 0);
}
```

<table>
<tr><th colspan="4">OnGetSceneImage ()</th></tr>
<tr><td rowspan="3">Syntax</td><td colspan="3">OnGetSceneImage (vBmpImage)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>vBmpImage</td><td>[in] VARIANT</td><td>- Scene image

- Color 1600x1200 BMP image</td></tr>
<tr><td>Return Type</td><td>None</td><td colspan="2"></td></tr>
<tr><td>Purpose</td><td colspan="3">This command gets a scene image as PressButton() is called after StartCapture() with MODE_SCENE.</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>
class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 public event EventHandler OnGetSceneImage;
 public event EventHandler OnGetFaceImage;
 public event EventHandler OnGetLiveImage;
 public event EventHandler OnGetIrisImage;
 public event EventHandler OnGetStatus;
 public event EventHandler OnGetIrisImageInfo;
 public event EventHandler OnGetError;

 public void Open()
 {
  int result;

  if (this.iCam != null)
   throw new Exception("Camera opened already");

  // Initialize on the camera
  this.iCam = new IAiCamTD100ControlClass();

  // Delegate
  this.iCam.OnGetStatus += iCam_OnGetStatus;
  this.iCam.OnGetLiveIrisInfo += iCam_OnGetLiveIrisInfo;
  this.iCam.OnGetLiveImage += iCam_OnGetLiveImage;
  this.iCam.OnGetIrisImage += iCam_OnGetIrisImage;
  this.iCam.OnGetFaceImage += iCam_OnGetFaceImage;
  this.iCam.OnGetSceneImage += iCam_OnGetSceneImage;

  result = this.iCam.Open(out this.serialNumber);
  if (result != (int)CamError.None)
  {
   this.iCam = null;
   throw new Exception("Cannot connect to the camera");
  }

  result = this.iCam.SetLED((int)CamLED.Normal);
  if (result != (int)CamError.None)
  {
   this.iCam = null;
   throw new Exception("Cannot control indicator");
  }
 }

 void iCam_OnGetSceneImage(object bmpImage)
 {
  int result;

  CamMessage message;
</pre></td></tr>
</table>

```
    message = new CamMessage();

    result = this.iCam.SetLED((int)CamLED.Normal);
    if (result != (int)CamError.None)
    {
      message.strMessage = "Cannot control indicator";
      OnGetError(this, message);

      return;
    }

    message.sceneImage = bmpImage;

    OnGetSceneImage(this, message);
  }
}
```

Examples – Visual C++

```
class CCamTD100 : CCamTD100Event
{
  // For IAiCAMTD100Control.dll
  IIAiCamTD100ControlPtr m_pCam;
};

void CCamTD100::OnGetSceneImage(VARIANT bmpImage)
{
  CCamMessage* pMessage;

  // Initialize variables;
  pMessage = NULL;

  if (m_pCam == NULL)
    return;

  // Check if it is open
  pMessage = new CCamMessage;

  // Set the LED from LED_BUSY to LED_NORMAL
  if (SetLED(CCamTD100::LED_NORMAL) != CAM_ERR_NONE)
  {
    pMessage->m_strMessage = "Cannot set LED";

    ::PostMessage(m_hTargetWnd, WM_GET_ERROR, (WPARAM)pMessage, 0);

    return;
  }

  pMessage->m_pSceneImage = CVariant::ConvertToArray(&bmpImage);

  ::PostMessage(m_hTargetWnd, WM_GET_SCENE_IMAGE, (WPARAM)pMessage, 0);
}
```

<table>
<tr><th colspan="4">OnGetLiveImage ()</th></tr>
<tr><td rowspan="5">Syntax</td><td colspan="3">OnGetLiveImage (vBmpImage, lRange, lDistance)</td></tr>
<tr><td>Argument</td><td>Type</td><td>Description</td></tr>
<tr><td>vBmpImage</td><td>[in] VARIANT</td><td>- Color BMP image
- 320x240 resolution image in all IRIS modes
- 400x300 resolution image in MODE_FACE and MODE_SCENE
- Guide boxes are imprinted in MODE_FACE and all IRIS modes</td></tr>
<tr><td>lRange</td><td>[in] LONG</td><td>No longer supported</td></tr>
<tr><td>lDistance</td><td>[in] LONG</td><td>- Distance from a camera to a user
- Unit is a millimeter
- 0 means it cannot measure the distance</td></tr>
<tr><td>Return Type</td><td>None</td><td colspan="2"></td></tr>
<tr><td>Purpose</td><td colspan="3">This command gets live images. It includes distance information in only all IRIS modes of StartCapture().</td></tr>
<tr><td>Note</td><td colspan="3">Parameter lRange is no longer supported; always returns 0.
To get range information, use OnGetLiveIrisInfo ().</td></tr>
<tr><td>Examples – C#</td><td colspan="3"><pre>
class IAiCamTD100
{
 IAiCamTD100ControlClass iCam;

 public event EventHandler OnGetSceneImage;
 public event EventHandler OnGetFaceImage;
 public event EventHandler OnGetLiveImage;
 public event EventHandler OnGetIrisImage;
 public event EventHandler OnGetStatus;
 public event EventHandler OnGetIrisImageInfo;
 public event EventHandler OnGetError;

 public void Open()
 {
  int result;

  if (this.iCam != null)
   throw new Exception("Camera opened already");

  // Initialize on the camera
  this.iCam = new IAiCamTD100ControlClass();

  // Delegate
  this.iCam.OnGetStatus += iCam_OnGetStatus;
  this.iCam.OnGetLiveIrisInfo += iCam_OnGetLiveIrisInfo;
  this.iCam.OnGetLiveImage += iCam_OnGetLiveImage;
  this.iCam.OnGetIrisImage += iCam_OnGetIrisImage;
</pre></td></tr>
</table>

	this.iCam.OnGetFaceImage += iCam_OnGetFaceImage;
	this.iCam.OnGetSceneImage += iCam_OnGetSceneImage;
	
	result = this.iCam.Open(out this.serialNumber);
	if (result != (int)CamError.None)
	{
	this.iCam = null;
	throw new Exception("Cannot connect to the camera");
	}
	
	result = this.iCam.SetLED((int)CamLED.Normal);
	if (result != (int)CamError.None)
	{
	this.iCam = null;
	throw new Exception("Cannot control indicator");
	}
	}
	
	void iCam_OnGetLiveImage(object bmpImage, int range, int distance)
	{
	CamMessage imageMsg;
	
	imageMsg = new CamMessage();
	
	imageMsg.liveImage = bmpImage;
	
	//imageMsg.range = (CamRange)range; // No longer supported.
	imageMsg.distance = distance;
	
	OnGetLiveImage(this, imageMsg);
	}
	}

Examples –
Visual C++
	class CCamTD100 : CCamTD100Event
	{
	// For IAiCAMTD100Control.dll
	IIAiCamTD100ControlPtr m_pCam;
	};
	
	void CCamTD100::OnGetLiveImage(VARIANT bmpImage, LONG lRange, LONG lDistance)
	{
	CCamMessage* pMessage;
	
	// Initialize variables;
	pMessage = NULL;
	
	// Check if it is open
	if (m_pCam == NULL)
	return;
	
	pMessage = new CCamMessage;
	
	pMessage->m_pLiveImage = CVariant::ConvertToArray(&bmpImage);
	pMessage->m_range = (CamRange)lRange;
	pMessage->m_lDistance = lDistance;
	
	::PostMessage(m_hTargetWnd, WM_GET_LIVE_IMAGE, (WPARAM)pMessage, 0);
	}

7.6 Definitions

IAiCamTD100Control Error Code

#define CAM_ERR_NONE	0
#define CAM_ERR_PARAMETER	1001
#define CAM_ERR_OPEN	1002
#define CAM_ERR_ALREADY_OPEN	1003
#define CAM_ERR_CLOSED	1004
#define CAM_ERR_IMAGE_PROCESS	1005
#define CAM_ERR_SLEEPING	1006
#define CAM_ERR_UNKNOWN	-1

LED Type

#define LED_NORMAL	1
#define LED_SUCCESS	2
#define LED_FAILURE	3
#define LED_BUSY	4
#define LED_ERROR	5
#define LED_TURN_OFF	6

Language ID

#define LNG_NONE	0
#define LNG_KOREAN	1
#define LNG_ENGLISH	2
#define LNG_ARABIC	3
#define LNG_CHINESE_MANDARIN	4
#define LNG_CHINESE_CANTONESE	5
#define LNG_FRENCH	10
#define LNG_GERMAN	11
#define LNG_ITALIAN	16
#define LNG_JAPANESS	17
#define LNG_SPANISH	24
#define LNG_USER_1	200
#define LNG_USER_2	201

Voice Messages

#define SOUND_PRESENT	1
#define SOUND_COME_CLOSER	2
#define SOUND_MOVE_BACK	3
#define SOUND_TIMEOUT	4

#define SOUND_TRY_AGAIN	5
#define SOUND_CAPTURED	6
#define SOUND_IDENTIFIED	7
#define SOUND_NOT_IDENTIFIED	8
#define SOUND_GRANTED	9
#define SOUND_DENIED	10

Streaming Live Images

#define LIVE_TRANSFER_DISABLE	0
#define LIVE_TRANSFER_ENABLE	1

Face Guide Box Display

#define FACE_BOX_NONE	0
#define FACE_BOX_DISPLAY	1

Face Guide Box Dimensions

#define DEFAULT_FACE_IN_BOX_WIDTH	360
#define DEFAULT_FACE_IN_BOX_HEIGHT	520
#define DEFAULT_FACE_OUT_BOX_WIDTH	480
#define DEFAULT_FACE_OUT_BOX_HEIGHT	640

Camera Mode

#define MODE_SCENE	1
#define MODE_FACE	2
#define MODE_IRIS_ENROLL	3
#define MODE_IRIS_ENROLL_RIGHT	13
#define MODE_IRIS_ENROLL_LEFT	23
#define MODE_IRIS_RECOG	4
#define MODE_IRIS_RECOG_RIGHT	14
#define MODE_IRIS_RECOG_LEFT	24
#define MODE_IRIS_RECOG_EITHER	34

Camera Status

#define STAT_USB_DISCONNECTED	1
#define STAT_IRIS_IMAGE_CAPTURE_FAIL	2
#define STAT_CAMERA_WAKE_UP	3
#define STAT_ERROR	4
#define STAT_ERROR_TRANSACTION	5

Iris Images Dimension

#define IRIS_IMAGE_WIDTH	640
#define IRIS_IMAGE_HEIGHT	480
#define IRIS_IMAGE_SIZE	640 * 480

Range of Distance

#define INF_RANGE_OUT	1
#define INF_RANGE_NEAR	2
#define INF_RANGE_FAR	3
#define INF_RANGE_OPERATING	4

8. Troubleshooting

In the event that an error occurs with your system, installation or its functionality, please review the Frequently Asked Questions section and troubleshooting notes for assistance with finding answers to common issues.

If your issue persists, locate the Technical support section in this document for assistance in resolving questions and concerns.

8.1 Uninstalling the iCAM TD100 SDK Software

Removing the software:

To uninstall the application from the computer for any reason, go to the add/remove programs section of windows located from the Control Panel.

After the software has been uninstalled from the PC, the device drivers, icons, and files should be removed from your computer.

8.2 Resolving the "USB Device Not Recognized" issue

- *Note:* If you are attempting to use the TD100 SDK software version 2.x or above with an older iCAM TD100 device – such compatibility is not provided. Please contact Iris ID to inquire about iCAM TD100 firmware upgrade options. Additionally, please refer to the Iris ID website for any firmware version updates/changes and upgrade utilities as needed.

Fig. 1

If a message balloon "USB Device Not Recognized" appears when the USB cable of TD100 device is plugged in to a port of PC, as shown in Fig. 1, follow the below procedure:

1. Unplug the USB cable from the port of the PC.
2. Close any opened TD100 Application in the PC.
3. Unplug the power cable so that the TD100 device shuts down.
4. Wait at least two (2) seconds, in order to give a chance to refresh the internal memory of TD100 device.
5. Plug-in the power cable.
6. Plug-in the USB cable to a USB port of PC.

7. If the driver installation is still required, follow the instruction written in section 8.3 to manually install USB drivers and "to verify hardware/software/OS compatibility".

Once the procedures have been performed, the PC should detect the TD100 device correctly as "iCAM TD100 – Iris Capture Device"; and the subsequent operations should come back to normal state.

8.3 Manually installing TD100 device drivers

If your drivers have not been automatically installed, and section 8.2 did not resolve the USB device issue; the following procedures will assist you with manually installing the iCAM TD100 device.

Step 1: Determine what version of iCAM TD100 driver you need to install.

A. Once the iCAM TD100 is connected to the USB, the window shown in Figure A will appear. Select the radio button > "No, not this time" and press "Cancel".

Fig. 2

B. Launch Device Manager by opening Control Panel > System > Device Manager.

C. Expand "Other devices" and locate "USB Device" as shown in Figure 2.

**Note: This device may also appear in the device manager as either "Unknown Device" or iCAM TD100 Device"*

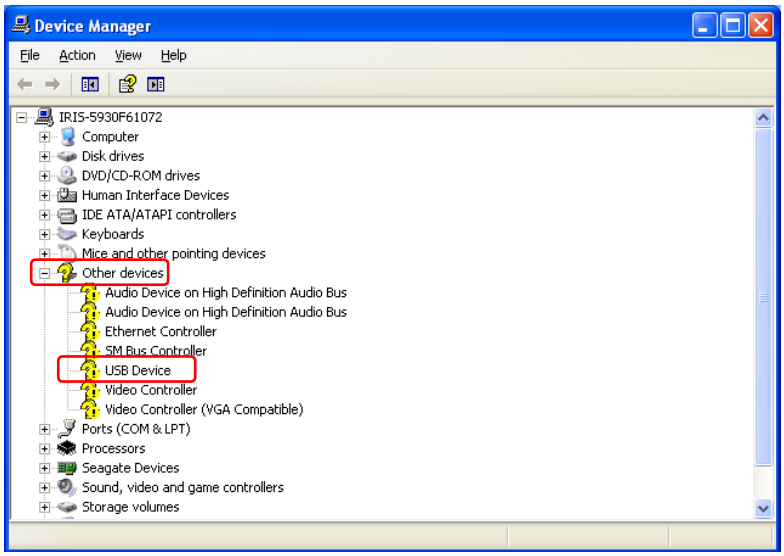


Fig. 3

D. Right click on USB Device (shown in Figure 3) and select "Properties".

E. From the USB Device Properties menu, select the "Details" tab.

F. Under the drop down list of USB Device, select Hardware Ids as shown in Figure 4 and 5.

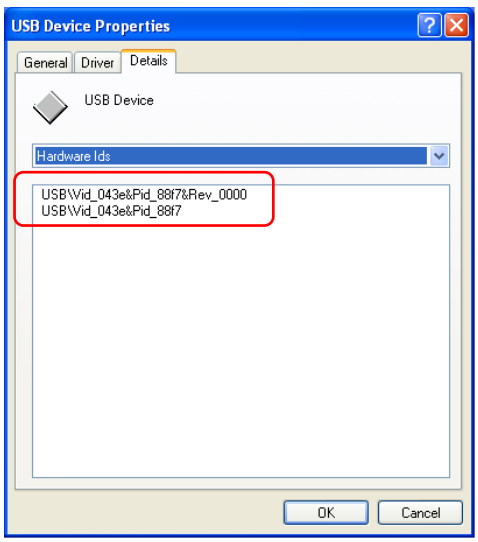


Fig. 4

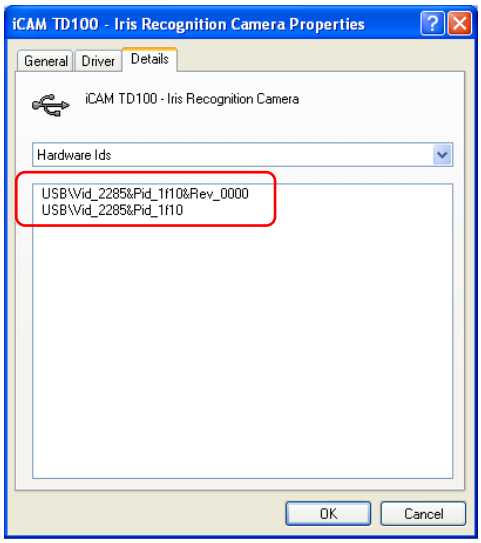


Fig. 5

G. The 2nd line which displays the USB\VID_####&Pid_####

H. Locate the correct drivers that pertain to the iCAM TD100 Hardware Ids below:

VID/PID	32-bit (x86) build	64-bit (x64) build
USB\Vid_043e&Pid_88f7 & older	\\Program Files\Iris ID\iCAM TD100 SDK\Drivers\TD100Down\VID_043E_PID_88F7 and \\Program Files\Iris ID\iCAM TD100 SDK\Drivers\TD100\VID_043E_PID_88F6	NA
USB\Vid_2285&Pid_1f10 & newer	\\Program Files\Iris ID\iCAM TD100 SDK\Drivers\TD100Down\VID_2285_PID_1F00 and \\Program Files\Iris ID\iCAM TD100 SDK\Drivers\TD100\VID_2285_PID_1F10	\\ProgramFiles 64\Iris ID\iCAM TD100 SDK\ Drivers\TD100\Win7

Compatibility version Chart

Based on the Compatibility version chart, determine what driver version to install.

****Note:*** *If you are attempting to use the iCAM TD100 device Vid_043e&Pid_88f7 & older on 64-bit OS – such compatibility is not provided. Please contact Iris ID to inquire about TD100 SDK software version options. Additionally, please refer to the Iris ID website for any firmware version updates/changes and upgrade utilities as needed.*

Step 2: Installing the iCAM TD100 Device Driver.

The following steps provide information for installing the iCAM TD100 driver manually. The driver for the iCAM TD100 may *NOT* automatically be installed and must be installed separately after software installation.

Before proceeding with driver installation, first determine which driver is required for compatible installatoin (see Troubleshooting 8.3 – Step 1, if you have not followed this procedure already).

NOTE: *The below shows an example of manual installation of TD100 Driver installation with use of the Windows XP (32-bit) Operating System and an iCAM TD100 Device with USB\Vid_043e&Pid_88f7 firmware currently installed. (If using another operating system such as Windows 7 (32-bit) - make sure to follow the below instructions but substitute the appropriate driver as needed.)*

Installation of device driver Steps (For Windows XP example with USB\Vid_2285&Pid_1f10 & newer):

1. Connect the iCAM TD100 to the USB 2.0 port of the PC running windows XP (or Windows 7-32 Bit). (If already connected, the "Found New hardware wizard" will automatically appear.)

****Note:*** A message balloon "Found New Hardware" may appear. (This balloon message may close automatically as the "Found New Hardware Wizard" will appear).

2. Select the radio button > "**Yes, now and every time I connect a device**" from the Found New Hardware Wizard Welcome screen and then press > "**Next**" as shown in Figure 6.

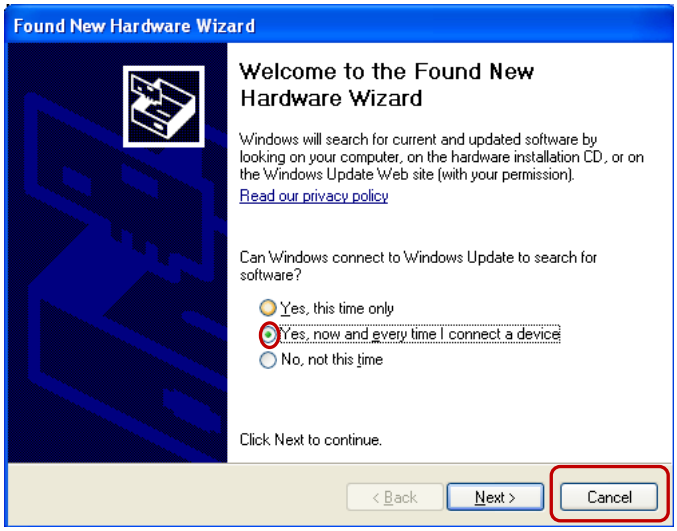


Fig. 6

3. Select the radio button > "install from a specific location (advanced) and then press > Next as shown in Figure 7.

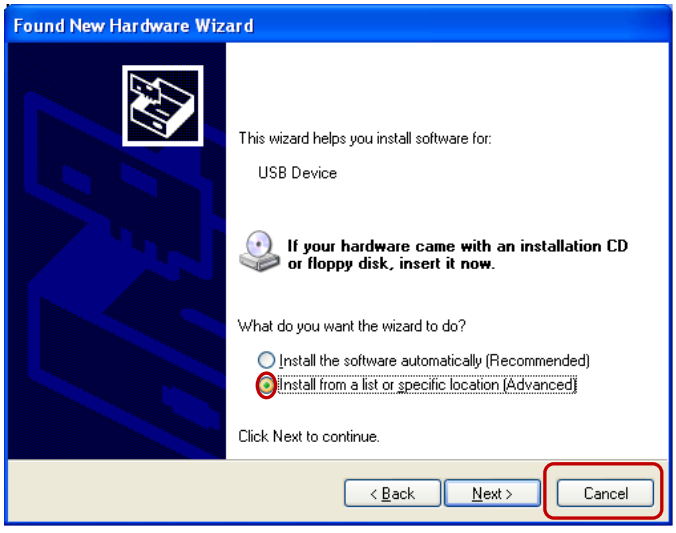


Fig. 7

4. Select > "**Don't search. I will choose the driver to install.**" and press > "**Next**" as shown in Figure 8.

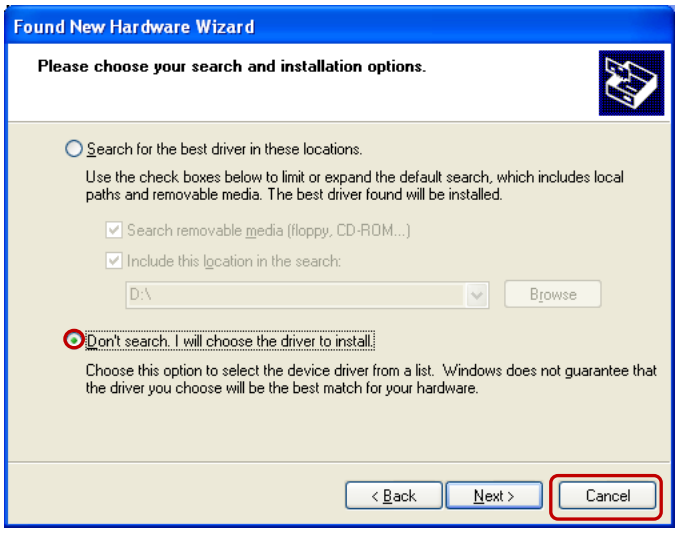


Fig. 8

***Note:** If a "Hardware Type" screen comes up as shown in Figure 9, select > "**Universal Serial Bus controllers**". Then click > "**Next**".*

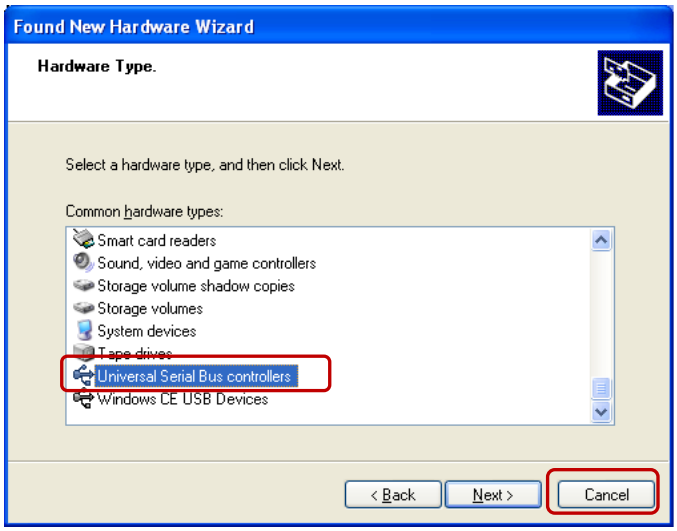


Fig. 9

5. Press > "**Have Disk…**" as shown in Figure 10.

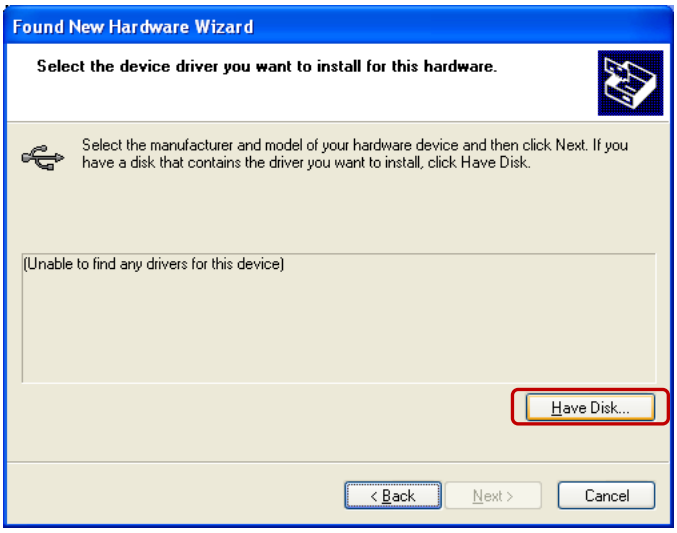


Fig. 10

6. Press > "**Browse**…" as shown in Figure 11

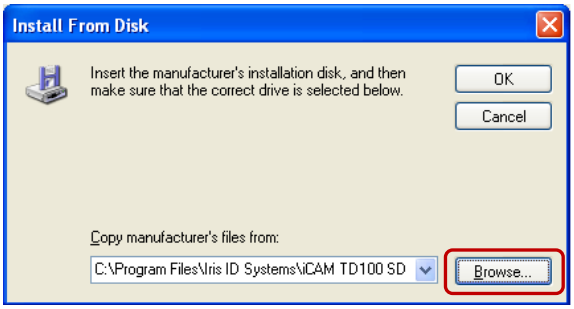


Fig. 11

7. Determine the iCAM **TD100Down** driver you need to install based on the chart (found in Section 8.3 Step 1). For this example we will be updating the iCAM TD100 Vid _043e&Pid_88f7 Hardware: "/Program Files/Iris ID Systems/iCAM TD100 SDK/Drivers/TD100Down/VID_043E_PID_88F7", select "TD100Down.inf" and press > "**Open**" as shown in Figure 12.

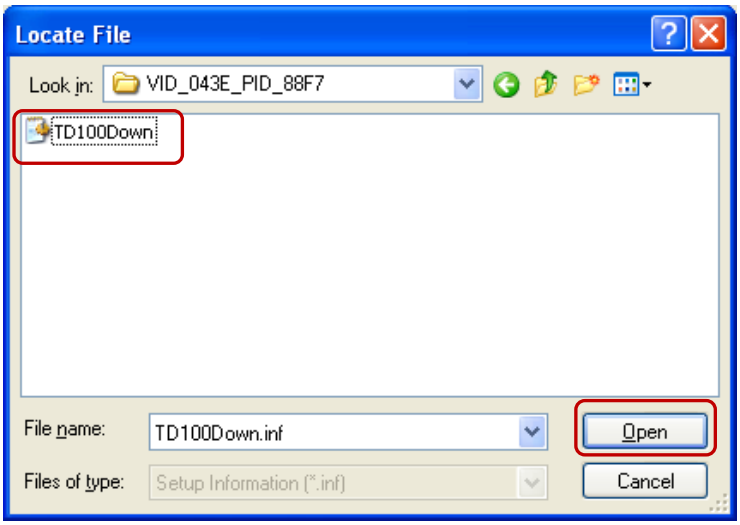


Fig. 12

***Note:** If a wrong INF file is selected, an error message box may appear as shown in Figure 13.*

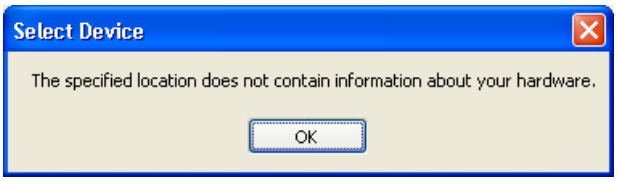


Fig. 13

8. Press > "**OK**" button as shown in Figure 14, to proceed to next screen.

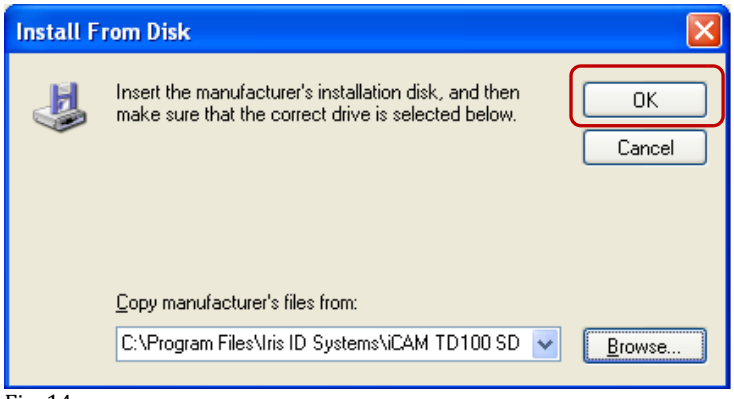


Fig. 14

9. Press > "**Next**" button as shown in Figure 15, to proceed to next screen.

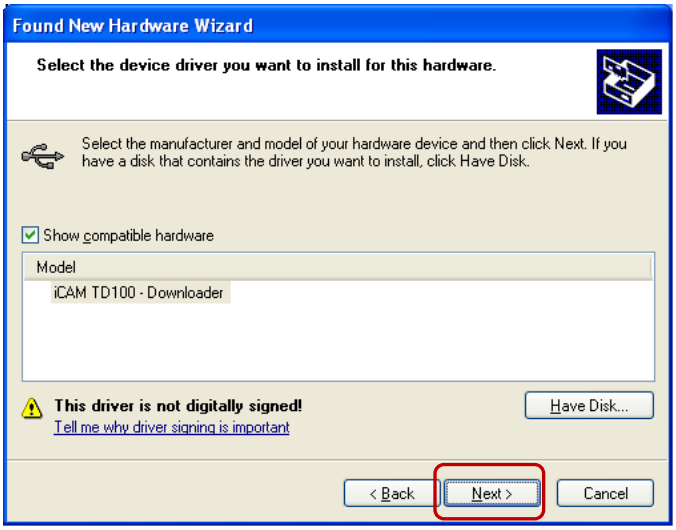


Fig. 15

10. Select > "**Continue Anyway**" if an attention box indicating that the "iCAM TD100 – Iris Capture Device has not passed windows Logo testing to verify…"to continue installing driver files as shown in Figure 16.

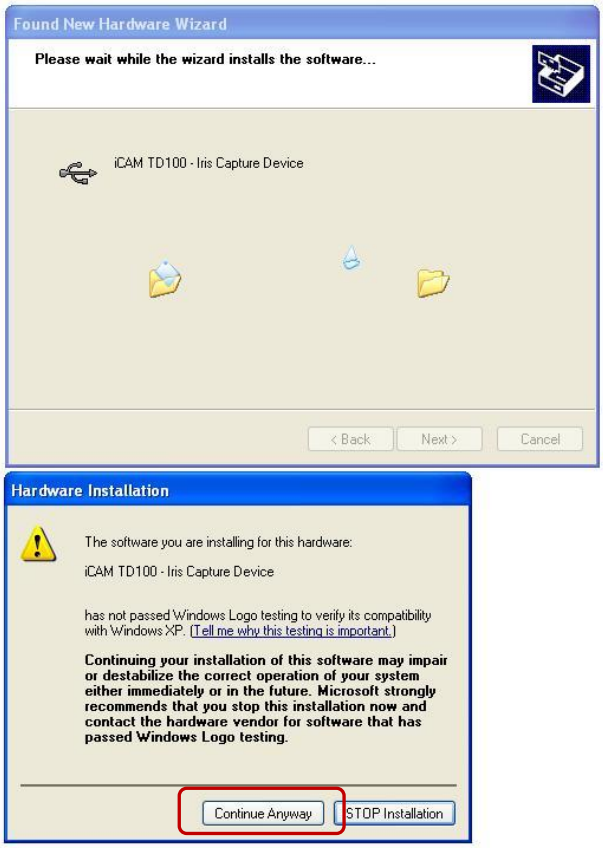


Fig. 16

11. Select > "**Finish**" at the "Completing the Found New Hardware Wizard" window for *iCAM TD100 - Downloader* as shown in Figure 17.

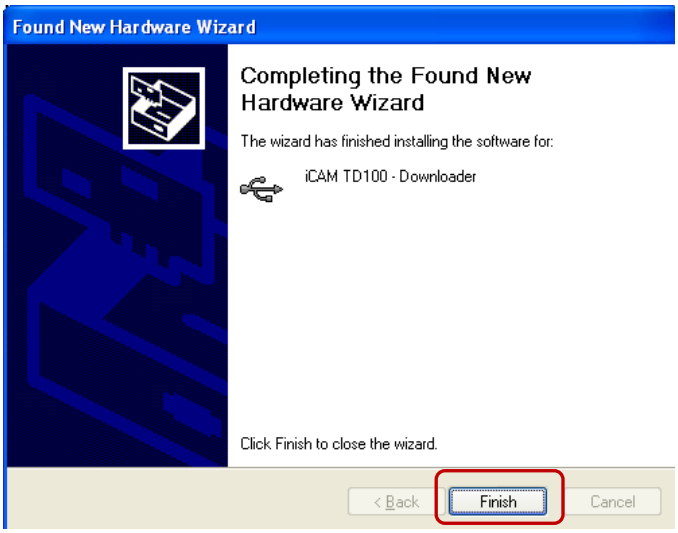


Fig. 17

12. Driver installation for the iCAM TD100 should now be completed. A message indicating "Found New Hardware – Your new hardware is installed and ready to use" may appear at the end of a successful driver installation process as shown in Figure 18.

Fig. 18

***Note:** If you are unable to associate a device driver using the methods provided in this document. Verify that the Operating system version, version of TD100 SDK Software, and iCAM TD100 device are compatible with each other. Remember, 64-bit versions of Windows Vista are not supported with USB\Vid_2285&Pid_1f10 & newer Hardware devices. Also, the TD100 SDK v.2.x and above are also not compatible with Windows Vista. In addition, the iCAM TD100 camera units with USB\Vid_2285&Pid_1f10 & newer are not compatible with versions of TD100 SDK Software 2.x (and above).*

If you experience an issue with compatibility either change the OS that is being used, upgrade the iCAM TD100 Firmware (if available), or use the appropriate version of iCAM TD100 Software as needed.

For additional information contact Iris ID for specific Compatibility requirements and options.

13. A message "*Found New Hardware Wizard*" will appear. Select "**Install from a list or specific location (Advanced)**" radio button and **Next** to continue (as shown in Figure 19).

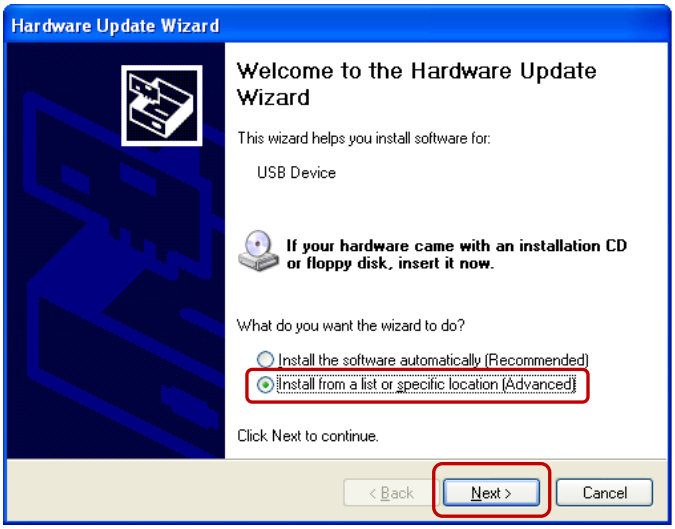


Fig. 19

14. Select "**Don't search. I will choose the driver to install**" radio button and click **Next** to continue As (shown in Figure 20).

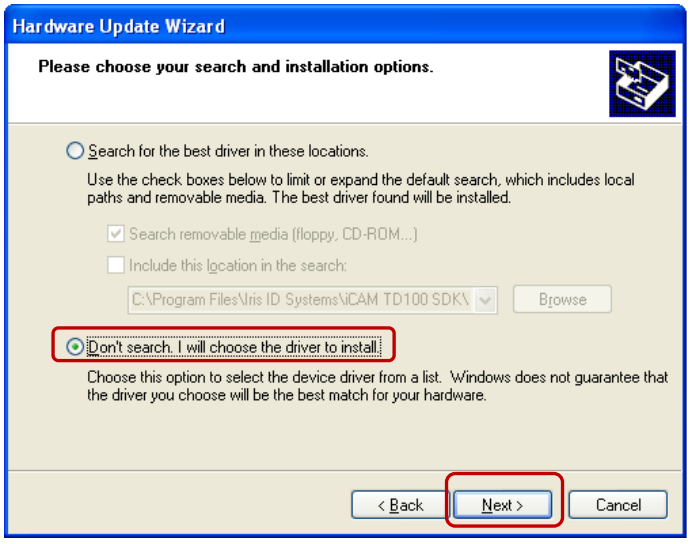


Fig. 20

15. Select Universal Serial Bus controllers and click **Next** (As shown in Figure 21).

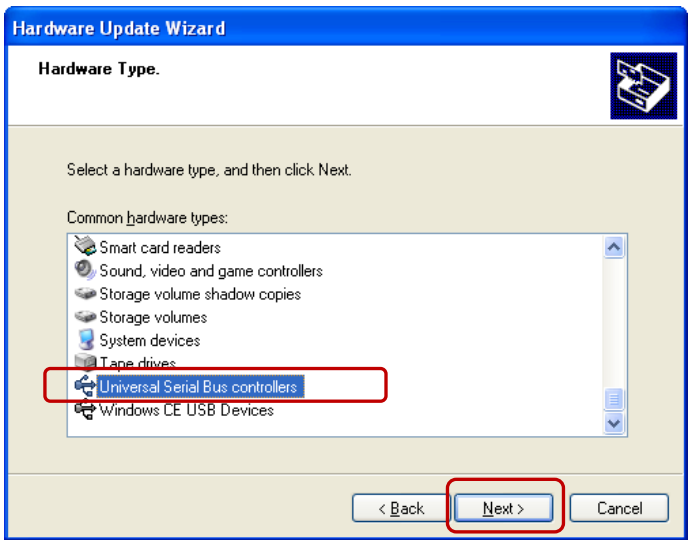


Fig. 21

16. Select **Have Disk** button as shown in Figure 22.

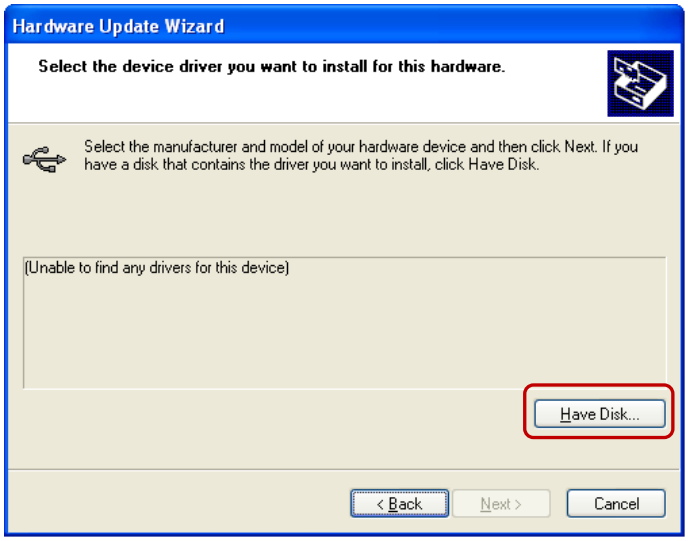


Fig. 22

17. Click Browse as shown in Figure 23.

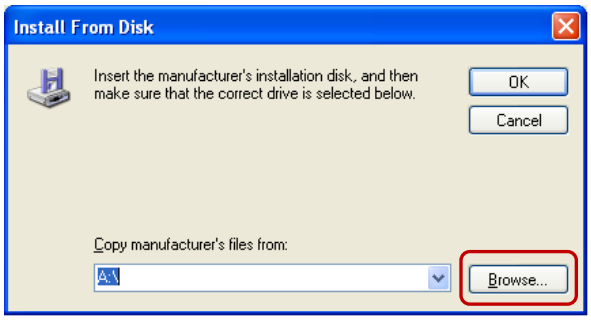


Fig. 23

18. Determine the iCAM TD100 driver you need to install based on the chart (located in Step 1 of section 8.3). For this example we will be updating the iCAM TD100 Vid _043e&Pid_88f7 Hardware with Windows XP 32-bit OS:

"C:/Program Files/Iris ID Systems/iCAM TD100 SDK/Drivers/TD100/VID_043E_PID_88F6", select "TD100.inf" and press > "**Open**" (as shown in Figure 24).

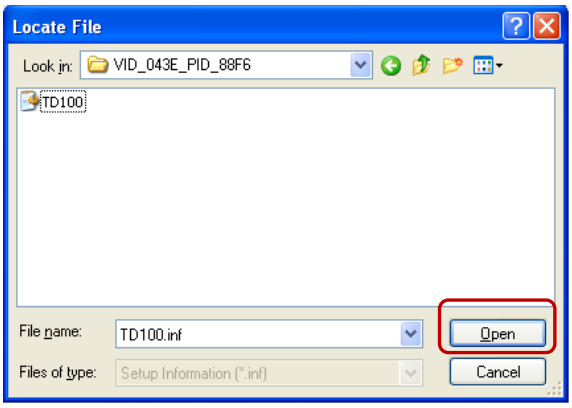


Fig. 24

19. Select **Next** (as shown in Figure 25).

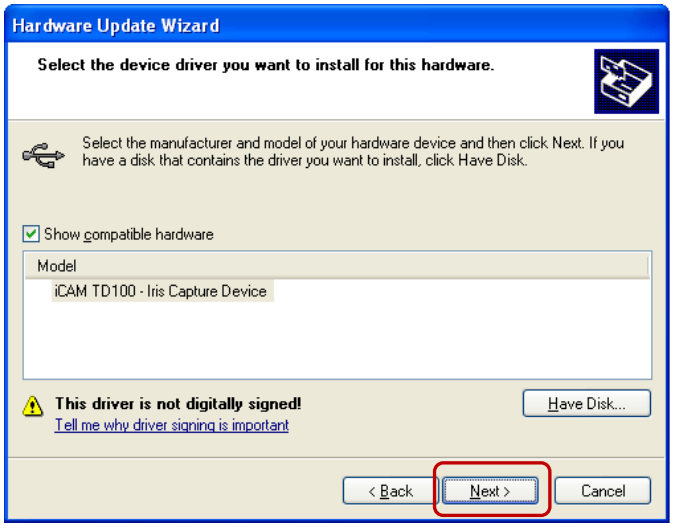


Fig. 25

20. Select **Continue Anyway** if the Hardware Installation window appears (as shown in Figure 26).

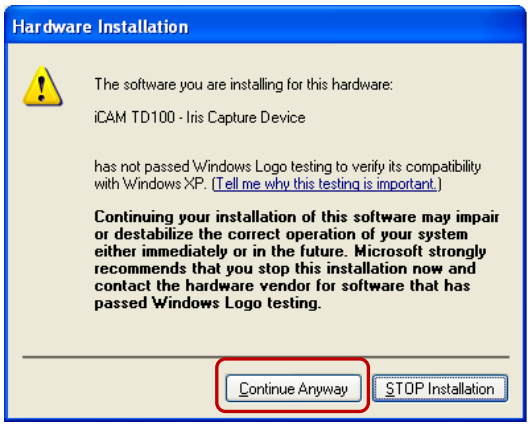


Fig. 26

21. Upon successful hardware installation, Click **Finish** to complete driver installation (as shown in Figure 27).

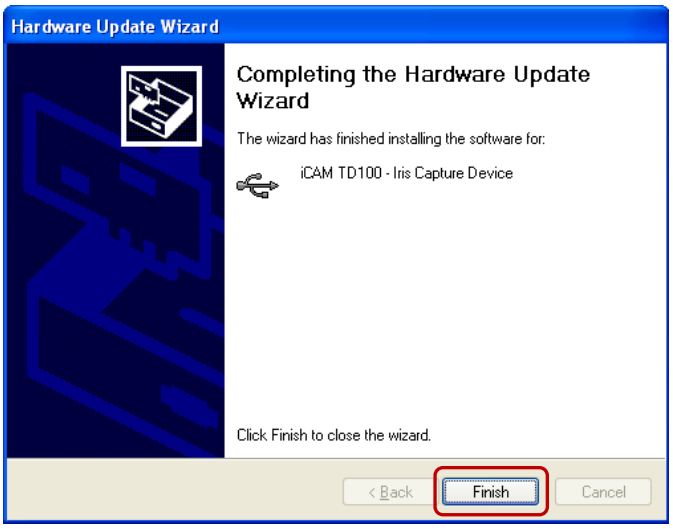


Fig. 27

8.4 FAQ

Please read below for commonly asked questions and answers for your Iris ID iCAM TD100 Product and SDK software.

Q: The iCAM TD100 will not power on, why?

A: The iCAM TD100 requires the Power + USB adapter cable to be connected. Make sure that this cable along with the Power Adapter are properly connected and plugged into a working power outlet. Additionally, make sure that the USB port being used is USB 2.0 compliant.

Q: Why must I install the software before I connect the USB cable to the PC?

A: The iCAM TD100 SDK software includes device drivers for your product. The USB driver software must be installed on the PC prior to connecting the camera unit in order to function correctly. Because the camera unit requires a USB driver to properly run the camera unit, the software must be installed first in order to avoid any attempts of your Operating System to try and install incompatible generic drivers for your device.

Q: The iCAM TD100 camera unit has locked up, or is displaying a strange image on the LCD screen, why?

A: Like any device, it is possible for the iCAM TD100 to lock-up or act strangely from time to time. If this occurs, close any software that may be running to communicate with the iCAM TD100 and disconnect the USB cable from the USB port on the PC. Make sure that the USB + Power cable is properly fastened to each other by disconnecting and re-connecting the connecting cable. Plug the USB port back in to the PC and start the application to re-initiate the camera unit.

Q: After a successful installation of the product, how does the iCAM TD100 perform an iris template creation and/or matching?

A: Although the iCAM TD100 comes bundled with driver files and a demo application, iris template creation and matching are not included as part of this TD100 SDK software. The iCAM TD100 SDK does not include IAiCAMIris.dll (matching and quality assessment functions). However the sample program code in the iCAM TD100 SDK does provide detail on how to call IAiCAMIris.dll for iris quality assessment and matching functions. If you need IAiCAMIris.dll functionality, please contact Iris ID sales dept. at: sales@irisid.com or call 609-819-4747 and select option 2. Additionally, For large scale 1:N matching applications the Iris ID's Iris IrisAccelerator™ can be used. Please visit http://www.irisid.com/irisaccelerator for more information on the IrisAccelerator™.

Q: What version of .NET Framework is being used with the iCAM TD100 SDK?

A: Natively, the iCAM TD100 SDK uses .NET Framework version 3.5. Without version 3.5 (or higher) installed on your PC the sample application may not run correctly. Additionally, the .NET Framework 3.5 is included as part of the installation package.

Q: What version of Visual Studio can be used with this iCAM TD100 SDK?

A: Visual Studio 2008 Service Pack 1 (SP1) is suggested for use when recompiling the sample application. However, if a developer wants to write his/her own application (without using the sample application), VS 2003, and VS 2005 will be compatible for use in addition to VS 2008.

Q: Where can I find the most current version of software?

A: The iCAM TD100 SDK may often be downloaded from the Iris ID Website at: www.irisid.com

Q: Why does my iCAM TD100 not work with Windows Vista 64-bit OS Versions?

A: The iCAM TD100 USB\Vid_2285&Pid_1f10 & newer is not compatible with Windows Vista 64 bit Operating systems. Windows Vista 32-bit Operating Systems are supported.

Q: Why is my iCAM TD100 device not working with Windows 7 64-Bit OS?

A: Only iCAM TD100 camera units with USB\Vid_2285&Pid_1f10 & newer are compatible with Windows 7 64-Bit OS. In order to use the TD100 with USB\Vid_2285&Pid_1f10 & newer, the x64-Bit setup version of the iCAM TD100 SDK is required.

**Note:* If attempting to use iCAM TD100 USB\Vid_043e&Pid_88f7 & older it will be necessary to upgrade the TD100 camera firmware for compatibility usage. Please view the following matrix chart below for compatibility.

Q: Is my new iCAM TD100 device compatible with my old 1.x TD100 SDK software?

A: No. It will be required to upgrade the software version to iCAM TD100 SDK version 2.x and above. Please view the following matrix chart below for compatibility.

Q: Is my older iCAM TD100 device compatible with newer software such as TD100 SDK version 2.03 or 2.05?

A: Yes. In most cases, an iCAM TD100 issued prior to the release of TD100 SDK v2.03 is compatible. Please view the following matrix charts in section 9.2 for compatibility:

9. Appendix

9.1 Compatibility between Hardware & Software versions

When using an iCAM TD100 device, it is important to know which version of firmware you have for your unit. The version of iCAM TD100 Software (firmware) being used will help determine compatibility between Operating System versions and TD100 SDK software versions needed for use with your TD100 camera unit device. Please see the following notation which assists in determining what version of iCAM Firmware is being used on the TD100 device. *(Additionally, follow section 8.3 in the Troubleshooting section of this manual for understanding of how to find the version of USB Hardware ID.)*

Note:

iCAM TD100 devices w/ Hardware Ids: USB\Vid_2285&Pid_1f10 = Firmware A0.12.10 (newer)

iCAM TD100 devices w/ Hardware Ids: USB\Vid_043e&Pid_88f7 & older = Firmware C0.03.31

9.2 Compatibility Matrix Chart for SDK/Firmware Compatibility

- **SDK / Firmware Compatibility:**

Simple Matrix	Firmware (older)	Firmware (newer)
SDK / Firmware Compatibility	**C0.03.31**	**A0.12.10**
SDK 1.00.03	O	X
SDK 2.0x.00 (x64)	X	O
SDK 2.0x.00 (x86)	O	O

Key:

O = Supported

X = Not Supported

- **Detailed Matrix OS Compatibility:**

Detailed Matrix OS Compatibility	Firmware Ver.	XP (32-bit)	Vista (32-bit)	7 (32-bit)	7 (64 bit)
SDK 1.00.03	**C0.03.31**	O	O	O	X
	A0.12.10	X	X	X	X
	A1.06.17	X	X	X	X
SDK 2.0x.00 (x64)	**C0.03.31**	X	X	X	X
	A0.12.10	X	X	X	O
	A1.06.17	X	X	X	O
	A1.11.08	X	X	X	O
SDK 2.0x.00 (x86)	**C0.03.31**	O	O	O	X
	A0.12.10	O	O	O	X
	A1.06.17	O	O	O	X
	A1.11.08	O	O	O	X

Key:

O = Supported

X = Not Supported

9.3 Upgrading the iCAM TD100 Software (Firmware)

If an upgrade of the iCAM TD100 Software (also known as iCAM TD100 Firmware) is required, please refer to the www.irisid.com website for up-to-date software download options that may be available. For further questions or assistance, please contact Iris ID directly.

10. Technical Support

Additional information and technical assistance are available on the Iris ID Systems' support web site at www.irisid.com, click on Support & Service then Technical Support.