signotec Biometric API

Biometric API

signotec GmbH
Am Gierath 20 b
D-40885 Ratingen

+49 (0) 2102/53575-10
www.signotec.com
support@signotec.de

Version 1.4, 2017-06-12

Contents

Legal notice

1 Documenten history

Version	Date	Person responsible	Status/note
1.0	04 January 2013	Valentin Tkachev	Document created.
1.1	15 October 2013	Valentin Tkachev	New front page deposited.
1.2	19 March 2015	Valentin Tkachev	Footer fixed.
1.3	30 May 2017	Paul Grütter	Documentation redesigned completely
1.4	12 June 2017	Paul Grütter	Changes in 2, 3.1.1, 4.1, 4.4

2 Function overview

The signotec Biometric API contains the component SignKrnl.dll and several helper components. The component SignKrnl.dll can be used to perform a biometric comparison of pad signatures. At least two biometric data blocks (SignData format) are required. Usually several reference signatures should be stored in a database, compared to a new signature and the comparison reault should then be calculated as mean value.

The following table provides an overview of the components included in the signotec Biometric API.

Dateiname	Kurzbeschreibung
SignKrnl.dll	Self-registering COM component for comparison for biometric signature data
signtiff.dll	Self-registering COM component used by the SignKrnl.dll. Needed on 32 bit only.
libtif35.dll	Native DLL used by the SignKrnl.dll.
signcap.dll	Visual control element (ActiveX/COM) for capturing signatures. This component is used by the sample applications and documented in the signoAPI.
signcap.ini	Control file for the signcap.dll.
STPadCapt.ocx	Visual control element (ActiveX/COM) used by the signcap.dll.
STPad.ini	Control file for the STPadCap.ocx.
Several Sample Applications	Applications and source code to demonstrate the functions of the SignKrnl.dll.

3 System requirements

3.1 Biometric API components for Windows

The signotec Biometric API for Windows can be run on all Windows versions as of Windows 7. It was tested under the following systems:

- Windows 7
- Windows 8
- Windows 8.1
- Windows 10

3.1.1 Dependencies

The components, applications and their dependencies respectively contained in the Biometric API sometimes require different versions of the Microsoft C++ libraries and / or the Microsoft.NET framework. The following provides you with an overview of the libraries that are required in each case (depending on the set-up variant for the x86 or x64 platforms):

Component	C++ library	.NET framework
SignKrnl.dll	-	-
signtiff.dll	-	-
libtif35.dll	-	-
signcap.dll	-	-
STPadCapt.ocx	Version 10.0 (VS 2010)	-
SignKrnl Demo App.exe (C++)	Version 10.0 (VS 2010)	-
SignKrnl Demo App.exe (C#)	-	Version 2.0

The signoPAD API Setup automatically installs the 'Visual Studio 2010 Redistributables'.

.NET 2.0 is not included in the signoPAD API and must be manually installed if necessary.

4 General information on the signoPAD API components

4.1 32- and 64-bit variants of the Biometric API

The signotec Biometric API is available in both x86 (32-bit) and x64 (64-bit). Windows 64-bit allows for the parallel installation of both set-ups.

The x86 version only contains components and applications that were compiled for the x86 platform. But both the set-up and all the components and applications can also be used on 32-bit and 64-bit versions of Windows.

The x64 version only contains components and applications that were compiled for the x64 platform. Both the set-up and all the components and applications can only be used on 64-bit versions of Windows.

Since the two versions of the components differ neither in name nor in the interface, it does not matter which one is used for development purposes. But the appropriate component for the present target platform must be used in the implementation. The following table shows which version of the components must be used for specific operating system or application versions:

Operating system	Application	Component
x86 (32 Bit)	x86 (32 Bit)	x86 (32 Bit)
	x64 (64 Bit)	not supported
	Any CPU	x86 (32 Bit)
x64 (64 Bit)	x86 (32 Bit)	x86 (32 Bit)
	x64 (64 Bit)	x64 (64 Bit)
	Any CPU	x64 (64 Bit)

4.2 SignKrnl.dll

The SignKrnl.dll is self-registering and supports the Microsoft IDispatch interface. This makes it equally available under environments such as **.NET, Delphi, Visual C++** or **Visual Basic**.

The component must be registered in the system using regsvr32 so that all applications access the same component.

4.2.1 ProgID allocation

Below is the IID for the dispatch interface:

Schnittstellenname	IID
ISignEngn	02E2EBD6-73E3-4867-9592-33EF24325D1E

The CLSID for the dispatch interface:

Schnittstellenname	CLSID
ISignEngn	6531551D-3968-4DF9-9D2C-3F97AA039F14

4.3 SignData structures

For comapring tow signatures they must be available as SignData. This format is an encrypted, compressed, biometric structur that can be stored in a database or as a tag in a TIFF or PDF document.

Please note that the SignKrnl.dll only supports the conventional SignData format. The new RSA encrypted SignData format is currently only supported by signoPAD API components and must be converted into the conventional format where required using one of these components. Users of the RSA-encrypted format must have the appropriate private RSA key otherwise the data cannot be decrypted. For more information please refer to the signoPAD-API documentation.

4.4 Notes for redistribution

You can, of course, redistribute individual files from the signotec Biometric API in a separate Setup. Essentially, only the files SignKrnl.dll, signtiff.dll, and libtif35.dll as well as the Microsoft Runtime files, are required. See also chapter 'Dependencies'. The files SignKrnl.dll and signtiff.dll must be registered in the system using regsvr32.

The signoPAD API Setup installs files at the following locations:

Komponente	Installationspfad
SignKrnl.dll	<Installation path>\Dll
signtiff.dll (32 bit only)	<Installation path>\Dll
libtif35.dll	<Installation path>\Dll
signcap.dll	<Installation path>\Dll
signcap.ini	<Installation path>\Dll
STPadCapt.ocx	<Installation path>\Dll
STPad.ini	%COMMONPROGRAMFILES%\signotec\Config or %COMMONPROGRAMFILES(x86)%\signotec\Config
Various sample applications including source code	<Installation path>\Biometric API
Documentation	<Installation path>\Biometric API\Documentation

5 Methods

5.1 Verify and VerifyVbs methods

Compares the dynamic or static characteristics of two signatures and calculates a similarity specified as a percentage value.

Note: The `SignData` property must be set before the function is called in order to ensure that the component's internal comparison memory is loaded (with the first signature). The second signature is passed when the method is called.

Note: When using JavaScript or VBScript please call the `VerifyVbs()` method, otherwise no return value will be generated (also see `VerifyResult`).

```
HRESULT Verify(VARIANT *pSignData, long *pResult)
```

```
HRESULT VerifyVbs(VARIANT *pSignData, long *pResult)
```

Parameter	Values	I/O	Description
`VARIANT* pSignData`	`!= NULL`	I	SignData as byte array in a `VARIANT` of the type `VT_ARRAY`
`long* pResult`	`!= NULL`	O	Contains the comparison result on return, where 0 means no and 99 means max. identicalness
Return value	**Values**	**Description**	
`HRESULT`	`S_OK`	Method was executed successfully	
	andere	Error	

5.1.1.1 Implementation in C#

```
Object signData1 = System.IO.File.ReadAllBytes(@"C:\test1.sdb");
Object signData2 = System.IO.File.ReadAllBytes(@"C:\test2.sdb");
int result = 0;
try
{
    signEngn.SignData = signData1;
    signEngn.Verify(ref signData2, ref result);
}
catch (Exception exc)
{
    MessageBox.Show(exc.Message);
}
```

5.2 SetVerifyQuality method

This method can be used to adjust the accuracy of the biometric signature comparison to suit requirements.

```
HRESULT SetVerifyQuality(int nMode)
```

Parameter	Values	I/O	Description
`int nMode`	`0 - 2`	I	Accuracy (a small value means high accuracy; 0 is the default value)
Return value	**Values**	**Description**	
`HRESULT`	`S_OK`	Method was executed successfully	
	andere	Error	

5.2.1.1 Implementation in C#

```
try
{
    signEngn.SetVerifyQuality(1);
}
catch (Exception exc)
{
    MessageBox.Show(exc.Message);
}
```

5.3 GetTiffImage method

Generates an image of the signature from the data in the internal comparison memory. Only functions once `SignData` has been set successfully.

```
HRESULT GetTiffImage(VARIANT* pVarTiff)
```

Parameter	Values	I/O	Description
`VARIANT* pVarTiff`	`!= NULL`	I	Contains a global handle to a TIFF image in fax group 4 on return
Return value	**Values**	**Description**	
`HRESULT`	`S_OK`	Method was executed successfully	
	andere	Error	

5.3.1.1 Implementation in C#

```
[DllImport("kernel32.dll", SetLastError = true, ExactSpelling = true)]
private static extern IntPtr GlobalLock(IntPtr handle);

[DllImport("kernel32.dll", SetLastError = true, ExactSpelling = true)]
private static extern UIntPtr GlobalSize(IntPtr hMem);

[DllImport("kernel32.dll", SetLastError = true, ExactSpelling = true)]
private static extern IntPtr GlobalUnlock(IntPtr handle);

IntPtr hGlobal = IntPtr.Zero;
try
{
    Object tiff = new Object();
    _signEngn.GetTiffImage(ref tiff);
    byte[] tiffArray =
                new byte[GlobalSize(new IntPtr((int)tiff)).ToUInt64()];
    hGlobal = GlobalLock(new IntPtr((int)tiff));
    try
    {
        System.Runtime.InteropServices.Marshal.Copy(hGlobal, tiffArray,
                                                    0, tiffArray.Length);
    }
    finally
    {
        GlobalUnlock(hGlobal);
    }
    System.IO.File.WriteAllBytes(@"C:\test1.tif", tiffArray);
}
catch (Exception exc)
{
    MessageBox.Show(exc.Message);
}
finally
{
    if (hGlobal != null)
        System.Runtime.InteropServices.Marshal.FreeHGlobal(hGlobal);
}
```

5.4 ResetAll method

Sets all internal data to its initial values. Memory that is already allocated is returned to the operating system.

`HRESULT ResetAll()`

Parameter	Values	I/O	Description
-	-	-	-
Return value	**Values**	**Description**	
HRESULT	S_OK	Method was executed successfully	
	andere	Error	

5.4.1.1 Implementation in C#

```
try
{
    signEngn.ResetAll();
}
catch (Exception exc)
{
    MessageBox.Show(exc.Message);
}
```

5.5 SetLicenseKey method

Sets the hardware-independent company licence key. Enables various functions and disables the demo warning.

`HRESULT SetLicenseKey(BSTR bstrLicense)`

Parameter	Values	I/O	Description
`BSTR bstrLicense`	`!= NULL`	I	License key
Return value	**Values**	**Description**	
`HRESULT`	`S_OK`	Method was executed successfully	
	andere	Error	

5.5.1.1 Implementation in C#

```
try
{
    signEngn.SetLicenseKey("1234");
}
catch (Exception exc)
{
    MessageBox.Show(exc.Message);
}
```

5.6 LoadPadData method

Loads the signature's non-standard format (PadData) into the SignKrnl component's internal comparison memory. `Verify()` can be subsequently used to perform a comparison.

The use of this method is no longer recommended. Please use the newer SignData signature format instead. This method is only included and documented for compatibility reasons (i.e., with older programs).

`HRESULT LoadPadData(BSTR bstrPadData)`

Parameter	Values	I/O	Description
`BSTR bstrPadData`	`!= NULL`	I	Data structure of the PadData data type (zero-terminated character string)
Return value	**Values**	**Description**	
`HRESULT`	`S_OK`	Method was executed successfully	
	andere	Error	

5.7 LoadPadDataFromFile method

Loads a reference signature in non-standard format (PadData) from the specified file into the SignKrnl component's internal comparison memory. erify() can be subsequently used to perform a comparison.

The use of this method is no longer recommended. Please use the newer SignData signature format instead. This method is only included and documented for compatibility reasons (i.e., with older programs).

```
HRESULT LoadPadData(BSTR bstrFilename)
```

Parameter	Values	I/O	Description
BSTR bstrFilename	!= NULL	I	Full, absolute path and name of the file in which a PadData structure has been stored.
Return value	**Values**	**Description**	
HRESULT	S_OK	Method was executed successfully	
	andere	Error	

5.8 LoadTiff method

Loads a reference signature in the TIFF fax group 4 format into the SignKrnl component's internal comparison memory. `Verify()` can be subsequently used to perform a comparison. An automatic cleaning process also takes place in the background.

The use of this method is no longer recommended. Please use the newer SignData signature format instead. This method is only included and documented for compatibility reasons (i.e., with older programs).

```
HRESULT LoadTIFF(VARIANT varTiffHandle, int nCleaningMode, int nSplitMode)
```

Parameter	Values	I/O	Description
VARIANT varTiffHandle	!= NULL	I	Global handle (HGLOBAL) to a TIFF image, stored as a `VARIANT` of type `VT_I4`
int nCleaningMode	0	I	Reserved for future functions
Int nSplitMode	0	I	Reserved for future functions
Return value	**Values**	**Description**	
HRESULT	S_OK	Method was executed successfully	
	andere	Error	

5.9 Other methods

All other methods are for internal use only und therefore not documented.

6 Properties

6.1 SignData property

For setting and fetching the signature to be compared in SignData format. Loads the signature into the SignKrnl component's internal comparison memory. `Verify()` can then be used to perform a comparison. Can only ever load one signature.

`VARIANT_BOOL SignData`

Wert	I/O	Description
`!= null`	I/O	SignData as byte array in a `VARIANT` of the type `VT_ARRAY`

6.1.1.1 Implementation in C#

See `Verify()` method.

6.2 VerifyResult property

This property holds the same value as pResult from the `Verify()` call and returns the comparison result after `Verify()` or `VerifyVbs()` has been called.

This is necessary, as the parameter value may not contain a result when JavaScript or VBScript is used and so the result has to be fetched via this property.

`VARIANT VerifyResult`

Wert	I/O	Description
`0 - 99`	O	`VARIANT` of the type `VT_I4` containing the result of the previous call to `Verify()` or `VerifyVbs()`

6.3 XResolution property

Sets the horizontal resolution for the signature (X axis).

`long XResolution`

Wert	I/O	Description
`> 0`	I	Horizontal resolution

6.3.1.1 Implementation in C#

```
try
{
    signEngn.XResolution = 5461;
}
catch (Exception exc)
{
    MessageBox.Show(exc.Message);
}
```

6.4 YResolution property

Sets the vertical resolution for the signature (Y axis).

```
long YResolution
```

Wert	I/O	Description
> 0	I	Vertical resolution

6.4.1.1 Implementation in C#

```
try
{
    signEngn.YResolution = 4096;
}
catch (Exception exc)
{
    MessageBox.Show(exc.Message);
}
```

6.5 TabletSizeLogWidth property

Sets the width of the signature field.

```
long TabletSizeLogWidth
```

Wert	I/O	Description
> 0	I	Width of the signature field

6.5.1.1 Implementation in C#

```
try
{
    signEngn.TabletSizeLogWidth = 640;
}
catch (Exception exc)
{
    MessageBox.Show(exc.Message);
}
```

6.6 TabletSizeLogHeight property

Sets the height of the signature field.

```
long TabletSizeLogHeight
```

Wert	I/O	Description
> 0	I	height of the signature field

6.6.1.1 Implementation in C#

```
try
{
    signEngn.TabletSizeLogHeight = 480;
}
catch (Exception exc)
{
    MessageBox.Show(exc.Message);
}
```

6.7 Other properties

All other properties are for internal use only und therefore not documented.