sgcBiometrics 1.1.0

Table of Contents

sgcBiometrics | Windows Biometric Framework

Every individual has unique characteristics that can be used for identification. Typically these characteristics are physical and include traits such as fingerprints, but they can also include behavioral traits such as gait and typing rhythm. The term biometrics encompasses both meanings. Biometric information is increasingly replacing passwords to identify and verify users. It is more secure and often more convenient for both user and administrator.

Sensors are used to capture biometric information. The information is captured by the sensor as a biometric sample. A single sample contains data that represents a single biometric characteristic for one individual. Multiple samples are averaged to create a biometric template, and the template is securely stored. Later, a sample from an unknown user is compared to the stored templates to establish and verify user identity. The Windows Biometric service, part of the Windows Biometric Framework (WBF), provides the following functionality. You can use the Windows Biometric Framework API to leverage this functionality.

- Captures biometric samples and uses them to create a template.
- Securely saves and retrieves biometric templates.
- Maps each template to a unique identifier such as a GUID or SID.

There are two types of sensor pools: private and system. This section describes each and explains how to create a private sensor pool.

To support both Windows authentication scenarios and vendor defined authentication, the Windows Biometric service organizes biometric units into three possible sensor pools:

- **Private pool:** a collection of biometric units allocated for exclusive use by a client application. Private pools can support authentication scenarios that are not Windows-based, and they make it possible for an application to access the hardware of a biometric unit in a vendor-defined fashion. There can be as many private sensor pools on the system as there are biometric units.
- **System sensor pool:** a collection of sharable biometric units that provide access to Windows authentication services. This pool is used by Winlogon, UAC, and any other client that associates a SID with a specific biometric template. Each biometric service provider has one system sensor pool.

Applications can use the shared system pool or they can create a private pool made up of biometric units removed from the system or unassigned pools. When an application releases its private pool, the biometric units are reconfigured and returned to their original pools. To prevent denial of service attacks, only privileged users are permitted to remove the last sensor from the system pool. For more information, see the following topics.

sgcBiometrics currently has the following components:

1. **TsgcWinBioFingerPrint:** allows to login using a fingerprint sensor.
2. **TsgcWinBioStorageFile:** allows to set a private pool where biometric units are allocated for use by a client application.

SensorPools

sgcBiometrics | System Sensor Pool

The system sensor pool is a collection of sharable biometric units that provide access to Windows authentication services. This pool is used by Winlogon, UAC, and any other client that associates a SID with a specific biometric template. Biometric units in the system pool:

- Can be shared by multiple client applications.
- Send event notices generated by the completion of biometric operations only to the application that has current window focus.
- Use account SIDs to represent the template identities. All of the templates associated with a single user account are tagged with the SID assigned to that account.

Depend on trustworthy template storage provided by the Windows Biometric Service.

A biometric unit can be included in the system pool if it can be:

- Configured to operate in basic mode and act only as a biometric capture device.
- Configured to operate in advanced mode but has no onboard template storage. That is, it must use the storage adapter and template store supplied by Microsoft.
- Configured to operate in advanced mode, contains onboard template storage, and can generate the required hashes.

Check TsgcWinBioFingerPrint to see how to use.

sgcBiometrics | Private Sensor Pool

A private sensor pool is a collection of biometric units reserved for exclusive use by a client application. Private pools support proprietary authentication methods and enable a client application to access a biometric unit by using vendor-specified control commands. Biometric units in a private sensor pool:

- Are available only to the client application that created the pool.
- Send event notices generated by the completion of biometric operations directly to the application without regard to current window focus.

- Use GUIDs to identify biometric templates.
- Share a single, application selected template database.

Private sensor pools must be used if the client application:

- Manages a collection of dedicated biometric units that use an application-specific template database. For an example, consider an employee attendance application where employees signal their arrival at work by swiping their finger on a fingerprint reader. The employees do not have Windows accounts on the computer running the application. Instead, their fingerprints are identified by GUIDs managed by the attendance application.
- Collects biometric samples rather than simply maps samples to SIDs.
- Places the biometric unit hardware into maintenance mode to update the firmware.
- Sends vendor-defined control commands to the biometric unit hardware or software.
- Attempts to configure a biometric unit with onboard storage to operate in advanced mode but the unit cannot perform the required hashing operations.
- Runs from a Remote Desktop client session.

Components

Components | TsgcWinBioFingerPrint

This component allows to login using a FingerPrint reader. By default uses System Sensor Pool, but if you attach a Storage Component, like TsgcWinBioStorageFile, you can setup your own Private Sensor Pool.

Basic Usage (System Sensor Pool)

1. Drop a **TsgcWinBioFingerPrint** in any form or datamodule.

2. Call **InitializeSensors** method to start to use your sensor.

```
TsgcWinBioFingerPrint1.InitializeSensors;
```

a. If initialization is successful, then **OnEnumBiometricUnit** event will be called.
b. If there is any error, **OnError** event is raised.

3. To save your FingerPrint in a Storage Template, call **EnrollBiometric** method

```
TsgcWinBioFingerPrint1.EnrollBiometric(aSubType);
```

Where aSubType can be any of the following:

```
WINBIO_ANSI_381_POS_RH_THUMB
WINBIO_ANSI_381_POS_RH_INDEX_FINGER
WINBIO_ANSI_381_POS_RH_MIDDLE_FINGER
WINBIO_ANSI_381_POS_RH_RING_FINGER
WINBIO_ANSI_381_POS_RH_LITTLE_FINGER

WINBIO_ANSI_381_POS_LH_THUMB
WINBIO_ANSI_381_POS_LH_INDEX_FINGER
WINBIO_ANSI_381_POS_LH_MIDDLE_FINGER
WINBIO_ANSI_381_POS_LH_RING_FINGER
WINBIO_ANSI_381_POS_LH_LITTLE_FINGER
```

Three Events will be called in Enroll process:

OnEnrollBegin: when enrollment starts.
OnEnrollCapture: every time you touch your sensor, you will get information about it, if it's correct sample, if it's bad...
OnErollCommit: when enrollment finishes.

4. To verify if your fingerprint has been saved, just call **Identify** method

```
TsgcWinBioFingerPrint1.Identify;
```

OnIdentify event will be raised and If you a SID or GUID is provided, means sample exists.

```
  Case aIdentity._Type of
    WINBIO_ID_TYPE_SID:             vId             :=
 aIdentity.AccountSid.Data;
    WINBIO_ID_TYPE_GUID:             vId             :=
 aIdentity.TemplateGuid.Guid;
    else
      vId := '';
  End;

  case aSubfactor of
    WINBIO_ANSI_381_POS_RH_THUMB:     vSubFactor     :=
'RH_THUMB';
    WINBIO_ANSI_381_POS_RH_INDEX_FINGER: vSubFactor :=
'RH_INDEX_FINGER';
    WINBIO_ANSI_381_POS_RH_MIDDLE_FINGER:    vSubFactor
:= 'RH_MIDDLE_FINGER';
    WINBIO_ANSI_381_POS_RH_RING_FINGER:  vSubFactor  :=
'RH_RING_FINGER';
    WINBIO_ANSI_381_POS_RH_LITTLE_FINGER:    vSubFactor
:= 'RH_LITTLE_FINGER';
    WINBIO_ANSI_381_POS_LH_THUMB:     vSubFactor     :=
'LH_THUMB';
    WINBIO_ANSI_381_POS_LH_INDEX_FINGER: vSubFactor :=
'LH_INDEX_FINGER';
    WINBIO_ANSI_381_POS_LH_MIDDLE_FINGER:    vSubFactor
:= 'LH_MIDDLE_FINGER';
    WINBIO_ANSI_381_POS_LH_RING_FINGER:  vSubFactor  :=
'LH_RING_FINGER';
    WINBIO_ANSI_381_POS_LH_LITTLE_FINGER:    vSubFactor
:= 'LH_LITTLE_FINGER';
  end;
```

5. To capture a biometric sample just call CaptureSample method

```
TsgcWinBioFingerPrint1.CaptureSample;
```

OnCaptureSample event will be raised is result is successful.

The fingerprint image data is just a grayscale bitmap. Assuming you have a correctly-sized bitmap available, here's how to draw the bitmap. (Note, I'm assuming here that the fingerprint image is 8 bits per pixel. Actually, the pixel width is stored in the aAnsiBdbHeader, and a real application

would have to use the value found there to calculate the proper X and Y values, based on pixel size.)

```
    for y := 0 to aAnsiBdbRecord.VerticalLineLength -
1 do
    begin
      for          x          :=          0          to
aAnsiBdbRecord.HorizontalLineLength - 1 do
      begin
        vByte        :=        aFirstPixel[(y        *
aAnsiBdbRecord.HorizontalLineLength) + x];
        vRGB := RGB(vByte, vByte, vByte);
      end;
    end;
```

Components | TsgcWinBioStorageFile

This component allows to setup a private sensor pool if attached to a biometric component like TsgcWinBioFingerPrint.

Basic Usage (Private Sensor Pool)

1. Drop a **TsgcWinBioFingerPrint** in any form or datamodule.

2. Drop a **TsgcWinBioStorageFile** in any form or datamodule.

3. Link **TsgcWinBioFingerPrint.Storage** to **TsgcWinBioStorageFile**.

4. In **TsgcWinBioStorageFile** you can customize your own **DatabaseId** (must be a GUID).

5. Call InitializeSensors method to start to use your sensor.

```
TsgcWinBioFingerPrint1.InitializeSensors;
```

a. If initialization is successful, then **OnEnumBiometricUnit** event will be called.
b. If there is any error, **OnError** event is raised.

6. To save your FingerPrint in a Storage Template, call **EnrollBiometric** method

```
TsgcWinBioFingerPrint1.EnrollBiometric(aSubType);
```

Where aSubType can be any of the following:

```
WINBIO_ANSI_381_POS_RH_THUMB
WINBIO_ANSI_381_POS_RH_INDEX_FINGER
```

```
    WINBIO_ANSI_381_POS_RH_MIDDLE_FINGER
    WINBIO_ANSI_381_POS_RH_RING_FINGER
    WINBIO_ANSI_381_POS_RH_LITTLE_FINGER

    WINBIO_ANSI_381_POS_LH_THUMB
    WINBIO_ANSI_381_POS_LH_INDEX_FINGER
    WINBIO_ANSI_381_POS_LH_MIDDLE_FINGER
    WINBIO_ANSI_381_POS_LH_RING_FINGER
    WINBIO_ANSI_381_POS_LH_LITTLE_FINGER
```

Three Events will be called in Enroll process:

OnEnrollBegin: when enrollment starts.
OnEnrollCapture: every time you touch your sensor, you will get information about it, if it's correct sample, if it's bad...
OnErollCommit: when enrollment finishes.

7. To verify if your fingerprint has been saved, just call **Identify** method

```
TsgcWinBioFingerPrint1.Identify;
```

OnIdentify event will be raised and If you a SID or GUID is provided, means sample exists.

```
  Case aIdentity._Type of
    WINBIO_ID_TYPE_SID:             vId             :=
 aIdentity.AccountSid.Data;
    WINBIO_ID_TYPE_GUID:             vId             :=
 aIdentity.TemplateGuid.Guid;
    else
      vId := '';
  End;

  case aSubfactor of
    WINBIO_ANSI_381_POS_RH_THUMB:     vSubFactor     :=
'RH_THUMB';
    WINBIO_ANSI_381_POS_RH_INDEX_FINGER: vSubFactor :=
'RH_INDEX_FINGER';
    WINBIO_ANSI_381_POS_RH_MIDDLE_FINGER:    vSubFactor
:= 'RH_MIDDLE_FINGER';
    WINBIO_ANSI_381_POS_RH_RING_FINGER:  vSubFactor  :=
'RH_RING_FINGER';
    WINBIO_ANSI_381_POS_RH_LITTLE_FINGER:    vSubFactor
:= 'RH_LITTLE_FINGER';
    WINBIO_ANSI_381_POS_LH_THUMB:     vSubFactor     :=
'LH_THUMB';
    WINBIO_ANSI_381_POS_LH_INDEX_FINGER: vSubFactor :=
'LH_INDEX_FINGER';
    WINBIO_ANSI_381_POS_LH_MIDDLE_FINGER:    vSubFactor
:= 'LH_MIDDLE_FINGER';
```

```
    WINBIO_ANSI_381_POS_LH_RING_FINGER:  vSubFactor  :=
'LH_RING_FINGER';
    WINBIO_ANSI_381_POS_LH_LITTLE_FINGER:    vSubFactor
:= 'LH_LITTLE_FINGER';
  end;
```

Index