Software Developer Guide

SigCompare ActiveX Control

Version 1.0

Table of Contents

Overview

INSTALLATION NOTE: The control should be properly setup and registered when running the setup program. To register the control manually, copy "SigCompare.ocx" into the C:\Windows\Sigplus directory. Then, run regsvr32 C:\Windows\Sigplus\SigCompare.ocx.

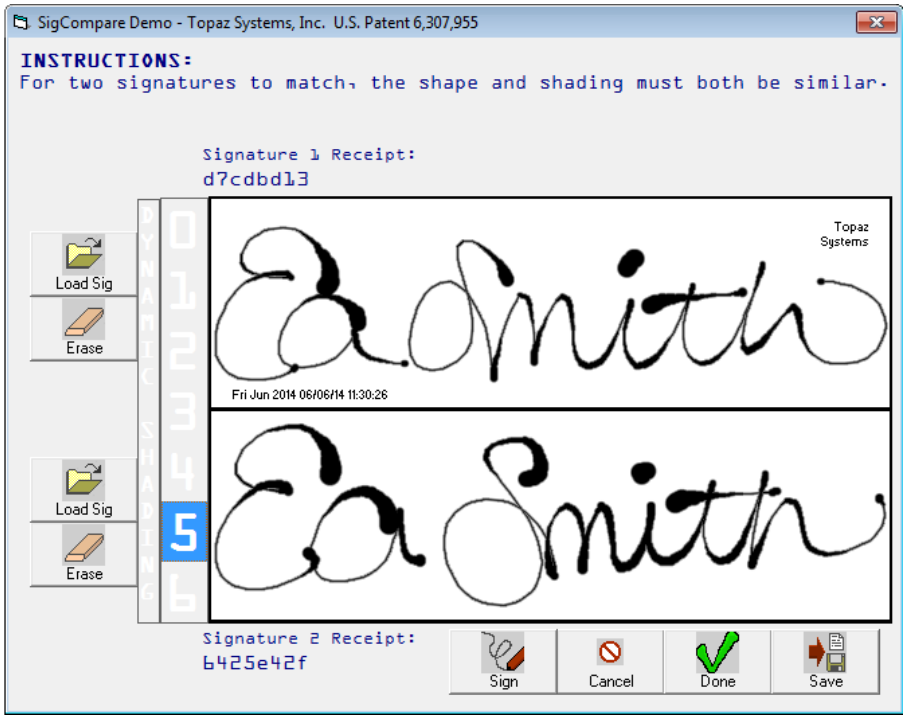
SigCompare.ocx is an ActiveX control that allows the developer to compare two Topaz signatures side by side. Signatures can be viewed with optional, adjustable dynamic shading. Dynamic shading is used to redraw the signature with ink thickness based upon capture velocity. Areas where the author signed slower will appear thicker. Areas where the velocity of the pen was increased will appear thinner. Velocity plays a key role in signature biometrics, and because SigCompare provides visual cues to the signatures velocity, signature comparison accuracy is greatly increased.

Covered under US Patent 6,307,955 - Electronic Signature Management System.
You may review this patent at **www.uspto.gov.**

FOR A VB6 DEMO USING SIGCOMPARE.OCX, PLEASE SEE YOUR WIN\SigPlus\SigCompare\Proj DIRECTORY AFTER INSTALLATION.

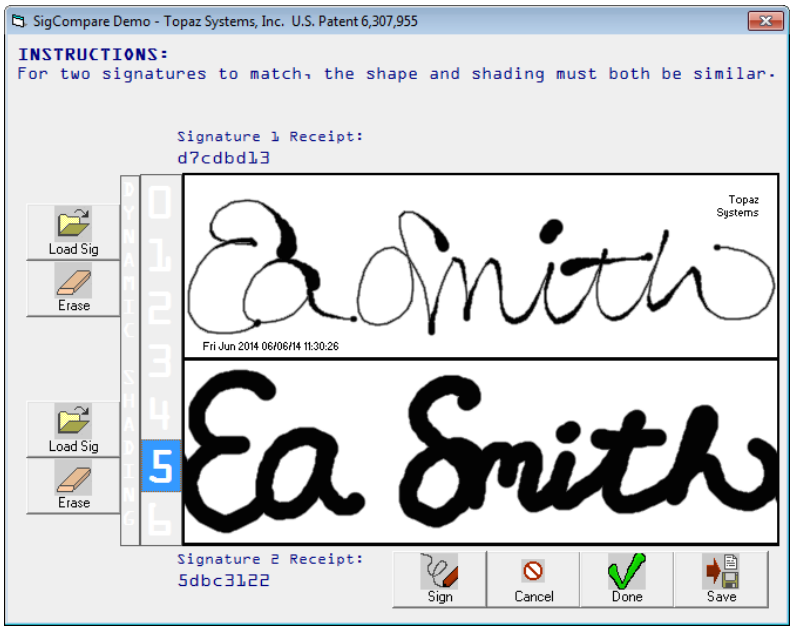
Example 1: True Signer

The upper reference signature is used to compare the authenticity of the lower questioned signature. The lower signature is very likely to be true because it is similar in shape and dynamic shading to the reference; but not identical to the reference.



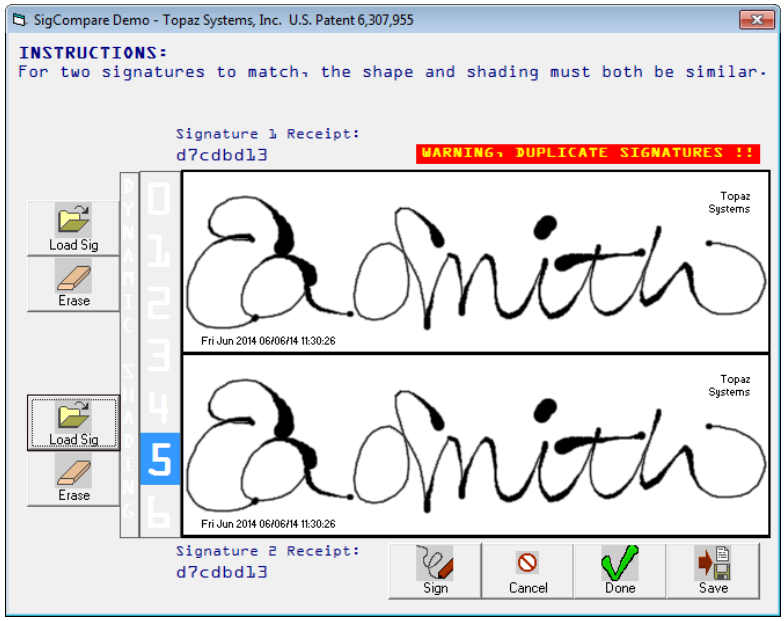
Example 2: Likely Forgery

The upper reference signature is used to compare the authenticity of the lower questioned signature. The lower signature is very likely to be false, because while similar in shape, it exhibits the “slow signing effect” typical of the actions of a forger.



Example 3: Duplicate Signature

The upper reference signature is used to compare the authenticity of the lower questioned signature. The lower signature is a duplicate or copy and therefore is clearly not representative of a unique act-of-signing event. This copy is detected by comparing signature receipts.



Properties

.DynamicShadingLevel =

Integer. (0 to 6)

There are 6 dynamic shading levels, and an “off” position (0).

The shading levels 1, 2 and 3 are standard shading. 1 is best for slower-captured signatures, 2 for medium-speed-captured signatures, and 3 if for high-velocity signatures.

Levels 4, 5 and 6 are exaggerated; the thickness of slower-velocity portions is increased, thus widening the difference between the thinner and thicker portions of the signatures. 4 is best for slower-captured signatures, 5 for medium-speed-captured signatures, and 6 if for high-velocity signatures. This should be used if the difference when using levels 1, 2 and 3 are not great enough for useful comparison. Thicker areas indicate slower pen velocity, and thinner areas indicate higher pen velocity.

DEFAULT: .DynamicShadingLevel = 0

Example:

```
SigCompare1.DynamicShadingLevel = 3 ‘set to level 3
```

Methods

.ClearTablet(TabletIndex As Integer)

This method clears the signature from signature 1 or 2, depending upon the TabletIndex argument.

Return: None

Example:

```
SigCompare1.ClearTablet(1) ‘clears signature 1
SigCompare1.ClearTablet(2) ‘clears signature 2
```

.ExportSigFile(SigPath As String, CompressionMode As Integer, SigIndex As Integer) As Boolean

This method exports the current signature as a Topaz .SIG file:

Arguments:

SigPath: The path to the location you wish to save the .SIG file

CompressionMode: Compresses the signature at the mode supplied (see SigPlus.doc, SigCompressionMode (0 is no compression)

SigIndex: 1=signature1, 2=signature2 (as displayed in the SigCompare control)

Return: Boolean, whether the .SIG file was written successfully

Example:

```
Dim blnRet As Boolean
blnRet = SigCompare1.ExportSigFile (“C:\myfolder\mysignature.sig”, 0, 1)
```

.ExtractSigString(CompressionMode As Integer, SigIndex As Integer) As String

This method returns a Topaz SigString (see SigPlus.doc, SigString property):

Arguments:

CompressionMode: Compresses the signature at the mode supplied (see SigPlus.doc, SigCompressionMode (0 is no compression)

SigIndex: 1=signature1, 2=signature2 (as displayed in the SigCompare control)

Return: String, Topaz SigString

Example:

```
Dim strSigString As String
strSigString = SigCompare1.ExtractSigString (1, 2) ‘mode 1 compression, 2nd signature
```

.LoadSigFile(SigPath As String, CompressionMode As Integer, EncryptionMode As Integer, EncryptionType As Integer, EncryptionString As String, SigIndex As Integer) As Boolean

This method imports a Topaz .SIG file into one of the signature blocks in the SigCompare control.

Arguments:

SigPath: The path to the location of the .SIG file you wish to import

CompressionMode: If the .SIG file has been compressed, you must pass in the mode at which it has been compressed (see SigPlus.doc, SigCompressionMode).

EncryptionMode: If the .SIG file has been encrypted, you must pass in the mode at which it has been encrypted (see SigPlus.doc, EncryptionMode).

EncryptionType: Whether you have encrypted the .SIG file with an external document, or by passing in string values. This is related to whether you did or did not call.

AutoKeyStart() when the .SIG file was encrypted. Calling AutoKeyStart() means you have encrypted the .SIG file to literal string values. Not calling AutoKeyStart() means you have encrypted the .SIG file by passing in a path to a file. If you encrypted using literal string values, this is type 0. If you encrypted by passing in a path to a file, this is type 1.

EncryptionString: The string used to originally encrypt your .SIG file, whether it was a literal string, or a path to a file. If you used multiple strings (calling AutoKeyData more than once) be sure to concatenate your strings to form one string to pass in here.

SigIndex: 1=signature1, 2=signature2 (as displayed in the SigCompare control)

Return: Boolean, whether the .SIG file was loaded successfully

Example:

```
Dim strMyEncString As String
Dim blnRet As Boolean

strMyEncString = ‘obtain this value as necessary

blnRet = SigCompare1.LoadSigFile(“C:\myfolder\mysignature.sig”, 1, 2, 0,
strMyEncString, 1)
‘this will load a .SIG file from the location specified. This .SIG file was compressed at
‘Mode1, Encrypted at Mode2, was encrypted using a literal string. The string used to
‘encrypt it originally is kept in the strMyEncString variable, and this .SIG file will be loaded
‘into the 1st-position signature in SigCompare
```

.LoadSigString(SigString As String, CompressionMode As Integer, EncryptionMode As Integer, EncryptionType As Integer, EncryptionString As String, SigIndex As Integer) As Boolean

This method imports a Topaz SigString into one of the signature blocks in the SigCompare control.

Arguments:

SigString: The original SigString you wish to import (see SigPlus.doc, SigString property).

CompressionMode: If the SigString has been compressed, you must pass in the mode at which it has been compressed (see SigPlus.doc, SigCompressionMode).

EncryptionMode: If the SigString has been encrypted, you must pass in the mode at which it has been encrypted (see SigPlus.doc, EncryptionMode).

EncryptionType: Whether you have encrypted the SigString with an external document, or by passing in string values. This is related to whether you did or did not call AutoKeyStart() when the SigString was encrypted. Calling AutoKeyStart() means you have encrypted the SigString to literal string values. Not calling AutoKeyStart() means you have encrypted the SigString by passing in a path to a file. If you encrypted using literal string values, this is type 0. If you encrypted by passing in a path to a file, this is type 1.

EncryptionString: The string used to originally encrypt your SigString, whether it was a literal string, or a path to a file. If you used multiple strings (calling AutoKeyData more than once) be sure to concatenate your strings to form one string to pass in here.

SigIndex: 1=signature1, 2=signature2 (as displayed in the SigCompare control).

Return: Boolean, whether the SigString was imported successfully.

Example:

```
Dim mySigString As String
Dim blnRet As Boolean

strSigString = 'your original SigString
blnRet = SigCompare1.LoadSigFile(strSigString, 0, 0, 0, "", 2)

'this will load the SIG string from the mySigString variable. This SigString was
uncompressed and unencrypted. This SigString will be loaded 'into the 2nd-position
signature in SigCompare
```

.RefreshSignature(SigIndex As Integer)

This method forces a refresh of the specified signature; useful for refreshing the signature after a new dynamic shading mode is set.

Arguments:

SigIndex: 1=signature1, 2=signature2 (as displayed in the SigCompare control).

Return: None

Example:

```
SigCompare1.RefreshSignature 1 'refreshes the first signature
```

.GetSigReceipt (SigIndex As Integer) As String

This method returns an 8-character hash of the signature, used for easy comparison purposes. If the SigReceipt for both displayed signatures match, you have identical signatures loaded.

Arguments:

SigIndex: 1=signature1, 2=signature2 (as displayed in the SigCompare control).

Return: String – 8-character hash of the signature

Example:

```
Dim strSigReceipt As String
strSigReceipt = SigCompare1.GetSigReceipt
```

.SetTabletState(SigIndex As Integer, State As Integer)

This method opens/closes the port (as set in the SigPlus.ini) for signature capture.

Arguments:

SigIndex: 1=signature1, 2=signature2 (as displayed in the SigCompare control)

State – 0=close, 1=open

Return: None

Example:

```
SigCompare1.SetTabletState(2,1) ‘open signature 2 port for capture
```

.GetTabletState(SigIndex As Integer) As Integer

This method returns the current state of the port.

Arguments:

SigIndex: 1=signature1, 2=signature2 (as displayed in the SigCompare control)

Return: Integer 0=closed, 1=open

Example:

```
SigCompare1.GetTabletState(2) ‘returns the current tablet state of signature 2
```

.GetTabletComPort(SigIndex As Integer) As Integer

This method returns the currently-assigned com port.

Arguments:

SigIndex: 1=signature1, 2=signature2 (as displayed in the SigCompare control)

Return: Integer: com port assignment

Example:

```
Dim intComPort As Integer
strComPort = SigCompare1.GetComPort(2) ‘return com port assignment for signature 2
```

.SetTabletComPort(SigIndex As Integer, ComPort As Integer)

This method sets the com port.

Arguments:

SigIndex: 1=signature1, 2=signature2 (as displayed in the SigCompare control)

ComPort: The com port assignment

Return: None

Example:

```
SigCompare1.SetComPort(1, 1) 'sets signature 1 com port to COM1
```

.GetTabletType(SigIndex As Integer) As Integer

This method returns the current tablet connection type (0=serial, 2=USB, 6=HSB).

Arguments:

SigIndex: 1=signature1, 2=signature2 (as displayed in the SigCompare control)

Return: Integer: Tablet connection type

Example:

```
Dim intTabType As Integer
strTabType = SigCompare1.TabType(1) 'return tablet connection type for signature 1
```

.SetTabletType(SigIndex As Integer, TabletType As Integer)

This method sets the tablet connection type.

Arguments:

SigIndex: 1=signature1, 2=signature2 (as displayed in the SigCompare control)

TabletType: The tablet type to set (0=serial, 2=USB, 6=HSB)

Return: None

Example:

```
SigCompare1.SetTabletType(2, 6) 'sets signature 2 to a Topaz HSB tablet
```

Installation Notes

Topaz Systems SigCompare post-installation directory structure is:

WIN

\SigPlus

SigCompare.ocx (the Fingerprint Capture and Validation Active-X Control)

\SigCompare

SigCompare.exe (The SigCompare demo using SigCompare.ocx)

\Docs SigCompare.doc (this development documentation)

\Proj SigCompare.zip(VB6 demo using SigCompare ActiveX control)

SigCompare Dependency List

VBRuntime files (for SigCompare.exe only)