Readme for the IQeye ActiveX control version 1.0.3.5

The IQeye ActiveX control allows you to easily embed IQeye image streaming and control into your application or web page.

Install the files

Run the installer. This will copy and register the IQeye control on your system.

If you're redistributing this and need to incorporate the installation into your own install package, do the following:

Copy iqeye.ocx to your windows system directory
From a command prompt run "regsvr32 <full_path_to_file>\iqeye.ocx" to register the control.

Using the IQeye ActiveX (ocx) control in VC++

Start VC++
Create a new MFC AppWizard Dialog based project:
File->New...
Click on Projects tab
Select MFC AppWizard (exe)
Type in a name for the project, say "test"
Select "Dialog based"
Click on "Finish" to take all the defaults

Add the control to your toolbox:
Project->Add To Project->Components and Controls
Select "Registered ActiveX Controls"
Select "Iqeye Control"
Click "Insert"
Click "OK" when it asks you to verify
Click "OK" in the "Confirm Classes" dialog
Close the component selection dialog.

Select the Iqeye control from the toolbox and place it onto your dialog box.

Right click on the control and choose Class Wizard.
Select Member Variables tab and select IDC_IQEYECTRL1 and click "Add Variable..."

Type the name of the new variable, "m_iqeye" for example and click OK.

Close the Class Wizard dialog.

Go to Class View and expand the CTestDlg class and double click on OnInitDialog() to edit that function.

Add this to the OnInitDialog function where it says:

```
// TODO: Add extra initialization here
m_iqeye.SetUrl ("http://<camera>/now.jpg?snap=spush");

where <camera> is the ip address or hostname of the iqeye camera.

Compile and run.  You should get streaming images in the dialog.

Using the control in a web page
===============================

iqeye.html.dp is a sample web page which uses the IQeye control.  It
can be ftp'd to the iqeye camera.  If you then request iqeye.html with
IE, you will get a streaming image.

iqeye.html uses an absolute URL and does not have to live on the
camera.  Substitute your camera's hostname or IP address for "mycamera".


=============================================================================
Properties
=============================================================================
Display
     Type: boolean
     Default: true
     Description: If true, decodes and displays incoming jpegs.  If
     false, does not decode or display incoming jpegs, but still receives
     the stream.

Reconnect
     Type: boolean
     Default: true
     Description: Automatically reconnect if the connection is
     broken.  The delay before reconnect is controlled by
SetReconnectTime.

Url
     Type: string
     Description: complete URL to receive video stream.
     Ex:
       "http://mycamera/now.jpg?snap=spush" (for stream)
       "http://mycamera/now.jpg?snap=pre"   (for single image)
     When this value is set the stream will start.  You do not have to
     call StartDownload

InternalDptz
     Type: boolean
     Default: true
     Description: If true, the control will dptz on mouse click
     events.  If false, it will ignore the mouse.  The default
     is true.  The mouse behaves differently based on StreamDptz.

     Mouse Control for StreamDptz = true:
     o single left click    = zoom in
```

```
     o single right click   = zoom out
     o left click and hold  = continuous zoom in
     o right click and hold = continuous zoom out
     o middle click         = recenter
     o click and drag       = adds pan/tilt

     Mouse Control for StreamDptz = false:
     o click to recenter for all buttons

StreamDptz
     Type: boolean
     Default: true
     Description: If true, the control will use per-stream dptz
     instead of local dptz.  This results in better image quality and
     lower bandwidth requirements on cameras which support this
     feature.  Currently IQeye6xx cameras do not support stream dptz.

Logging
     Type: boolean
     Default: false
     Description: If true, the control logs certain events to the system
log.

Windowless
     Type: boolean
     Default: false
     Description: If true, display in windowless mode, which allows
     drawing on top of the control.

OffsetX
     Type: long
     Default: 0
     Description: x-offset to display area

OffsetY
     Type: long
     Default: 0
     Description: y-offset to display area

DisplayWidth
     Type: long
     Default: 0
     Description: width of display area within control's window.  If
zero, use full width of control.

DisplayHeight
     Type: long
     Default: 0
     Description: height of display area within control's window.  If
zero, use full height of control.

Smoothing
     Type: boolean
     Default: true
```

Description: If true, use smooth stretching mode. This gives a nicer image but takes more processing power. If false, use a faster stretching mode, but results in jagged edges.

==========
Events
==========

Click
Description: Occurs when there is a mouse click within the control.

Error
Description: Occurs when the control reports an error message.

ConnectionBroken
Description: Occurs when the connection is broken.

ConnectionFailed
Description: Occurs when the connection fails.

ConnectionEstablished
Description: Occurs when the connection is (re)established.

NewImage
Description: Occurs when the control receives a new image

MouseDown
Description: Occurs on a mouse down event within the control

MouseMove
Description: Occurs on a mouse move event

MouseUp
Description: Occurs on a mouse up event within the control

TriggerImage (LPCTSTR reason)
Description: Occurs when the control receives a trigger image
In order for this to work, triggering has to
be enabled on the camera, and "pragma=trigger" has to be
specified as part of the URL.
Example:
"/now.jpg?snap=spush&pragma=trigger"

'reason' is the trigger reason string. Possible values are:
"input" : input trigger
"timer" : timer based (periodic) trigger
"motion <windows>" : motion window trigger
"test" : test trigger
Example:
"motion 2 3"
if motion windows 2 and 3 triggered

FileSaved (LPCTSTR remote, LPCTSTR local, long total, long result)
Description: Occurs when we have finished an asynchronous file

download.

    'remote' is the remote name specified in SaveFileAsync
    'local' is the local file name
    'total' is the total number of bytes written
    'result' is the result code for SaveFileAsync. See SaveFile

==========
Methods
==========
long SetOid (LPCTSTR oid, LPCTSTR value)
Sets oid to value
ex: rv = m_iqeye.SetOid ("2.4", "mycamera");
All OID values are expresses as strings, so an OID which takes a float
of 0.45 will be "0.45"
You must call Authenticate() before calling SetOid(). Also, SetOid()
will fail currently if the camera is set for secure authentication (as
opposed
to basic).

Returns 0 on success, else an error code.

long GetOid (LPCTSTR oid, BSTR* value)
Retrieves the value of oid.
ex:
BSTR camera_name;
long rv;
rv = m_iqeye.GetOid ("2.4", &camera_name);
SysFreeString (camera_name);
All OID values are expresses as strings, so an OID which takes a float
of 0.45 will be "0.45"

Returns 0 on success, else an error code. If the call succeeds, you
will need to call SysFreeString on the BSTR.

BSTR GetOidString (LPCTSTR oid)
A variant of GetOid which returns the result instead of using a
pass by reference parameter.

long Authenticate (LPCTSTR hostname, LPCTSTR username, LPCTSTR password);
Establishes credentials for a camera. You must do this
before any calls to SetOid(). You only need to call this once per
hostname, and you can only be authenticated to one camera at a time.
Calls to this function will invalidate any previous authentication and
start a new one.

"hostname" is the ip address or ip hostname of the camera, NOT the url.
ex: Authenticate ("camera_ip", "root", "system");
"hostname" can also take an optional http port number. It defaults to
80.
ex: Authenticate ("192.168.1.2:8000", "root", "system");

Returns 0 on success, else an error code. Success does not
necessarily mean that your username/password is correct;

authentication checking doesn't happen until a priviledge operation is attempted, like SetOid(). The camera must be set for basic authentication for subsequent calls to SetOid() to work.

void StartDownload()
Starts downloading an image stream from the Url.

void StopDownload()
Stops the image stream.

void SetReconnectTime (long msec)
Sets the time, in milliseconds, between reconnection attempts if Reconnect is true.

void SaveImage (LPCSTR FileName)
Saves the current jpeg image to a file.

void LoadImage (LPCSTR FileName)
Loads a file into the current jpeg image.

long ZoomIn(float factor)
Zooms in by factor.

long ZoomOut(float factor)
Zooms out by factor.

long ZoomTo(float factor)
Zooms to a given factor. Minimum zoom is 1.0.

long PanLeft(short increment)
Pans left by 'increment' pixels.

long PanRight(short increment)
Pans right by 'increment' pixels.

long TiltUp(short increment)
Tilt up by 'increment' pixels.

long TiltDown(short increment)
Tilt down by 'increment' pixels.

[Note that for the pan/tilt methods, you can only pan/tilt if you are zoomed in.]

long SaveAs()
Brings up a SaveAs dialog box to allow you to save a jpeg.

void AboutBox()
Brings up the control's about box.

bool GetClicked()
Returns true if the user has clicked in the control.

void ClearClicked()

Sets the control's clicked value to false.

long StartRecording (LPCTSTR filename, long duration)
Start recording live frames to an avi file.

filename: name of file (with full path) to record to.
          If NULL or "" then record to a temp file and
          prompt the user for the real file name when
          recording stops

duration: number of seconds to record. If zero, record
          until StopRecording is called.

Returns:
  0 = success
 -1 = already recording
 -2 = unable to create temp file
 -3 = unable to open file for writing
 -4 = unable to create the avi stream in the file
 -5 = unable to set the MJPEG stream format in the file

long StopRecording ()
Stop recording the AVI file. If no file name was passed to
StartRecording then the user will be prompted for a destination file.

Returns:
  0 = success
 -1 = not recording
 else errno from rename()

float BytesRecorded ()
Returns number of bytes recorded since StartRecording(). This is
actually a large integer returned as a float.

float FramesRecorded ()
Returns number of frames recorded since StartRecording(). This is
actually a large integer returned as a float.

float AvailableRecordingSpace()
Returns bytes of free space for recording. This is
actually a large integer returned as a float.

float FramesDisplayed()
Returns number of total frames displayed. This is
actually a large integer returned as a float. Callers can use this to
get the display rate.

bool Recording()
Returns true if we are recording and false if we are not recording.
The control will stop recording on various error conditions or when
the record duration is passed.

long SaveFile (LPCTSTR remote, LPCTSTR local)

Reads 'remote' file off camera and saves to 'local' file on the local machine. Url must be set before this is called.

remote: name of file on camera, with or without leading '/'
      e.g.: "/ram/test.jpg"
            "ram/trig+0.jpg"
            "hdisk/test.txt"
local: local file path specification. If "" then the control displays
      a "SavAs" dialog
      e.g.: "c:\foo\bar\test.jpg"

Returns:
  0 = success
 -1 = unable to open local file for writing
 -2 = file write error
 -3 = network failure
 -4 = url not valid

long SaveFileAsync (LPCTSTR remote, LPCTSTR local)
Same functionality as SaveFile(), but runs asynchronously. Returns after the download has been started with the expected local file size. When the download is complete the FileSaved event is fired. Download progress can be checked with BytesSaved().

Returns:
 >0 = expected local file size
 -1 = unable to open local file for writing
 -2 = file write error
 -3 = network failure
 -4 = url not valid

long BytesSaved (LPCTSTR remote)
Returns the number of bytes saved from 'remote'.
Returns zero if 'remote' is not currently downloading.

float CurrentSequence()
Returns the current frame sequence number. This is the sequence number which is parsed out of the JSON comment in the JPEG header.

BSTR JsonObject()
Returns the JSON comment for the current frame as a string.