

# *Programmer's Guide for Web Solutions*



*Wislon Technology Corp.*

*Addr: 13F-5, No. 295, Sec. 2, Kuang-Fu Rd., Hsin-Chu 300, Taiwan, R.O.C.*

*Tel: 886-3-5753901*

*Fax: 886-3-5753908*

*Email : raymond@wislon.com.tw*

# 1. Environment Setup

See the procedure running in the **SetupServer.bat** and **SetupClient.bat**. These batch files do the environment setup for the client-server applications.

## Server Side

1. Install the IIS(Internet Information Services) 4.0 or above.
2. The server side needs the following files:

<b>WebClient.asp</b>	the main form of the demo
<b>WebEnroll.asp</b>	the form to save the user info to the database.
<b>WebVerify.asp</b>	the form to verify the template of the known user with the user's enrolled template in the database. (1 to 1 matching)
<b>WebIdentify.asp</b>	the form to identify the template of the unknown user with the enrolled templates in the database (1 to N matching). It is suggested the database be not more that 100 templates.
<b>WebSvrMatch.dll</b>	the Dll to do the matching process.
<b>WebFileIO.dll</b>	the Dll to do the database access. The system integrator can replace the dll by saving the templates directly to the actual database such as Access 、 SQL server 、 Oracle and etc. All the data are saved in "C:\TestLog".
<b>WIS_Ext.dll</b>	an auxialiary dll to use in WebFileIO.dll.

3. Copy \*.asp to C:\inetPub\wwwroot (or another entry point of the server).
4. Copy \*.dll to the system32 or any specified folder.
5. The server needs a fixed IP, let's say, 192.168.1.1.
6. Run regsvr32 /s c:\windows\system32\WebSvrMatch.dll to register the Dll, where c:\windows\system32 indicates any location of the dll.
7. Run regsvr32 /s c:\windows\system32\WebFileIO.dll to register the Dll, where c:\windows\system32 indicates any location of the dll.
8. The WebFileIO.dll need the WIS\_Ext.dll in the same location.

## Client Side

1. The client side needs the following files :

<b><i>Driver</i></b>	
<b>WISCMS12.inf</b> <b>WISCMS12.sys</b>	Driver for the CMOS reader
<b><i>API</i></b>	
<b>WIS_API.ocx</b>	The fingerprint object to run on the browser.
<b>WIS_API.dll</b>	The APIs for use of the OCX.
<b>WISCMS12.dll</b>	The low-level API used by the OCX for CMOS.

2. Copy \*.ocx 、 \*.exe and \*.dll to the system32 or any specific folder.
3. Run regsvr32 C:\windows\system32\WIS\_API.OCX to register the ocx, where C:\windows\system32 indicates any location of the ocx.
4. Run the internet explorer. Select Tools\”Internet Option” and set the safety level to the lowest.

## 2. Function List

### ➤ **WIS\_API.OCX**

All the functions in **WIS\_API.OCX** have their corresponding functions in WIS\_API.dll. The only difference is that the binary data (BYTE \*) in the functions of the OCX or Dll running on the browser are always in **Variant** type. See detailed description of each function in the “ProgrammerGuide.pdf”.

These functions are listed below:

short <b>WISInitDriver</b> (short device);
void <b>WISTerminateDriver</b> ();
short <b>WISTestDevice</b> ();
short <b>WISCheckNoFinger</b> ();
short <b>WISInitCapture</b> ();
short <b>WISEndCapture</b> ();
long <b>WISCapture</b> (short count);
short <b>WISCreateTemplate</b> (VARIANT FAR* rRawTemplate);
short <b>WISSetEnrollMode</b> (short Mode);
short <b>WISEnroll</b> (VARIANT FAR* rEnrTemplate);
short <b>WISReleaseEnroll</b> ();
short <b>WISVerifyTemplate</b> (VARIANT FAR& RawTemplate, VARIANT FAR& EnrTemplate, short Security, long FAR* rScore);
short <b>WISDisplayImage</b> (short nStartX, short nStartY, short nDestWidth, short nDestHeight);
BOOL <b>WISSetParameter</b> (short Brightness, short Contrast, short Gamma);

There are new functions specially designed for interfacing the TunelImage() function of the web applications. The programmer can also design their own user interface and set the parameters of the image through the WISSetParameter().

BOOL <b>WISTunelImageInit</b> (short Brightness, short Contrast, short Gamma) : To set the value of the parameters thorough a displayed dialog.
short <b>WISGetTunedBrightness</b> () : To get the value of the current brightness
short <b>WISGetTunedContrast</b> () : To get the value of the current contrast.
short <b>WISGetTunedGamma</b> () : To get the value of the current gamma.

## ➤ WebSvrMatch.dll

All the parameters of the functions running on the server for use of the browser are always in **Variant** type.

### WIServerVerifyTemplate

#### Synopsis

**WIServerVerifyTemplate(VARIANT rawTemplate, VARIANT enrTemplate, VARIANT securityLevel, VARIANT \*rScore, VARIANT \*rResult);**

#### Parameter

<b>rawTemplate</b>	The fingerprint code generated through <b>WISCreateTemplate()</b> .
<b>enrTemplate</b>	The final fingerprint template generated through <b>WISEnroll()</b> .
<b>securityLevel</b>	A parameter to set the threshold that determines where the verification can be passed. See "ProgrammerGuide.pdf".
<b>rScore</b>	The similarity of two fingerprints to be compared, ranged from 0 ~100. A higher score means a higher similarity.
<b>rResult</b>	The return value. 0 indicates successful verification, otherwise fails.

## ➤ **WebFileIO.dll**

This dll is just for simulating the database process. It provides the functions of saving, loading the enrolled data of the uses. All the data are saved to a predefined directory “C:\TestLog”. The programmers may update the functions of the dll by using their own database, such as Access、SQL Server and etc to insert or select the information of the users.

### **WriteUserData**

#### **Synopsis**

**WriteUserData(VARIANT uid, VARIANT fingerId, VARIANT quality,  
VARIANT enrITemplate);**

#### **Parameter**

<b>Uid</b>	The id or name of the user to be saved.
<b>fingerId</b>	The id of the finger to be saved. 1 ~ 5 corresponds to Right Thumb to Right Little respectively. 6 ~ 10 corresponds to Left Thumb to Left Little respectively.
<b>Quality</b>	The enrolled quality of the finger, must be QUALITY_A ~ QUALITY_D.
<b>enrITemplate</b>	The enrolled template.

### **ReadUserData**

#### **Synopsis**

**ReadUserData(VARIANT uid, VARIANT fingerId, VARIANT\* rQuality,  
VARIANT\* rEnrITemplate, VARIANT\* rResult);**

#### **Parameter**

<b>uid</b>	The id or name of the user to be loaded.
<b>fingerId</b>	The id of the finger to be loaded.
<b>rQuality</b>	The loaded enrolled quality of the finger.
<b>rEnrITemplate</b>	The loaded enrolled template of the user with specified finger Id.
<b>RResult</b>	The return value. <b>0</b> indicates success. <b>-3</b> indicates user is not found. <b>-1</b> indicates finger Id is not between 1 ~10. <b>-2</b> indicates the id/name is blank.

## IsUserExisted

### Synopsis

IsUserExisted(VARIANT uid, VARIANT fingerId, VARIANT \*rExist);

### Parameter

<b>uid</b>	The id or name of the user to be loaded.
<b>fingerId</b>	The id of the finger to be loaded.
<b>rExist</b>	The return value. <b>1</b> indicates existing of the user with specified finger Id. <b>0</b> indicates user is not found. <b>-1</b> indicates finger Id is not between 1 ~10. <b>-2</b> indicates the id/name is blank.

## GetUserCount

### Synopsis

GetUserCount(VARIANT\* rCount);

### Parameter

<b>rCount</b>	The return value. Indicates the number of users in the database.
---------------	--

## GetUserList

### Synopsis

GetUserList(VARIANT inFlag, VARIANT\* rUid, VARIANT\* rFingerId,  
VARIANT\* rQuality, VARIANT\* rEnrITemplate, VARIANT\* rResult);

### Description

This function is used for the identification process. The function will automatically move to the next record for subsequent call. And please free all the resource by setting the inFlag to 1 while the loading is terminated.

### Parameter

<b>InFlag</b>	<b>0</b> indicates to continue loading the user's info. <b>1</b> indicates to terminate loading the data and free the resource.
<b>rUid</b>	The id or name of the user currently loaded.
<b>rFingerId</b>	The id of the finger currently loaded.
<b>rQuality</b>	The loaded enrolled quality of the finger.
<b>rEnrITemplate</b>	The loaded enrolled template of the user with specified finger Id.
<b>rResult</b>	The return value. <b>0</b> indicates success. <b>1</b> indicates success in freeing the resource. <b>-3</b> indicates the end of the records. <b>-2</b> indicates specified user is not found. <b>-1</b> indicates database is not found.

## DeleteUserData

### Synopsis

DeleteUserData (VARIANT uid, VARIANT fingerId, VARIANT\* rResult);

### Parameter

<b>uid</b>	The id or name of the user to be deleted.
<b>fingerId</b>	The id of the finger to be deleted.
<b>rResult</b>	The return value. <b>0</b> indicates success. <b>-3</b> indicates user is not found. <b>-1</b> indicates finger Id is not between 1 ~10. <b>-2</b> indicates the id/name is blank.

## ➤ **Web\_Ext.dll**

The auxiliary DLL is used to convert the binary data to variant type or vice versa. The data of variant type is need in the use of the browser. In addition, to save the binary template to the database such as Access, the variant type is also required.

### **WIS\_BinaryToVariant**

#### **Synopsis**

```
void WINAPI WIS_BinaryToVariant( unsigned char *EnrITemplate, VARIANT  
                                *enrITemplate, int Size);
```

#### **Parameter**

<b>EnrITemplate</b>	The input binary template.
<b>enrITemplate</b>	The output template of Variant type.
<b>Size</b>	The size of the template, should be TEMPLATE_SIZE in this case.

### **WIS\_VariantToBinary**

#### **Synopsis**

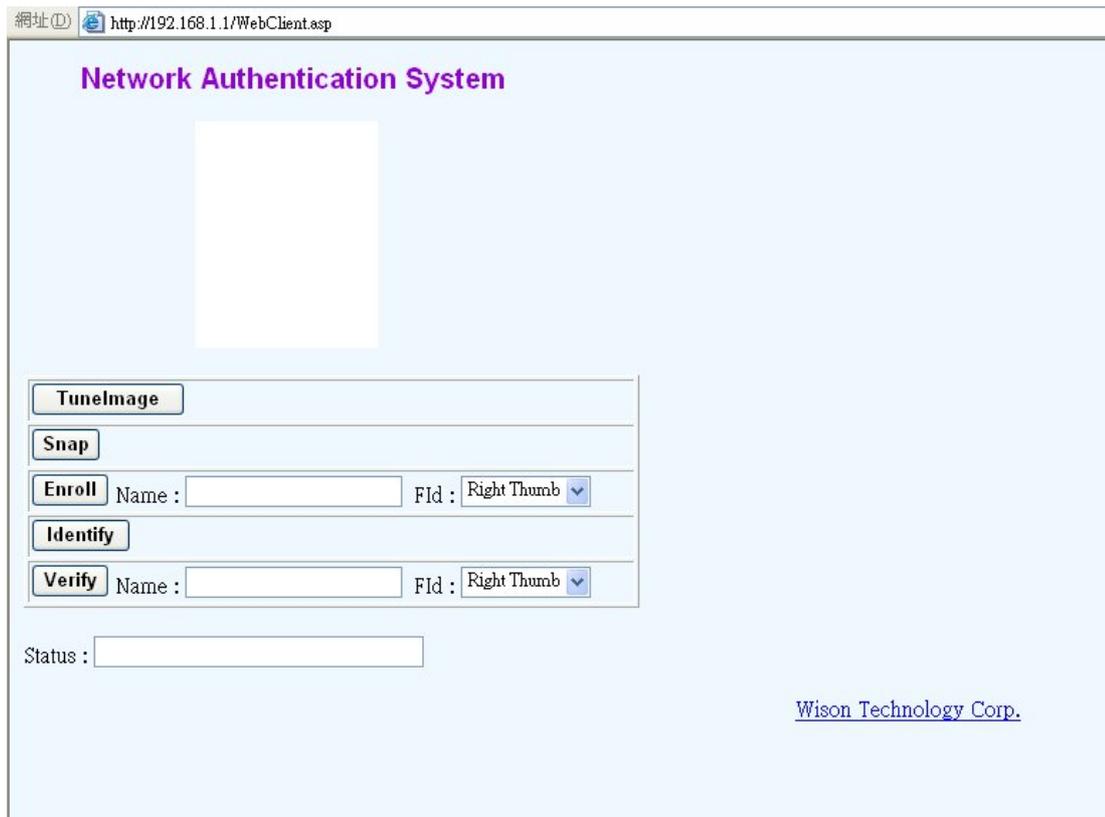
```
void WINAPI WIS_VariantToBinary (VARIANT *enrITemplate, unsigned char  
                                *EnrITemplate, int Size);
```

#### **Parameter**

<b>enrITemplate</b>	The input template of Variant type.
<b>EnrITemplate</b>	The output binary template.
<b>Size</b>	The size of the template, should be TEMPLATE_SIZE in this case.

### 3. Running the program

1. Run the internet explorer.
2. Run the demo by typing 192.168.1.1\WebClient.asp, where 192.168.1.1 refer to the fixed IP of the server.
3. The screen below will be shown on the browser.



The screenshot shows a web browser window with the address bar displaying "http://192.168.1.1/WebClient.asp". The main content area has a light blue background and is titled "Network Authentication System" in purple text. Below the title is a large white rectangular area, likely for a fingerprint image. The interface includes several sections: "Tunelimage" with a button, "Snap" with a button, "Enroll" with a "Name" input field and a "FId" dropdown menu (set to "Right Thumb"), "Identify" with a button, "Verify" with a "Name" input field and a "FId" dropdown menu (set to "Right Thumb"), and a "Status" input field at the bottom left. A link for "Wison Technology Corp." is located at the bottom right.

- A. Click "**Tunelimage**" to tune the brightness and contrast of the device.
- B. Click "**Snap**" to test snapping a fingerprint.
- C. Input a name and select a finger and Click "**Enroll**" to enroll a finger. Once succeeded, the user's info will be sent to the server. All the subsequent process will be done in the WebEnroll.asp.
- D. Click "**Identify**" to do the identification process. The template will be sent to the server and all the identification process will be done in the WebIdentify.asp.
- E. Input a name and select a finger and Click "**Verify**" to verify a finger. The user's info will be sent to the server. All the subsequent process will be done in the WebVerify.asp.
- F. The status will be shown in the status window.
- G. The programmer can change their screen design and flow here.