signoPAD-API

Documentation

Software components for communication with
signotec Sigma, Zeta, Omega, Gamma, Delta and Alpha LCD pads

Version: 8.5.2060
Date: 15.09.2021

© signotec GmbH
www.signotec.de
Tel.: +49 (0) 2102 53575 10
E-mail: info@signotec.de

Contents

Legal notice

1 Document history

Version	Date	Person responsible	Status/note
2.19	30 May 2016	Paul Grütter	Changes in 4.8, 8, 8.1, 8.2, 8.3, 8.11 Various code examples revised
2.20	27 December 2016	Paul Grütter	Changes in 4.5, 5, 7.5, 8.1, 8.6, 8.11, 8.13, 8.14, 8.18, 8.19, 8.20, 8.22, 8.33, 8.35, 8.36, 8.39, 8.40, 8.41, 8.43, 8.44, 8.45, 8.46, 8.47, 8.50, 8.52, 8.53, 8.54, 8.57, 8.58, 8.59, 8.60, 8.64, 8.69, 8.70, 8.75, 8.77, 10.3 New items 7.1.4, 7.2.4, 8.16, 8.17 added
2.21	12 May 2017	Paul Grütter	Changes in 3.1, 3.3, 4.5, 5.2, 6, 7.1.2, 7.5.2, 8.14, 8.48, 8.56, 8.57, 8.60, 8.61, 8.72, 8.73 New items 8.8, 8.12, 8.13 added Items 3.1.2.1, 3.1.3.1 removed
2.22	6 November 2017	Paul Grütter	Changes in 4.8, 6, 8.12, 8.13, 8.53, 8.76
2.23	12 March 2018	Paul Grütter	Changes in 6, 8.17, 8.20, 8.22, 8.23, 8.28, 8.54, 8.56, 8.57, 8.61, 8.62.2, 8.64 Various code examples revised
2.24	8 November 2018	Paul Grütter	Changes in 4.1, 7.2.2, 8.46, 8.55, 8.56, 8.67, 8.68, 8.73.3, 8.74, 8.77, 8.78 New items 8.83, 8.84, 9.11 added Various code examples revised
2.25	16 January 2019	Paul Grütter	Changes in 3.1.1, 8.19, 8.21, 8.22 New item 3.1.2.3 added
2.26	9 April 2019	Paul Grütter	Changes in 8.23
2.27	31 July 2019	Paul Grütter	Changes in 2, 4.8, 5.1, 6, 8.8, 8.14, 8.73
2.28	10 December 2019	Christian Fistelmann, Paul Grütter	Changes in 6, 8.8, 8.21, 8.26, 8.65, 8.71, 8.72, 8.72, 8.77, 9.8, 10.3 New items 8.23, 8.24, 8.27, 8.78, 9.17 added Various code examples revised
2.29	15 June 2020	Christian Fistelmann	Changes in 0, 4.8, 5, 7.4, 7.5, 8.1, 8.6, 8.14, 8.16, 8.23, 8.24, 8.27, 8.45, 8.47, 8.48, 8.49, 8.50, 8.57, 8.71, 8.72, 8.76, 8.78, 8.79, 9.17 New items 7.1.2, 7.2.2, 8.58 added Various code examples revised
2.30	16 November 2020	Christian Fistelmann, Paul Grütter	Changes in 8.30, 8.81, 8.82 New item 0 added Corrections to various formulations
2.31	1 December 2020	Paul Grütter	Various code examples revised
2.32	11 January 2021	Paul Grütter	Changes in 6
8.5.2031	25 June 2021	Paul Grütter	Changes in 2, 4.8, 6, 8.63, 8.40, 8.44, 8.82, 8.84 Various code examples revised
8.5.2060	15 September 2021	Paul Grütter	Changes in 9.15 Various code examples revised

2 Functional overview

The signoPAD API contains visual and non-visual interfaces, allowing programmers to implement a wide range of functions for capturing electronic signatures and displaying graphics, text and buttons on a signotec LCD pad. The 'STPad' components all offer virtually the same range of functions – except for the display on the PC screen – and differ mainly in the technology used (COM, .NET Assembly, Native DLL). The 'STPadLib.dll' component does not contain a visual control element and can therefore not be used directly to display signatures in real time on a PC screen.

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

File name	Short description
STPadCapt.ocx	Visual control element (ActiveX/COM) for activating the Sigma, Zeta, Omega, Gamma, Delta and Alpha model types and for visualising signature data.
STPadLib.dll	Non-visual native library for activating the Sigma, Zeta, Omega, Gamma, Delta and Alpha model types.
STPadLibNet.dll	.NET class library for activating the Sigma, Zeta, Omega, Gamma, Delta and Alpha model types. Contains a non-visual as well as a visual (Windows Forms Control) class.
STPdfLib18.dll	DLL that is only required if the `DisplaySetPDF()`, `PDFLoad()`, `PDFGetPageCount()`, `PDFGetWidth()`, `PDFGetHeight()` or `PDFSelectRect()` method is used. It is not possible to use this DLL directly.
STPad.ini	Control file to set different kind of parameters for the pad communication. In addition, debug logging can be activated for the components listed above.
STPadStores.ini	Control file to assign non-volatile image memories exclusively to an application. Please refer to section 'Exclusive use of non-volatile memory' for details.
signview.dll	Some of the demo applications use SignDraw Control from signview.dll. This file is only included as a demo version. The full version and documentation are available separately in signoAPI.
Various sample applications	Applications and source code in different programming languages to demonstrate the functions of the STPad components.

Please note: The STPad.dll component contained in the signoPad API 8.0.18 or earlier is no longer required. The applications signoReset, signoImager 2 and signoIntegrator 2 that were contained in signoPAD-API 8.0.25 or earlier, are now contained in the separate signoPAD/Tools Setup, which can be downloaded free of charge from www.signotec.de.

3 System requirements

3.1 signoPAD API components for Windows

The signoPAD API for Windows can be run on all Windows versions as of Windows 7. It was tested under the following systems and development environments:

- Windows 7
- Windows 8
- Windows 8.1
- Windows 10
- Microsoft Visual Studio 2010 Professional Edition
- Microsoft Visual Studio 2012 Professional Edition
- Microsoft Visual Studio 2013 Professional Edition
- Microsoft Visual Studio 2015 Professional Edition
- Microsoft Visual Studio 2017 Professional Edition
- Microsoft Visual Studio 2019 Professional Edition
- CodeGear Delphi 2007 Professional Edition
- Embarcadero Delphi XE Architect Edition

3.1.1 Dependencies

The components, applications and their dependencies respectively contained in the signoPAD 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
STPadCapt.ocx	Version 14.1 (VS 2017)	-
STPadLib.dll	Version 14.1 (VS 2017)	-
STPadLibNet.dll	Version 14.1 (VS 2017)	Version 4.0 (Client Profile)
STPadCapt Demo App.exe (C++, Delphi, VB6)	Version 10.0 (VS 2010)	-
STPadCapt Demo App.exe (C# / VB.NET)	Version 10.0 (VS 2010)	Version 2.0
STPadLib Demo App.exe (C++)	Version 10.0 (VS 2010)	-
STPadLibNet Demo App.exe (C#)	Version 10.0 (VS 2010)	Version 4.0 (Client Profile)

The signoPAD API Setup automatically installs the 'Visual Studio 2010 Redistributables', 'Visual Studio 2017 Redistributables' and .NET 4 Client Profile where necessary.

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

All other components of the signoPAD API do not require .NET Framework.

3.1.2 Known problems

3.1.2.1 Visual C ++ and optional parameters

The COM wrapper in Visual C++ does not support the optional parameters of STPadCapt.ocx. In order to be able to use this feature, the wrapper class must be changed manually, for example for DeviceOpen() as in the following example:

```
long DeviceOpen(long nIndex, bool bEraseDisplay=true)
{
    VARIANT vaEraseDisplay;
    vaEraseDisplay.vt = VT_BOOL;
    vaEraseDisplay.boolVal = bEraseDisplay;
    long result;
    static BYTE parms[] = VTS_I4 VTS_VARIANT ;
    InvokeHelper(0x5, DISPATCH_METHOD, VT_I4, (void*)&result, parms,
                 nIndex, &vaEraseDisplay);
    return result;
}
```

More details can be found in the C++ STPadCapt demo application.

3.1.2.2 Multithreading

STPadCapt.ocx must run in the single-threaded apartment (STA). If the STPadCapt.ocx shall be used in an MTA application, you should create an STA Form including the STPadCapt.ocx and call this as a modal dialog from your application.

3.1.2.3 .NET 3.5 and older

Current versions of Visual Studio generate COM wrappers in projects with .NET 3.5 or older for .NET 4.0, resulting in run-time errors. This also applies to the included example projects for the STPadCapt.ocx.

Please use Visual Studio 2010 if you want to use the STPadCapt.ocx in a .NET application with .NET 3.5 or older.

3.1.2.4 .NET 4.0

STPadLibNet.dll has used .NET 4.0 since version 8.1.4. Older .NET versions are no longer supported. However, this also means that the .NET 2.0 runtime activation policy no longer needs to be enabled in .NET 4.0-based projects. The changes previously necessary to the 'app.config' file within the project are no longer required and may be reversed, if necessary.

3.2 signoPAD API components for Java

Please use the signoPAD API Java, which can be downloaded at www.signotec.de.

3.3 signoPAD API components for Linux

Please use the signoPAD API Linux. Please speak to your signotec contact if you are interested.

4 General information on the signoPAD-API components

One of the 'STPad' components is required to activate a signotec LCD pad. The required functionality and technology used determine which component is chosen. For further details, see the following sections.

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

The signoPAD 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. Some demo applications are only compiled for x86 and are not included in this release. 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, only one version has to be referenced. For different reasons, if you are developing using Visual Studio, it is recommended that you use the respective x86 version from the 32-bit set-up during development. The appropriate component for the target platform at hand 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)
	Any CPU (32 Bit preferred)	x86 (32 Bit)
x64 (64 Bit)	x86 (32 Bit)	x86 (32 Bit)
	x64 (64 Bit)	x64 (64 Bit)
	Any CPU	x64 (64 Bit)
	Any CPU (32 Bit preferred)	X86 (32 Bit)

4.2 STPadCapt.ocx

The STPadCapt.ocx component 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.

This component should be used if the signature is to be displayed in a window in real-time. The STPadLibNet.dll component should be used for .NET applications.

This OCX must be registered in the system using regsvr32 so that all applications access the same component. The WoW64 technology from Microsoft allows for the parallel installation and registration of the x86 and x64 versions of OCX.

Generally speaking, the control element is embedded in a window during development; most development environments will then automatically ensure correct initialisation. However, it is also possible to generate the element at runtime; the CreateControl() method is available in this case. For details, please see the Microsoft documentation for ActiveX components.

4.2.1 ProgID allocation

Below is the CLSID for the control:

ActiveX control name	CLSID
signotec Pad Capture Control	3946312B-1829-4D4F-A2DF-CD35C8908BA1

The IID for the dispatch interface:

Interface name	IID
_DSTPadCapt	DBCB0876-0133-4C3A-975D-2463747AC408

The IID for the event dispatch interface:

`Interface name`	`IID`
_DSTPadCaptEvents	30C53BC9-DAF3-423A-A283-BFEF408BD0A9

Below is the ProgID for the control:

ActiveX control name	Prog ID
signotec Pad Capture Control	STPadCapt.STPadCaptCtrl.1

4.3 STPadLib.dll

STPadLib.dll is a native and dynamically loadable library (DLL). A C header file (STPadLib.h) and a library file (STPadLib.lib) are included. This DLL can be used both statically and with dynamic linking; it can therefore be used in all common development environments. Initialisation is performed automatically as soon as the DLL is activated; before it is unloaded again, the `STControlExit()` method must be called to release resources used internally.

This component should be used if there is no window available in which a visual control element can be embedded, or if no real-time display is required. It is also very suitable for a simple program without user interface that only checks, for example, whether signotec pads are connected. The STPadLibNet.dll component should be used for .NET applications.

The DLL cannot be registered in the system, but must be in the application's search path at runtime. Various applications can therefore access various versions of the DLL.

When STPadLib.dll x86 is used with a programming language that uses the stdcall calling convention (for example, Visual Basic 6), methods must be used that are also exported with stdcall. These methods have the additional suffix '_stdcall' (for example, `STDeviceGetCount_stdcall()`), but are otherwise no different from the methods exported with cdecl.

4.4 STPadLibNet.dll

The STPadLibNet.dll is a .NET class library. It can be used in all common .NET-compatible development environments. All classes, enumerations, etc., are contained in the signotec.STPadLibNet namespace.

This component should be used if .NET is used for development purposes. It contains a non-visual as well as a visual (Windows Forms Control) class. The two classes can also be combined.

The component is a 'mixed mode DLL' component and contains both managed and non-managed code. In order that the non-managed objects can be released correctly from the memory, Dispose() should always be called as soon as the instance is no longer required.

The DLL is signed with a 'strong name' and may be registered in the 'Global Assembly Cache' (GAC). However, if the DLL is located in the search path of the application, this version will always be used. The assembly version will only change if the interface is changed and may therefore differ from the file version.

4.4.1 Using the STPadLibControl class

The STPadLibControl class is derived from `System.Windows.Forms.UserControl`. It can be embedded as a visual control element in a Windows form or in a WPF window via a Windows Forms Host. All documented methods, properties and events are available.

4.4.2 Using the STPadLib class

The STPadLib class is designed for communicating without a visual control element. Only the non-visual methods, properties and events are therefore available.

4.4.3 Using both classes

It is also possible to use the STPadLib and STPadLibControl classes simultaneously in a project. For example, a STPadLib object, which is responsible for basic communication tasks (searching for devices etc.), can remain valid during the entire runtime of a program. As soon as the signing process has started, a STPadLibControl object to which the STPadLib object is passed, can be generated in a dialog box (also see `ControlSetSTPadLib()`). This means that a pad search and initialisation does not need to be performed each time the dialog box is generated.

This technology is used in the supplied demo application.

4.5 Using multiple instances

The STPadCapt.ocx and STPadLibNet.dll classes can be instantiated more than once. If multiple instances of a class are used in different memory areas (for example, different programs), these instances are completely independent of each other and there is nothing else to be aware of.

If multiple instances of a class are used in the same memory area, please note the following:

- When `DeviceGetCount()` is called, it is valid for all instances and therefore only needs to be executed in one instance.
- If a connection to a device has already been opened by an instance, only the previously determined value is returned when `DeviceGetCount()` is called in another instance, i.e., no new search is carried out.

4.6 Data formats

The data formats of the individual components vary according to the technology used. The STPadCapt.ocx component uses only OLE-compliant and COM-compliant data types for complex data structures (**VARIANT**, **BSTR**, etc.), while the STPadLib.dll uses types such as **BYTE** and **LPCWSTR** and the STPadLibNet.dll uses **byte** and **string**.

Signatures can be returned in BMP, GIF, JPEG, PNG and TIFF format. Generally speaking, you should use PNG as it offers the best results with the smallest file size. JPEG is an image format with lossy compression and is not recommended.

4.7 SignData structures

The signoPAD API components can return a captured signature as a data structure in two different **SignData** formats. Both formats are encrypted, compressed, biometric structures that can be stored in a database or as a tag in a TIFF or PDF document.

The conventional SignData format is encrypted symmetrically and can therefore be unpacked again and visualised by other signotec components. 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 the `RSADecryptSignData()` method. Users of the RSA-encrypted format must have the appropriate private RSA key otherwise the data cannot be decrypted.

A separate API (signoAPI) is available for the (ISO-compliant) signature of PDF and TIFF documents. This API includes a wide range of functions for PDF management along with much more. Please speak to your signotec contact if you are interested.

Conventional SignData structures can be visualised in real time using the **signview.dll** component from the signoAPI. The component is included as a demo version; the full version is included in the signoAPI.

4.8 Notes for redistribution

The signoPAD-API setup may be installed with the 'silent' flag set. For details, please see the 'signoPAD API Installation Guide,' which can be found in the Download area at www.signotec.com.

You can, of course, redistribute individual files from the signoPAD API in a separate Setup. Essentially, only the 'STPad' component used by your application, possibly the STPad.ini and STPadStores.ini files, and, depending on the component used, Microsoft Runtime and/or Framework files, are required to support the signotec Sigma, Zeta, Omega, Gamma, Delta and Alpha LCD signature pads. See also section 'Dependencies'. If your application uses the STPadCapt.ocx component, it must be registered in the system using regsvr32.

The signoPAD API Setup installs files at the following locations:

Component	Installation path
STPadCapt.ocx	%PROGRAMFILES%\signotec\Dll or %PROGRAMFILES(x86)%\signotec\Dll
STPadLib.dll	<Installation path>\signoPAD-API\STPadLib
	<Installation path>\signoPAD-API\Samples\C++\Binary\STPadLib Demo App
STPadLib.h	<Installation path>\signoPAD-API\STPadLib
STPadLib.lib	<Installation path>\signoPAD-API\STPadLib
STPadLibNet.dll	<Installation path>\signoPAD-API\STPadLibNet
	<Installation path>\signoPAD-API\Samples\C#.NET (WPF)\Binary\STPadLibNetDemoApp
	<Installation path>\signoPAD-API\Samples\C#.NET\Binary\STPadLibNetDemoApp
	<Installation path>\signoPAD-API\Samples\VB.NET\Binary\STPadLibNetDemoApp

STPadLibNet.xml	<Installation path>\signoPAD-API\STPadLibNet
STPdfLib18.dll	<Installation path>\signoPAD-API\STPdfLib
	%PROGRAMFILES%\signotec\Dll or %PROGRAMFILES(x86)%\signotec\Dll
	<Installation path>\signoPAD-API\Samples\C#.NET\Binary\STPadLibNetDemoApp
	<Installation path>\signoPAD-API\Samples\VB.NET\Binary\STPadLibNetDemoApp
STPad.ini	%COMMONPROGRAMFILES%\signotec\Config or %COMMONPROGRAMFILES(x86)%\signotec\Config
STPadStores.ini	%COMMONPROGRAMFILES%\signotec\Config or %COMMONPROGRAMFILES(x86)%\signotec\Config
signview.dll	%PROGRAMFILES%\signotec\Dll or %PROGRAMFILES(x86)%\signotec\Dll
Various sample applications including source code	<Installation path>\signoPAD-API\Samples

5 Signing and encryption with signotec LCD signature pads

The signotec LCD Sigma (from firmware 1.16), Zeta, Omega (from firmware 1.25), Gamma, Delta and Alpha signature pads offer various options to encrypt or sign data with RSA keys. The biometric data of a signature captured with a device can be encrypted to prevent this extremely sensitive data from being manipulated. Hash values can be signed to prove the integrity of data. These hash values can be transferred to the signature device in the form of a SHA-1 or SHA-256. Alternatively, however, the device can calculate the SHA-1, SHA-256 or SHA-512 values itself using the displayed screen content.

In the case of advanced signatures using signotec pads, the signer is only a temporary owner of the signature certificate, which is stored in the signature device. While the signer is not the actual owner, at the moment of signing the signer is the holder of the certificate, which satisfies the requirements of the Signature Act. However, the signature cannot be assigned to the signer using the signature certificate; rather, it must be assigned based on the biometric characteristics of the signature. For this reason, this biometric data must be afforded special protection, comparable to the protection of a signature certificate and PIN in the case of a qualified electronic signature.

signotec pads provide this protection by enabling the encryption of this sensitive data with a public RSA key. Only the owner of the related private key is able to decrypt the biometric data. Therefore, it is recommended that this private key be stored in a trustworthy place (e.g., with a notary) and that it is only used when the identity of a signer must be authenticated by a handwriting expert (e.g., before a court).

In addition, the signotec pads offer a procedure that enables the inseparable assignment of biometric data to the signed document or image content. The unsigned document or the image displayed on the signature device can be signed on the signature device in the form of a hash value combined with the hash value of the biometric data. It is not technically possible to sign another document using the identical biometric data in the same signature device.

The following section describes two typical scenarios in which data is encrypted and signed. The sample code is written in C# and uses the STPadLibNet.dll component; the code is similar for other languages and components.

5.1 Signing documents

First, the hash value of the document to be signed must be calculated (hereafter referred to as hash 1). The SHA-1 and SHA-256 procedures are permissible for this purpose. Before signing begins, the hash value must be transferred to the signature device (prior to this, of course, the connection must be established with the pad and the desired image content displayed; these steps are not listed in the sample code):

```
try
{
    stPad.RSASetHash(hash1, HashAlgo.SHA256, HashFlag.None);
    stPad.SignatureStart();
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

Calling `SignatureStart()` starts the process of capturing the signature. Hash 1 is then encrypted and entered in the biometric data; once the signature has been captured, hash 1 can be digitally signed with the hash value of the biometrics (hereafter referred to as hash 2).

Since hash 1 can only be transferred before `SignatureStart()` is called, it is not possible to assign it to a set of biometric data retroactively.

Once the signature has been captured, the capturing process must be completed as usual using `SignatureStop()` or `SignatureConfirm()`.

Hash 1 and hash 2 can now be linked and signed. It is recommended to use the RSASSA-PSS signature scheme; alternatively, a padding can also be used without a hash OID in accordance with RSASSA-PKCS1-V1_5. This step can be skipped. However, in this case, it will only be possible to retrospectively check that hash 1 is correct by decrypting the biometric data.

```
byte[] signature = null;
try
{
    signature = stPad.RSASign(RSAScheme.PSS, HashValue.Combination,
                              SignFlag.None);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

The biometric data encrypted using RSA is then collected:

```
byte[] signData;
try
{
    signData = stPad.RSAGetSignData(SignDataGetFlag.None);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

The signature (`signature`) and the biometric data (`signData`) must then be integrated into the document. While this means the document is signed, it has not yet been provided with an advanced electronic signature. To do this, the entire document must be digitally signed again in the form of a hash value (hereafter referred to as hash 3). If the document is signed according to PKCS#1, the public certificate to verify the signature must be collected from the pad in addition to the signature:

```
byte[] signature = null;
X509Certificate2 cert;
try
{
    stPad.RSASetHash(hash3, HashAlgo.SHA256, HashFlag.None);
    signature = stPad.RSASign(RSAScheme.PKCS1_V1_5, HashValue.Hash1,
                              SignFlag.None);
    cert = (X509Certificate2)stPad.RSASaveSigningCertAsStream
                                                (CertType.Cert_DER);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

Alternatively, the signature can also be created in the PKCS#7 format, which already contains the public certificate. The following method is recommended when signing PDF documents according to the so-called DigSig standard:

```
byte[] signature = null;
try
{
    stPad.RSASetHash(hash3, HashAlgo.SHA256, HashFlag.None);
    signature = stPad.RSASign(RSAScheme.PKCS1_V1_5, HashValue.Hash1,
                              SignFlag.PKCS7 | SignFlag.IncludeChain);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

In the case of a DigSig signature, the signature (`signature`) and, if possible, the public certificate (`cert`) must be integrated into the document in accordance with the standard. In the case of a proprietary signature, this data must be stored in a different manner (e.g., in a database) and assigned to the document. In both cases, this signature is used to check whether the document was changed after signing.

5.2 Signing of image content (content signing)

signotec LCD signature pads can calculate and then sign a hash value using the image content displayed during the signature. To do this, the desired image content must first be transferred to the pad using the conventional methods `DisplaySetImage()`, `DisplaySetText()`, etc.; during this process, it does not matter whether the data is transferred directly to the visible foreground buffer or the invisible background buffer or non-volatile memory. However, it is recommended that you use the so-called DisplayHashBuffer.

Then the `RSACreateDisplayHash()` method must be accessed to calculate a hash value (hereafter referred to as hash 1) in the signature device using the content of the specified source memory. The algorithms SHA-1, SHA-256 or SHA-512 can be used for this purpose. The method returns the image data stream that was used to calculate hash 1 in a proprietary form so that hash 1 can be recalculated at a later stage. This image data stream can be converted into an image, either immediately or at a later stage, using the `RSACreateHashedImage()` method in order to be able to reproduce the image content displayed during the signature.

```
byte[] imageData = null;
Bitmap bitmap = null;
try
{
    imageData = stPad.RSACreateDisplayHash(HashAlgo.SHA256,
                                           DisplayTarget.DisplayHashBuffer);
    bitmap = stPad.RSACreateHashedImage(imageData, Color.Black, 11);
    stPad.SignatureStart();
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

Content signing is started through the calling of `SignatureStart()`. The image content that was previously used to calculate hash 1 is now displayed (if this has not already occurred), and the signature process begins. Outputs on the screen are no longer possible in this state. This allows the signature device to ensure that only the content that was previously used to calculate hash 1 is visible during signing. Hash 1 is then encrypted and entered in the biometric data; once the signature has been captured, hash 1 can be digitally signed with the hash value of the biometrics (hereafter referred to as hash 2).

Once the signature has been captured, the capturing process must be completed as usual using `SignatureStop()` or `SignatureConfirm()`.

First, hash 1 and hash 2 have to be linked and signed. It is recommended to use the RSASSA-PSS signature scheme; alternatively, a padding can also be used without a hash OID in accordance with RSASSA-PKCS1-V1_5. The public certificate to verify the signature must be collected from the pad in addition to the signature:

```
byte[] signature = null;
X509Certificate2 cert;
try
{
    signature = stPad.RSASign(RSAScheme.PSS, HashValue.Combination,
                              SignFlag.None);
    cert = (X509Certificate2)stPad.RSASaveSigningCertAsStream
                                                    (CertType.Cert_DER);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

The biometric data encrypted using RSA is then collected:

```
byte[] signData;
try
{
    signData = stPad.RSAGetSignData(SignDataGetFlag.None);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

The signature (`signature`) and the biometric data (`signData`) must then be stored, for example in a database, along with the image data stream (`imageData`) and the signature certificate (`cert`). This data can be used at a later stage to unambiguously prove that, during the capture of the signature defined by the biometric data, only the screen content defined by the image data stream was visible on the signature device. This is possible by calculating hash 1 and hash 2 based on this data and comparing these values with the signature.

6 Description of possible error messages

Most methods of the STPadCapt.ocx component and the STPadLib.dll return an integer value, which is always negative in the case of an error. A description of the respective error messages is provided in the following table. An error description can be retrieved at runtime by calling the `ControlGetErrorString()` method.

The STPadLibNet.dll throws an `STPadException` if an error occurs, which is why all calls should be enclosed with `try()` & `catch()`. The `Message` and `ErrorCode` properties of `STPadException` contain an error description or one of the error codes listed below.

The error descriptions are available in the components in German, English, French and Italian.

Error	Description
-1	A NULL pointer was passed.
-3	One of the parameters that were passed contains an invalid value.
-4	The signing pad is already being used by another application.
-5	No connection has been opened to this signature pad.
-6	A connection has already been opened.
-8	No device with this ID is connected.
-9	The LED colour that was passed cannot be set.
-12	The function could not be executed because the signature capture process is running.
-13	No further hotspots can be added
-14	The function could not be executed because the coordinates of an active hotspot overlap with the signature window or another active hotspot.
-15	The function could not be executed because no signature capture area has been set.
-17	The function could not be executed because no signature capture process was started.
-18	An error occurred while attempting to reserve memory.
-19	An error occurred while initialising a system resource.
-20	An error occurred while communicating with the signing pad.
-21	The rectangle that was passed is invalid.
-22	No compatible devices connected or the connection to a device has been cut.
-25	The connected device does not support this function or one of the parameters.
-26	Error while reading or writing a file.
-27	An error occurred when saving the key or certificate.
-28	The required key and/or certificate or CSR is not available.
-30	An internal error occurred in the signing pad during calculation.
-31	The password is invalid because it is too short or the data to be signed are too large for the key stored in the signing pad.
-32	Data was lost when transferring signature data from the signature device.
-33	The function could not be executed because the device is in 'Content signing' mode.
-34	The function could not be executed because the key is password protected and no password was entered or the password that was entered was incorrect
-36	The option could not be changed because it is password protected and no password was entered, or the password that was entered was incorrect.
-39	Signature capture could not be started because pen-controlled scrolling is activated.
-40	The function could not be executed because the device does not feature an NFC reader.

-41	The function could not be executed due to an unknown error. It may be necessary to update the software.
-42	The signature key could not be accessed because it is password protected and an incorrect password has been entered too often. Please renew the signature key.
-43	The function could not be executed because no signing password has been assigned yet. Please assign a signing password first.
-44	The function could not be executed because a certificate chain could not be built to a trusted root certification authority.
-45	Signature capture could not be started because a keypad hotspot is defined.
-46	The certificate or one of the certificates in the certificate chain has expired or is not yet valid.
-47	Trust for the certificate or one of the certificates in the certificate chain has been revoked.
-48	The certificate or one of the certificates in the certificate chain does not have a valid signature.
-81	The document could not be loaded because it is password protected and no password was entered, or the password that was entered was incorrect.
-82	The function could not be executed because neither the STPdfLib18.dll nor the STPdfLib16.dll could be loaded.
-83	The function cannot be executed because no document has been loaded.
-84	The function could not be executed because the secure sign mode is active.
-85	The data does not contain biometric data.
-86	The data could not be decrypted.
-87	The public key of the transferred certificate does not match the key stored in the pad.
-88	The certificate could not be decoded.
-89	The password that was passed is invalid.
-90	The function could not be executed because no Hash 1 has been passed.
-91	No Hash 2 has been computed on the biometric data because no hash algorithm has been set.
-92	The function could not be executed because the ‘Content signing’ mode has already been left.
-93	The function could not be executed because an overlay rectangle is set.
-94	The function could not be executed because the screen content is scrolled or pen-controlled scrolling is activated.
-95	The function could not be executed because it would have activated the scroll mode that is not possible if a hotspot outside the overlay rectangle is defined.
-97	An error occurred during initialisation. Please restart the software.
-98	The minimum password length could not be set because the connected device does not support this function or an invalid value was set for the property RSASignPasswordLength.
-99	The function could not be executed because keypad hotspots can only be inverted if the foreground memory is set as active device memory.

7 Information about the available image memory

The signotec LCD Signature Pads have several image memories, which can be used by different methods. An image memory has at least the size of the display and can store one picture in a maximum of this size. Adding another image overrides the areas it overlaps with the existing memory content. Adding multiple images to one memory can therefore create a collage.

Depending on the model, a different number of volatile and non-volatile memories are available.

7.1 Volatile image memory

All signotec LCD Signature Pads have at least two volatile image memories, one foreground memory containing the current display content and one background memory, which can be used to prepare the display content. It can be written in both of the memories.

The content of the volatile image memory is lost when you close the connection to the device.

7.1.1 Model type Sigma

The two volatile image memories have the size of the display (320 x 160 pixels).

The transmission and representation of images is usually so fast that there is no visible lag. For more complex representations that consist of several individual images, it may be useful to first save them in the background memory before copying them into the foreground memory.

7.1.2 Model type Zeta

The two volatile image memories have the size of the display (320 x 200 pixels).

The transmission and representation of images is usually so fast that there is no visible lag. For more complex representations that consist of several individual images, it may be useful to first save them in the background memory before copying them into the foreground memory.

7.1.3 Model type Omega

The Omega model has three volatile image memories, two that have the doubled size of the display (640 x 960 pixels) to be used as foreground and background memory and one that has the size of the display (640 x 480 pixels) to be used as overlay memory. Its contents can be overlaid over the current display content.

The speed of displaying an image in the Omega model with firmware up to Version 1.40 depends on the size and content of the images. The image composition is usually visible. Therefore, images should always be stored first in the background memory and then moved into the foreground memory.

An image is displayed in the Omega model with firmware 2.0 or newer only after it has been transferred; the image composition is not visible. The speed of the image transmission depends on the size and content of the images and the connection type. If possible, the device should always be operated in WinUSB mode. To this end, it is necessary to separately install the signotec WinUSB driver (this is a component of the signoPAD-API). For more

complex representations that consist of several individual images, it can generally be useful to first save them in the background memory before copying them into the foreground memory.

7.1.4 Model type Gamma

The Gamma model has three volatile image memories, two that are larger than the display (800 x 1440 pixels) to be used as foreground and background memory and one that has the size of the display (800 x 480 pixels) to be used as overlay memory. Its contents can be overlaid over the current display content.

With the Gamma model, an image is only displayed after it has been transferred; the image composition is not visible. The speed of the image transmission depends on the size and content of the images and the connection type. If possible, the device should always be operated in WinUSB mode. To this end, it is necessary to separately install the signotec WinUSB driver (this is a component of the signoPAD-API). For more complex representations that consist of several individual images, it can generally be useful to first save them in the background memory before copying them into the foreground memory.

7.1.5 Model type Delta

The Delta model has three volatile image memories, two that are larger than the display (1280 x 37,600 pixels) to be used as foreground and background memory one that is the size of the display (1280 x 800 pixels) to be used as overlay memory. Its contents can be overlaid over the current display content.

The speed of displaying a picture in the Delta model depends on the size and content of the images as well as the connection type. The image composition is usually visible. Therefore, images should always be stored first in the background memory and then moved into the foreground memory. If possible, the device should always be operated in WinUSB mode. To this end, it is necessary to separately install the signotec WinUSB driver (this is a component of the signoPAD-API).

7.1.6 Model type Alpha

The Alpha model has three volatile image memories, two that are larger than the display (2048 x 2048 pixels) to be used as foreground and background memory one that is the size of the display (768 x 1366 pixels) to be used as overlay memory. Its contents can be overlaid over the current display content.

With the Alpha model, an image is only displayed after it has been transferred; the image composition is not visible. The speed of the image transmission depends on the size and content of the images and the connection type. If possible, the device should always be operated in WinUSB mode. To this end, it is necessary to separately install the signotec WinUSB driver (this is a component of the signoPAD-API). For more complex representations that consist of several individual images, it can generally be useful to first save them in the background memory before copying them into the foreground memory.

7.2 Non-volatile image memory

Depending on the model, a different number of non-volatile image memories are available. The saving of images in non-volatile image memory lasts longer than storing in volatile image memory, but the content remains unchanged even after switching off the device. An intelligent memory management detects whether an image to be stored is already stored in the device so that only the first time it’s stored it comes to a delay.

7.2.1 Model type Sigma

The Sigma model has one non-volatile image memory in the size of the display (320 x 160 pixels), which can only be used for the standby image. Due to the rapid transmission and display of pictures, it is not necessary to be able to save other images permanently.

7.2.2 Model type Zeta

The Zeta model has one non-volatile image memory in the size of the display (320 x 200 pixels), which can only be used for the standby image. Due to the rapid transmission and display of pictures, it is not necessary to be able to save other images permanently.

7.2.3 Model type Omega

The Omega model has eleven non-volatile image memories, which can be used for the standby image, the slide show and optimizations of the program. The memories, used for the standby image or the slide show, are read-only and can be freed only by disabling the standby image or the slide show.

One non-volatile image memory has the doubled size of the display (640 x 960 pixels), ten memories have the size of the display (640 x 480 pixels).

To use a non-volatile memory, this must be reserved first. This is done by calling the `DisplaySetTarget()` method. The size of the currently selected memory can be queried using the `DisplayTargetWidth` and `DisplayTargetHeight` properties.

The WinUSB mode which is available from Firmware 2.0 does not require the use of non-volatile memory to optimise the program, as image transmission is very fast. However, it depends on the individual case at hand and the developer should make the final decision.

7.2.4 Model type Gamma

The Gamma model has ten non-volatile image memories, which can be used for the standby image, the slide show and optimisations of the program. The memories, used for the standby image or the slide show, are read-only and can be freed only by disabling the standby image or the slide show.

The ten non-volatile memories are the same size as the display (800 x 480 pixels).

To use a non-volatile memory, this must be reserved first. This is done by calling the `DisplaySetTarget()` method. The size of the currently selected memory can be queried using the `DisplayTargetWidth` and `DisplayTargetHeight` properties.

The WinUSB mode does not require the use of non-volatile memory to optimise the program, as image transmission is very fast. However, it depends on the individual case at hand and the developer should make the final decision.

7.2.5 Model type Delta

The Delta model has 32 non-volatile image memories, which can be used for the standby image, the slide show and optimisations of the program. The memories, used for the standby image or the slide show, are read-only and can be freed only by disabling the standby image or the slide show.

The 32 non-volatile memories are the same size as the display (1280 x 800 pixels).

To use a non-volatile memory, this must be reserved first. This is done by calling the `DisplaySetTarget()` method. The size of the currently selected memory can be queried using the `DisplayTargetWidth` and `DisplayTargetHeight` properties.

The WinUSB mode does not require the use of non-volatile memory to optimise the program, as image transmission is very fast. However, it depends on the individual case at hand and the developer should make the final decision.

7.2.6 Model type Alpha

The Alpha model has ten non-volatile image memories, which can be used for the standby image, the slide show and optimisations of the program. The memories, used for the standby image or the slide show, are read-only and can be freed only by disabling the standby image or the slide show.

The ten non-volatile memories are the same size as the volatile memories (2048 x 2048 pixels).

To use a non-volatile memory, this must be reserved first. This is done by calling the `DisplaySetTarget()` method. The size of the currently selected memory can be queried using the `DisplayTargetWidth` and `DisplayTargetHeight` properties.

The WinUSB mode does not require the use of non-volatile memory to optimise the program, as image transmission is very fast. However, it depends on the individual case at hand and the developer should make the final decision.

7.3 Copying between image memories

The contents can be copied between the most of the available image stores. The content of background memory cannot be copied to the foreground memory; it can only be moved. The contents of the overlay memory cannot be copied but only overlaid over the display content.

Typical copy operations are copying from a non-volatile image memory in a volatile image memory and moving from the volatile background memory into the foreground memory. Copying an image within the device is always faster than sending this image from the PC to the device. Please refer to the descriptions of the `DisplaySetImageFromStore()` and `DisplaySetOverlayRect()` methods for details.

7.4 The typical process

Most applications use the same images with possibly variable units (such as document-related texts) for the signature process. It therefore makes sense to store images that are the same each time in one of the non-volatile memory if possible. The following is the typical work flow for this scenario (C# sample code and use of the component STPadLibNet.dll; code is similar for other languages and components):

First, the images are loaded, which will be permanently stored in the device, since they change rarely. A memory is reserved by calling the `DisplaySetTarget()` method with the value -1. The return value of the method is the ID of the memory used. If no non-volatile image memory is available, the background buffer is set as an image buffer (ID = 1). This is always the case when using the Sigma or Zeta models. When using the Omega, Gamma and Alpha models, the number of available memories can be less than expected when a slide show is configured.

Text and images that are added to a non-volatile memory are only saved locally to begin with and are sent to the device only when `DisplaySetImageFromStore()` or `DisplayConfigSlideShow()` is called in order to be able to compare the image (which may be composed of several texts and images) with the image already stored in the device. Thus only when one of these methods is called, there will be a noticeable delay.

```
DisplayTarget targetStore;
try
{
    targetStore = stPad.axSTPadCapt1.DisplaySetTarget
                                        (DisplayTarget.NewStandardStore);
    stPad.DisplaySetImageFromFile(10, 10, @"C:\1.bmp");
    stPad.DisplaySetText(200, 160, TextAlignment.Left, "Signature:");
    stPad.DisplaySetImageFromFile(220, 400, @"C:\2.bmp");
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

The content can now be copied to a volatile image memory, typically the background memory (`DisplaySetTarget(DisplayTarget.BackgroundBuffer)`). If the images have already been written to the background memory because no non-volatile memory was available (see above), the `DisplaySetImageFromStore()` method will not function, however it will also not produce any errors and can therefore be safely called.

```
try
{
    stPad.DisplaySetTarget(DisplayTarget.BackgroundBuffer);
    stPad.DisplaySetImageFromStore(targetStore);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

Now content, that change with every signature process, can be added to the background memory.

```
try
{
    stPad.DisplaySetImageFromFile(120, 400, @"C:\3.bmp");
    stPad.DisplaySetText(200, 160, TextAlignment.Left, "01.01.2010");
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

In the background memory there is now a collage of two images and a text copied from a non-volatile memory and an image and a text that have been sent from the PC. This collage can now be moved into the foreground memory and thus displayed on the screen. The total composition has happened before in the background buffer and thus "invisible".

```
try
{
      stPad.DisplaySetTarget(DisplayTarget.ForegroundBuffer);
      stPad.DisplaySetImageFromStore(DisplayTarget.BackgroundBuffer);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

The process described must be performed every time a connection is opened. When a connection is closed all information about reserved memories is lost. Only information regarding which display content is stored in which non-volatile memory remains saved on the device (even when it is switched off).

7.5 The standby feature

The signotec LCD signature pads (Omega, Gamma, Delta and Alpha models only) can display one or more images automatically when not in use (no established connection). These images are stored permanently in the device and they are displayed without launching any application on the PC.

Image memories that are used by the standby feature are write protected and cannot be used for by an application.

7.5.1 Displaying a logo

In all devices, an image that is displayed automatically in standby can be stored permanently. Please refer to the descriptions of the `DisplaySetStandbyImage()` and `DisplaySetStandbyImageFromFile()` methods for details.

7.5.2 Displaying a slide show

As an alternative to a logo, the Omega, Gamma, Delta and Alpha models can display a slide show containing up to ten (Gamma and Alpha), 11 (Omega) or 32 (Delta) images. To configure a slide show, please follow these steps:

First, a standby mode that may be configured must be disabled by calling `DisplayConfigSlideShow()` in order to remove write protection from all images. The current configuration can be retrieved with the `DisplayGetStandbyId()` method.

Then any contents can be written to one or more of the non-volatile image memories (as described in 7.4). When all the contents have been written, the desired image memories must be configured using the `DisplayConfigSlideShow()` or `DisplayConfigSlideShowEx()` method.

7.6 Exclusive use of non-volatile memory

Since after closing the connection to a device all the information about reserved stores is lost, two applications may use the same memory and thus always overwrite it. As a result, the use of non-volatile memories can even cause slower program execution.

To avoid this problem, a user may exclusively reserve up to ten image memories of an Omega device for a particular application. These memories are not available to other applications and thus cannot be overwritten accidently.

7.6.1 Implementation in an application

Applications must provide the component with their name in order to support this functionality. This is done with the `ControlAppName` property. Users can enter this name in the configuration file and reserve a certain number of memories for this application.

7.6.2 Assign non-volatile memory to an application

Memories are assigned in the STPadStores.ini file, which must be located in the '%COMMONPROGRAMFILES%\signotec\config' folder, or in the case of 64-bit Windows, in the '%COMMONPROGRAMFILES(x86)%\signtotec\config' folder. The configuration in this file only applies to this workstation; if the pad is connected to another PC without configuration, all memories will be available there again.

The file can contain up to ten sections (for ten applications). The names of the sections are `[Application1]`, `[Application2]`, etc.

Each section contains two keys named `Name` and `StoreCount`. The `Name` key contains the name given by the application (see above). The `StoreCount` key contains the number of memories to be reserved. It does not matter if memory is reserved for one or several applications, but the sum of the reserved memories must not exceed 10. The Omega memory with a size of 640 x 960 pixels cannot be used exclusively.

```
[Application1]
Name=MyGreatApp
StoreCount=2

[Application2]
Name=Another Great App
StoreCount=4
```

If not all of the available memory is assigned exclusively, all further image memories are available for all applications. If a standby image or a slide show has already been configured for the device, the maximum number of available memories is reduced accordingly. If a standby image or slide show is configured by an application on a workstation with a memory configuration, only memories can be used in this context that have been reserved for the respective application or have not been reserved at all.

8 Methods

Methods are named according to the following naming convention:

- Methods that set or query general hardware properties begin with '**Device**'
- Methods that set or query sensor properties begin with '**Sensor**'
- Methods that apply to the signature begin with '**Signature**'
- Methods that set or query LCD properties begin with '**Display**'
- Methods that set or query component properties begin with '**Control**'
- Methods that are connected to the RSA functionality of the device begin with **'RSA'.**
- Methods that are connected to the loading of PDF documents begin with **'PDF'**

The methods of the STPadLib.dll component all begin with 'ST,' for example, 'STDeviceOpen()'.

Some methods and parameters are not contained in all components, so you should always take note of the given method declaration. The methods are declared by the STPadCapt.ocx component in MIDL syntax, by the STPadLib.dll component in C syntax and by the STPadLibNet.dll component in C# and VB syntax.

8.1 DeviceSetComPort method

This method defines the interfaces to be searched for devices when `DeviceGetCount()` is called up. A search will only be made for USB devices unless this method is called up.

In a Terminal Server environment, it is possible to pass through devices via a virtual channel from the client to the server. The API independently detects whether such a channel is configured. The channel appears as COM1 or COM2 for the application. This means that existing applications that communicate with a signature device via the serial interface do not need to be adjusted for operation via a virtual channel. For details of the signotec Virtual Channel, please get in touch with your contact at signotec.

In the STPad.ini configuration file, you can specify that the search will initially be made via the virtual channel provided at least one COM port has been specified. For this purpose, the `ForceVC` key must be set to `YES`.

In order to query the connection to which an existing device is connected please use the`DeviceGetComPort()` and `DeviceGetIPAddress()`methods.

Parameter	Values	I/O	Description

`BSTR bstrPortList`
`LPCWSTR szPortList`
`string portList`
`ByVal portList As String`
	A list separated by semi-colons with at least one of the following members:		

	`"HID"`	I	A search will be made for HID and WinUSB devices.
	`"IP=<Address>:<Port>"`	I	An Alpha or Delta with an Ethernet connection or an USB device that is connected to a signotec Ethernet USB adapter will be searched for at the specified IP address; `<Address>` is the IP address of the device and has the format `x.x.x.x` or the host name, `<Port>` is the port via which the signature device should be activated, generally `1002`. A valid value would therefore be, for example, `'IP=192.168.100.100:1002'` or `'IP=host1:1002'`; for details about the signotec Ethernet USB adapter, please refer to your contact at signotec.
	`"all"`	I	A search will be made for devices on all COM ports and/or virtual channels; this search may take an extremely long time depending on the hardware configuration.
	Whole number	I	A search will be made for a serial device on the COM port and/or virtual channel with the specified number (>=0); this search may take an extremely long time if no signotec device is connected to the port.
	`"LowSpeed"`	I	A search will be made on the COM ports only for devices that communicate at a baud rate of 115200 (Sigma, Zeta, Omega, Gamma plus Delta and Alpha in 'Low Speed' mode). The search is accelerated if COM ports to which no signotec LCD signature pads are connected are also searched; this does not influence the search on virtual channels or for HID, WinUSB or IP devices
	`"High Speed"`	I	A search will be made on the COM ports only for devices that communicate at a baud rate of 2 Mbaud (Delta and Alpha in 'High Speed' mode), which accelerates the search; this does not influence the search on virtual channels or for HID, WinUSB or IP devices
Return value	**Values**	**Description**	

`LONG`
`int`
`Integer`
	`>= 0`	Number of detected members in the transferred list (for the member `"all"` the return value 256 will be added to old versions instead of 1 for compatibility reasons)	

	`< 0`	Error (not for STPadLibNet.dll)	

8.1.1 STPadCapt.ocx

Available from Version 8.0.6. The status described is available from Version 8.1.2.

```
LONG DeviceSetComPort(BSTR bstrPortList)
```

8.1.1.1 Implementation in C#

```
int nPortCount =
        axSTPadCapt1.DeviceSetComPort("HID;1;4;IP=192.168.100.100:1002");
if (nPortCount < 0)
    MessageBox.Show(String.Format("Error {0}", nPortCount);
else
    MessageBox.Show(String.Format("{0} ports set.", nPortCount);
```

8.1.1.2 Implementation in Visual Basic

```
Dim nPortCount As Integer =
         AxSTPadCapt1.DeviceSetComPort("HID;1;4;IP=192.168.100.100:1002")
If nPortCount < 0 Then
    MsgBox("Error " & CStr(nPortCount))
Else
    MsgBox(CStr(nPortCount) & " ports set.")
End If
```

8.1.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.1.2.

```
LONG STDeviceSetComPort(LPCWSTR szPortList)
```

8.1.2.1 Implementation in C++

```
LONG nPortCount = STDeviceSetComPort(L"HID;1;4;IP=192.168.100.100:1002");
if (nPortCount < 0)
    wprintf(L"Error %d", nPortCount);
else
    wprintf(L"%d ports set", nPortCount);
```

8.1.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.1.2.

```
int DeviceSetComPort(string portList)
```

```
Function DeviceSetComPort(ByVal portList As String) As Integer
```

8.1.3.1 Implementation in C#

```
try
{
    int nPortCount =
                stPad.DeviceSetComPort("HID;1;4;IP=192.168.100.100:1002");
    MessageBox.Show(String.Format("{0} ports set.", nPortCount));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.1.3.2 Implementation in Visual Basic

```
Try
    Dim nPortCount As Integer =
                STPad.DeviceSetComPort("HID;1;4;IP=192.168.100.100:1002")
    MsgBox(CStr(nPortCount) & " ports set")
Catch ex As STPadException
    MsgBox(ex.Message)

End Try
```

8.2 DeviceGetConnectionType method

This method returns the type of connection via which a device is connected.

Parameter	Values	I/O	Description
LONG nIndex int index ByVal index As Integer	>= 0	I	Index of the device whose connection type is to be queried

Return value	Values	Description
LONG int Integer	0	HID
	1	WinUSB
	2	Serial or virtual channel
	3	Ethernet
	< 0	Error (not for STPadLibNet.dll)

8.2.1 STPadCapt.ocx

Available from Version 8.1.2.

```
LONG DeviceGetConnectionType(LONG nIndex)
```

8.2.1.1 Implementation in C#

```
int nType = axSTPadCapt1.DeviceGetConnectionType(0);
switch (nType)
{
    case 0:
        MessageBox.Show("The device is connected via HID.");
        break;
    case 1:
        MessageBox.Show("The device is connected via WinUSB.");
        break;
    case 2:
        MessageBox.Show("The device is connected to a serial port.");
        break;
    case 3:
        MessageBox.Show("The device is connected via IP.");
        break;
    default:
        MessageBox.Show(String.Format("Error {0}", nType);
        break;
}
```

8.2.1.2 Implementation in Visual Basic

```
Dim nType As Integer = AxSTPadCapt1.DeviceGetConnectionType(0)
Select Case nType
    Case 0
        MsgBox("The device is connected via HID.")
    Case 1
        MsgBox("The device is connected via WinUSB.")
    Case 2
        MsgBox("The device is connected to a serial port.")
    Case 3
        MsgBox("The device is connected via IP.")
    Case Else
        MsgBox("Error " & CStr(nType))
End Select
```

8.2.2 STPadLib.dll

Available from Version 8.1.2.

```
LONG STDeviceGetConnectionType(LONG nIndex)
```

8.2.2.1 Implementation in C++

```
LONG nType = STDeviceGetConnectionType(0);
switch (nType)
{
    case 0:
        wprintf(L"The device is connected via HID.");
        break;
    case 1:
        wprintf(L"The device is connected via WinUSB.");
        break;
    case 2:
        wprintf(L"The device is connected to a serial port.");
        break;
    case 3:
        wprintf(L" The device is connected via IP.");
        break;
    default:
        wprintf(L"Error %d", nType);
        break;
}
```

8.2.3 STPadLibNet.dll

Available from Version 8.1.2.

```
int DeviceGetConnectionType(int index)
```

```
Function DeviceGetConnectionType(ByVal nIndex As Integer) As Integer
```

8.2.3.1 Implementation in C#

```
try
{
    int nType = stPad.DeviceGetConnectionType(0);
    switch (nType)
    {
        case 0:
            MessageBox.Show("The device is connected via HID.");
            break;
        case 1:
            MessageBox.Show("The device is connected via WinUSB.");
            break;
        case 2:
            MessageBox.Show("The device is connected to a serial port.");
            break;
        case 3:
            MessageBox.Show("The device is connected via IP.");
            break;
        default:
            MessageBox.Show(String.Format("Unknown connection type: {0}",
                            nType));
            break;
    }
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.2.3.2 Implementation in Visual Basic

```
Try
   Dim nType As Integer = STPad.DeviceGetConnectionType(0)
    Select Case nType
        Case 0
            MsgBox("The device is connected via HID.")
        Case 1
            MsgBox("The device is connected via WinUSB.")
        Case 2
            MsgBox("The device is connected to a serial port.")
        Case 3
            MsgBox("The device is connected via IP.")
        Case Else
            MsgBox("Unknown connection type: " & CStr(nType))
    End Select
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.3 DeviceGetComPort method

This method returns the number of the COM port or virtual channel to which a device is connected.

Parameter	Values	I/O	Description

`LONG nIndex`
`int index`
`ByVal index As Integer`
	`>= 0`	I	Index of the device whose port number is to be queried

Return value	**Values**	**Description**	

`LONG`
`int`
`Integer`
	`>= 0`	Number of the COM port for serial devices. (Note: Devices that are not serially connected return 0; use `DeviceGetConnectionType()` to ascertain the type of connection)	

	`< 0`	Error (not for STPadLibNet.dll)	

8.3.1 STPadCapt.ocx

Available from Version 8.0.8.

```
LONG DeviceGetComPort(LONG nIndex)
```

8.3.1.1 Implementation in C#

```
int nPort = axSTPadCapt1.DeviceGetComPort(0);
if (nPort < 0)
    MessageBox.Show(String.Format("Error {0}", nPort));
else
    MessageBox.Show(String.Format("This device is connected to COM port
                                   {0}.", nPort));
```

8.3.1.2 Implementation in Visual Basic

```
Dim nPort As Integer = AxSTPadCapt1.DeviceGetComPort(0)
If nPort < 0 Then
    MsgBox("Error " & CStr(nPort))
Else
    MsgBox("This device is connected to COM port " & CStr(nPort))
End If
```

8.3.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDeviceGetComPort(LONG nIndex)
```

8.3.2.1 Implementation in C++

```
LONG nPort = STDeviceGetComPort(0);
if (nPort < 0)
    wprintf(L"Error %d", nPort);
else
    wprintf(L"This device is connected to COM port %d", nPort);
```

8.3.3 STPadLibNet.dll

Available from Version 8.0.19.

```
int DeviceGetComPort(int index)
Function DeviceGetComPort(ByVal nIndex As Integer) As Integer
```

8.3.3.1 Implementation in C#

```
try
{
    int nPort = stPad.DeviceGetComPort(0);
    MessageBox.Show(String.Format(
                    "This device is connected to COM port {0}.", nPort));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.3.3.2 Implementation in Visual Basic

```
Try
    Dim nPort As Integer = STPad.DeviceGetComPort(0)
    MsgBox("This device is connected to COM port " & CStr(nPort))
Catch ex As STPadException
    MsgBox(ex.Message)

End Try
```

8.4 DeviceGetIPAddress method

You can use this method to retrieve the IP address of a Delta or Alpha with Ethernet connection or a signotec Ethernet USB adapter to which a device is connected.

Parameter	Values	I/O	Description
`BSTR* pbstrAddress` `WCHAR szAddress[32]`	`""`	O	The device is not connected via IP
	max. 32 characters	O	Device IP address and port (`"x.x.x.x:x"`)

`LONG nIndex`
`int index`
`ByVal index As Integer`
	`>= 0`	I	Index of the device whose information is to be queried

Return value	**Values**	**Description**	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	
`string` `String`	`""`	The device is not connected via IP	
	max. 32 characters	Device IP address and port (`"x.x.x.x:x"`)	

8.4.1 STPadCapt.ocx

Available from Version 8.0.25.

```
LONG DeviceGetIPAddress(BSTR* pbstrAddress, LONG nIndex)
```

8.4.1.1 Implementation in C#

```
string strAddress = "";
int nResult = axSTPadCapt1.DeviceGetIPAddress(ref strAddress, 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
else
    MessageBox.Show(String.Format("The device is connected to: {0}",
                                  strAddress);
```

8.4.1.2 Implementation in Visual Basic

```
Dim strAddress As String = ""
Dim nResult As Integer = AxSTPadCapt1.DeviceGetIPAddress(strAddress, 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
Else
    MsgBox("The device is connected to: " & strAddress)
End If
```

8.4.2 STPadLib.dll

Available from Version 8.0.25.

```
LONG STDeviceGetIPAddress(WCHAR szAddress[32], LONG nIndex)
```

8.4.2.1 Implementation in C++

```
WCHAR szAddress[32];
LONG nResult = STDeviceGetIPAddress(szAddress, 0);
if (nResult < 0)
    wprintf(L"Error %d", nResult);
else
    wprintf(L"The device is connected to: %s", szAddress);
```

8.4.3 STPadLibNet.dll

Available from Version 8.0.25.

```
string DeviceGetIPAddress(int index)
```

```
Function DeviceGetIPAddress(ByVal index As Integer) As String
```

8.4.3.1 Implementation in C#

```
try
{
    MessageBox.Show(String.Format("The device is connected to: {0}",
                                  stPad.DeviceGetIPAddress(0)));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.4.3.2 Implementation in Visual Basic

```
Try
    MsgBox("The device is connected to: " & STPad.DeviceGetIPAddress(0))
Catch ex As STPadException
    MsgBox(ex.Message)

End Try
```

8.5 DeviceGetCount method

This method searches for connected devices, generates an internal index beginning with 0 and returns the number of devices detected. This value should be cached so that the method only needs to be called, if the number of connected devices has changed. A device's index is retained until the method is called again. The index can be assigned to a device via the information returned by `DeviceGetInfo()`.

By default, a search will only be made for USB devices that are locally connected. A search will only be made for other devices if this has been configured previously by calling up `DeviceSetComPort()`.

Please observe the relevant information in the 'Using multiple instances' section.

Parameter	Values	I/O	Description
-	-	-	-
Return value	**Values**	**Description**	
LONG int Integer	>= 0	Number of devices detected	
	< 0	Error (not for STPadLibNet.dll)	

8.5.1 STPadCapt.ocx

Available from Version 8.0.1.

```
LONG DeviceGetCount()
```

8.5.1.1 Implementation in C#

```
int nDeviceCount = axSTPadCapt1.DeviceGetCount();
if (nDeviceCount < 0)
    MessageBox.Show(String.Format("Error {0}", nDeviceCount);
else
    MessageBox.Show(String.Format("{0} devices detected.", nDeviceCount);
```

8.5.1.2 Implementation in Visual Basic

```
Dim nDeviceCount As Integer = AxSTPadCapt1.DeviceGetCount
If nDeviceCount < 0 Then
    MsgBox("Error " & CStr(nDeviceCount))
Else
    MsgBox(CStr(nDeviceCount) & " devices detected.")
End If
```

8.5.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDeviceGetCount()
```

8.5.2.1 Implementation in C++

```
LONG nDeviceCount = STDeviceGetCount();
if (nDeviceCount < 0)
    wprintf(L"Error %d", nDeviceCount);
else
    wprintf(L"%d devices detected.", nDeviceCount);
```

8.5.3 STPadLibNet.dll

Available from Version 8.0.19.

```
int DeviceGetCount()
```

```
Function DeviceGetCount() As Integer
```

8.5.3.1 Implementation in C#

```
try
{
    int nDeviceCount = stPad.DeviceGetCount();
    MessageBox.Show(String.Format("{0} devices detected.",
                                  nDeviceCount));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.5.3.2 Implementation in Visual Basic

```
Try
    Dim nPort As Integer = STPad.DeviceGetCount()
    MsgBox(CStr(nDeviceCount) & " devices detected.")
Catch ex As STPadException
    MsgBox(ex.Message)

End Try
```

8.6 DeviceGetInfo method

You can use this method to retrieve the serial number and model type of a connected device in order to uniquely identify it.

Parameter	Values	I/O	Description

`BSTR* pbstrSerial`
`WCHAR szSerial[16]`
`out string serial`
`ByRef serial As String`
	max. 16 chars	O	Serial number

LONG* pnType out int type ByRef type As Integer	1	O	'Sigma USB' model type
	2	O	'Sigma serial' model type
	5	O	'Zeta USB' model type
	6	O	'Zeta serial' model type
	11	O	'Omega USB' model type
	12	O	'Omega serial' model type
	15	O	'Gamma USB' model type
	16	O	'Gamma serial' model type
	21	O	'Delta USB' model type
	22	O	'Delta serial' model type
	23	O	'Delta IP' model type
	31	O	'Alpha USB' model type
	32	O	'Alpha serial' model type
	33	O	'Alpha IP' model type
	other	O	Reserved for further model types
LONG nIndex int index ByVal index As Integer	>= 0	I	Index of the device whose information is to be queried
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.6.1 STPadCapt.ocx

Available from Version 8.0.1. The status described is available from Version 8.3.1.

```
LONG DeviceGetInfo(BSTR* pbstrSerial, LONG* pnType, LONG nIndex)
```

8.6.1.1 Implementation in C#

```
string strSerial = "";
int nType = 0;
int nResult = axSTPadCapt1.DeviceGetInfo(ref strSerial, ref nType, 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
else
    MessageBox.Show(String.Format("Type: {0}, Serial: {1}", nType,
                                  strSerial);
```

8.6.1.2 Implementation in Visual Basic

```
Dim strSerial As String = ""
Dim nType As Integer = 0
Dim nResult As Integer = AxSTPadCapt1.DeviceGetInfo(strSerial, nType, 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
Else
    MsgBox("Type: " & CStr(nType) & ", Serial: " & strSerial)
End If
```

8.6.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.3.1.

```
LONG STDeviceGetInfo(WCHAR szSerial[16], LONG* pnType, LONG nIndex)
```

8.6.2.1 Implementation in C++

```
WCHAR szSerial[16];
LONG nType = 0;
LONG nRc = STDeviceGetInfo(szSerial, &nType, 0);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
else
    wprintf(L"Type: %d, Serial: %s", nType, szSerial);
```

8.6.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.3.1.

```
void DeviceGetInfo(out string serial, out int type, int index)
```

```
Sub DeviceGetInfo(ByRef serial As String, ByRef type As Integer, ByVal index As
Integer)
```

8.6.3.1 Implementation in C#

```
try
{
    string strSerial = "";
    int nType = 0;
    stPad.DeviceGetInfo(out strSerial, out nType, 0);
    MessageBox.Show(String.Format("Type: {0}, Serial: {1}", nType,
                                  strSerial));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.6.3.2 Implementation in Visual Basic

```
Try
    Dim strSerial As String = ""
    Dim nType As Integer = 0
    STPad.DeviceGetInfo(strSerial, nType, 0)
    MsgBox("Type: " & CStr(nType) & ", Serial: " & strSerial)
Catch ex As STPadException
    MsgBox(ex.Message)

End Try
```

8.7 DeviceGetVersion method

You can use this method to retrieve the version number of a connected device's firmware. It is intended primarily for support purposes.

Parameter	Values	I/O	Description
`BSTR* pbstrVersion` `WCHAR szVersion[16]`	max. 16 chars	O	Firmware version number (major.minor)

LONG nIndex int index ByVal index As Integer	>= 0	I	Index of the device whose information is to be queried
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	
string String	max. 16 chars	Firmware version number (major.minor)	

8.7.1 STPadCapt.ocx

Available from Version 8.0.3.

```
LONG DeviceGetInfo(BSTR* pbstrVersion, LONG nIndex)
```

8.7.1.1 Implementation in C#

```
string strVersion = "";
int nResult = axSTPadCapt1.DeviceGetVersion(ref strVersion, 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
else
    MessageBox.Show(String.Format("Firmware: {0}", strVersion);
```

8.7.1.2 Implementation in Visual Basic

```
Dim strVersion As String = ""
Dim nResult As Integer = AxSTPadCapt1.DeviceGetVersion(strVersion, 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
Else
    MsgBox("Firmware: " & strVersion)
End If
```

8.7.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDeviceGetVersion(WCHAR szVersion[16], LONG nIndex)
```

8.7.2.1 Implementation in C++

```
WCHAR szVersion[16];
LONG nRc = STDeviceGetVersion(szVersion, 0);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
else
    wprintf(L"Firmware: %s", szVersion);
```

8.7.3 STPadLibNet.dll

Available from Version 8.0.19.

```
string DeviceGetVersion(int index)
Function DeviceGetVersion(ByVal index As Integer) As String
```

8.7.3.1 Implementation in C#

```
try
{
    MessageBox.Show(String.Format("Firmware: {0}",
                                  stPad.DeviceGetVersion(0)));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.7.3.2 Implementation in Visual Basic

```
Try
    MsgBox("Firmware: " & STPad.DeviceGetVersion(0))
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.8 DeviceGetCapabilities method

You can use this method to retrieve various properties of a connected device.

Parameter	Values	I/O	Description

`LONG nIndex`
`int index`
`ByVal index As Integer`
	`>= 0`	I	Index of the device whose information is to be queried

Return value	**Values**	**Description**	
`LONG`	`0`	Bitmask from which the properties can be read (for details see below)	
	`< 0`	Error	
`PadCapabilities`	`!= NULL`	Class instance from which the properties can be read (for details see below)	

Support of a property is indicated for the STPadCapt.ocx and STPadLib.dll components through a set bit in the returned bitmask. In the case of the STPadLibNet.dll component, the respective property of the returned instance will be `true` if the property is supported. The following properties can be currently supported:

STPadCapt.ocx STPadLib.dll	STPadLibNet.dll	Description
`0x00000001`	`HasColorDisplay`	Device has a colour screen
`0x00000002`	`HasBacklight`	Device has a backlit screen
`0x00000004`	`SupportsVerticalScrolling`	Device supports vertical scrolling
`0x00000008`	`SupportsHorizontalScrolling`	Device supports horizontal scrolling
`0x00000010`	`SupportsPenScrolling`	Device supports scrolling with the pen
`0x00000020`	`SupportsServiceMenu`	The service menu can be called using the DeviceStartService() method

0x00000040	SupportsRSA	Device supports the RSA functions
0x00000080	SupportsContentSigning	Device supports content signing
0x00000100	SupportsH2ContentSigning	Device supports content signing where only hash 2 is signed
0x00000200	CanGenerateSignKey	Device can generate a signature key pair
0x00000400	CanStoreSignKey	Device can save an externally supplied signature key pair
0x00000800	CanStoreEncryptKey	Device can save an externally supplied biometric key
0x00001000	CanSignExternalHash	Device can sign a hash calculated externally (otherwise only a hash calculated through content signing)
0x00002000	SupportsRSAPassword	RSA keys can be protected through a device password.
0x00004000	SupportsSecureModePassword	The 'Secure mode' can be protected through a device password.
0x00008000	Supports4096BitKeys	Device supports RSA keys with a length of up to 4096 bits (otherwise up to 2048 bits)
0x00010000	HasNFCReader	Device has an NFC reader
0x00020000	Keypad	Device supports keypad encryption with a length of up to 8 characters
0x00040000	Keypad32	Device supports keypad encryption with a length of up to 32 characters
0x00080000	HasDisplay	Device has a screen (please note: Support for this property is always displayed even with a Sigma Lite up to firmware 2.3)
0x00100000	RSASignPassword	The signature key can be protected by a key password

8.8.1 STPadCapt.ocx

Available from Version 8.4.0. The status described is available from Version 8.4.2.4.

```
LONG DeviceGetCapabilities(LONG nIndex)
```

8.8.1.1 Implementation in C#

```
int nCapabilities = axSTPadCapt1.DeviceGetCapabilities(0);
if (nCapabilities < 0)
    MessageBox.Show(String.Format("Error {0}", nCapabilities);
else if (nCapabilities & 0x40)
    MessageBox.Show("Device supports RSA");
```

8.8.1.2 Implementation in Visual Basic

```
Dim nCapabilities As Integer = AxSTPadCapt1.DeviceGetCapabilities(0)
If nCapabilities < 0 Then
    MsgBox("Error " & CStr(nCapabilities))
ElseIf nCapabilities And &H40& Then
    MsgBox("Device supports RSA")
End If
```

8.8.2 STPadLib.dll

Available from Version 8.4.0. The status described is available from Version 8.4.2.4.

```
LONG STDeviceGetCapabilities(LONG nIndex)
```

The following values defined in the header file can be used to evaluate the return value:

```
#define STPAD_CAP_COLORDISPLAY            0x000001
#define STPAD_CAP_BACKLIGHT               0x000002
#define STPAD_CAP_VERTICALSCROLLING       0x000004
#define STPAD_CAP_HORIZONTALSCROLLING     0x000008
#define STPAD_CAP_PENSCROLLING            0x000010
#define STPAD_CAP_SERVICEMENU             0x000020
#define STPAD_CAP_RSA                     0x000040
#define STPAD_CAP_CONTENTSIGNING          0x000080
#define STPAD_CAP_H2CONTENTSIGNING        0x000100
#define STPAD_CAP_GENERATESIGNKEY         0x000200
#define STPAD_CAP_STORESIGNKEY            0x000400
#define STPAD_CAP_STOREENCRYPTKEY         0x000800
#define STPAD_CAP_SIGNEXTERNALHASH        0x001000
#define STPAD_CAP_RSAPASSWORD             0x002000
#define STPAD_CAP_SECUREMODEPASSWORD      0x004000
#define STPAD_CAP_4096BITKEY              0x008000
#define STPAD_CAP_NFCREADER               0x010000
#define STPAD_CAP_KEYPAD                  0x020000
#define STPAD_CAP_KEYPAD32                0x040000
#define STPAD_CAP_DISPLAY                 0x080000
#define STPAD_CAP_RSASIGNPASSWORD         0x100000
```

8.8.2.1 Implementation in C++

```
LONG nCapabilities = STDeviceGetCapabilities(0);
if (nCapabilities < 0)
    wprintf(L"Error %d", nCapabilities);
else if (nCapabilities & 0x40)
    wprintf(L"Device supports RSA");
```

8.8.3 STPadLibNet.dll

Available from Version 8.4.0. The status described is available from Version 8.4.2.4.

```
PadCapabilities DeviceGetCapabilities(int index)
```

```
Function DeviceGetCapabilities(ByVal index As Integer) As PadCapabilities
```

8.8.3.1 Implementation in C#

```
try
{
    if (stPad.DeviceGetCapabilities(0).SupportsRSA)
        MessageBox.Show("Device supports RSA");
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.8.3.2 Implementation in Visual Basic

```
Try
    If STPad.DeviceGetCapabilities(0).SupportsRSA Then
        MsgBox("Device supports RSA")
    End If
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.9 DeviceOpen method

This method opens a connection to a device. The backlight is switched on if this is configured in the STPad.ini file (see method `DisplaySetBacklight()`).

Please observe the relevant information in the 'Using multiple instances' section.

Parameter	Values	I/O	Description

`LONG nIndex`
`int index`
`ByVal index As Integer`
	`>= 0`	I	Index of the device to which a connection is to be opened

`VARIANT bEraseDisplay`
`BOOL bEraseDisplay`
`bool eraseDisplay`
`ByVal eraseDisplay As Boolean`
	`true`	I	The device display screen will be erased (Default)

	`false`	I	The content of the display screen will not change; the screen content displayed cannot be copied into another image memory at a later stage; the control element cannot display the screen content (Optional)
Return value	**Values**	**Description**	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	

8.9.1 STPadCapt.ocx

Available from Version 8.0.1. The status described is available from Version 8.0.25.

```
LONG DeviceOpen(LONG nIndex, [optional]VARIANT bEraseDisplay)
```

Note: The `bEraseDisplay` parameter is optional and must contain a Boolean value if it is transferred.

8.9.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.DeviceOpen(0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.9.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.DeviceOpen(0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.9.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.0.25.

```
LONG STDeviceOpen(LONG nIndex, BOOL bEraseDisplay=TRUE)
```

8.9.2.1 Implementation in C++

```
LONG nRc = STDeviceOpen(0);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.9.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.0.25.

```
void DeviceOpen(int index)
void DeviceOpen(int index, bool eraseDisplay)
Sub DeviceOpen(ByVal index As Integer)
Sub DeviceOpen(ByVal index As Integer, ByVal eraseDisplay As Boolean)
```

8.9.3.1 Implementation in C#

```
try
{
    stPad.DeviceOpen(0);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.9.3.2 Implementation in Visual Basic

```
Try
    STPad.DeviceOpen(0)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.10 DeviceClose method

This method closes the connection to a device. It can also be opened in another instance, provided it is running in the same memory area as the instance that is currently being used. Before closing, a currently running signature capture process is terminated and the backlight switched off, where appropriate, if so configured in the STPad.ini file (see method `DisplaySetBacklight()`).

Captured signature data is discarded. When the STPadCapt.ocx component is used, this method is called automatically as soon as the window containing the control element is closed.

Parameter	Values	I/O	Description
LONG nIndex int index ByVal index As Integer	>= 0	I	Index of the device whose connection is to be closed
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.10.1 STPadCapt.ocx

Available from Version 8.0.1.

```
LONG DeviceOpen(LONG nIndex)
```

8.10.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.DeviceClose(0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.10.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.DeviceClose(0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.10.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDeviceClose(LONG nIndex)
```

8.10.2.1 Implementation in C++

```
LONG nRc = STDeviceClose(0);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.10.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void DeviceClose(int index)
```

```
Sub DeviceClose(ByVal index As Integer)
```

8.10.3.1 Implementation in C#

```
try
{
    stPad.DeviceClose(0);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.10.3.2 Implementation in Visual Basic

```
Try
    STPad.DeviceClose(0)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.11 DeviceSetLed method

This method sets the colour of the LED on the front of the pad. The `DeviceLedDefaultFlag` property should be set to FALSE when this method is used, in order to ensure that the colour is not changed when `SignatureStart()`, `SignatureCancel()` and `SignatureConfirm()` are called. The LED always lights up yellow as soon as the device has been detected by the PC operating system and is ready for use.

Parameter	Values	I/O	Description

`LONG nLedColor`
 	
`LedColorFlag ledColor`
 	
`ByVal ledColor As LedColorFlag`
	Bitmask containing one or more hexadecimal values from the following list:		

	`0x01`	I	Yellow
	`0x02`	I	Green
Return value	**Values**	**Description**	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	

8.11.1 STPadCapt.ocx

Available from Version 8.0.1.

```
LONG DeviceSetLed(LONG nLedColor)
```

8.11.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.DeviceSetLed(1);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.11.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.DeviceSetLed(1)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.11.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDeviceSetLed(LONG nLedColor)
```

The following values defined in the header file can be used for the `nLedColor` parameter:

```
#define STPAD_LED_YELLOW 0x01
#define STPAD_LED_GREEN  0x02
```

8.11.2.1 Implementation in C++

```
LONG nRc = STDeviceSetLed(STPAD_LED_YELLOW);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.11.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void DeviceSetLed(signotec.STPadLibNet.LedColorFlag ledColor)
```

```
Sub DeviceSetLed(ByVal ledColor As signotec.STPadLibNet.LedColorFlag)
```

The `LedColorFlag` enumeration is defined as follows:

```
Off = 0,
Yellow = 0x01,
Green = 0x02
```

8.11.3.1 Implementation in C#

```
try
{
    stPad.DeviceSetLed(LedColorFlag.Yellow);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.11.3.2 Implementation in Visual Basic

```
Try
    STPad.DeviceSetLed(LedColorFlag.Yellow)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.12 DeviceGetNFCMode method

This method reads out the operating mode of the optionally installed NFC reader. Whether the connected device has an NFC reader can be queried using the `DeviceGetCapabilities()` method.

Parameter	Values	I/O	Description
LONG nIndex int index ByVal index As Integer	>= 0	I	Index of the device whose information is to be queried
Return value	**Values**	**Description**	
LONG	0–1	Operating mode (see also DeviceSetNFCMode())	
	< 0	Error	
NFCMode	Off On	Operating mode (see also DeviceSetNFCMode())	

8.12.1 STPadCapt.ocx

Available from Version 8.4.0. The status described is available from Version 8.4.1010.

```
LONG DeviceGetNFCMode(LONG nIndex)
```

8.12.1.1 Implementation in C#

```
int nMode = axSTPadCapt1.DeviceGetNFCMode(0);
switch (nMode)
{
    case 0:
        MessageBox.Show("The NFC reader is currently switched off.");
        break;
    case 1:
        MessageBox.Show("The NFC reader is currently switched on.");
        break;
    default:
        MessageBox.Show(String.Format("Error {0}", nMode);
        break;
}
```

8.12.1.2 Implementation in Visual Basic

```
Dim nMode As Integer = AxSTPadCapt1.DeviceGetNFCMode(0)
Select Case nMode
    Case 0
        MsgBox("The NFC reader is currently switched off.")
    Case 1
        MsgBox("The NFC reader is currently switched on.")
    Case Else
        MsgBox("Error " & CStr(nMode))
End Select
```

8.12.2 STPadLib.dll

Available from Version 8.4.0. The status described is available from Version 8.4.1010.

```
LONG STDeviceGetNFCMode(LONG nIndex)
```

The following values defined in the header file can be used for the nMode parameter:

```
#define STPAD_NFC_OFF   0
#define STPAD_NFC_ON    1
```

8.12.2.1 Implementation in C++

```
LONG nMode = STDeviceGetNFCMode(0);
switch (nMode)
{
    case STPAD_NFC_OFF:
        wprintf(L"The NFC reader is currently switched off.");
        break;
    case STPAD_NFC_ON:
        wprintf(L"The NFC reader is currently switched on.");
        break;
    default:
        wprintf(L"Error %d", nMode);
        break;
}
```

8.12.3 STPadLibNet.dll

Available from Version 8.4.0. The status described is available from Version 8.4.1010.

```
NFCMode DeviceGetNFCMode(int index)
```

```
Function DeviceGetNFCMode(ByVal index As Integer) As
signotec.STPadLibNet.NFCMode
```

The `NFCMode` enumeration is defined as follows:

```
Off = 0,
On = 1
```

8.12.3.1 Implementation in C#

```
try
{
    switch (stPad.DeviceGetNFCMode(0))
    {
        case NFCMode.Off:
            MessageBox.Show("The NFC reader is currently switched off.");
            break;
        case NFCMode.On:
            MessageBox.Show("The NFC reader is currently switched on.");
            break;
    }
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.12.3.2 Implementation in Visual Basic

```
Try
    Select Case STPad.DeviceGetNFCMode(0)
        Case NFCMode.Off
            MsgBox("The NFC reader is currently switched off.")
        Case NFCMode.On
            MsgBox("The NFC reader is currently switched on.")
    End Select
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.13 DeviceSetNFCMode method

This method changes the operating mode of the optionally installed NFC reader. It can only be called if a connection to the device has not been opened in another application. Whether the connected device has an NFC reader can be queried using the `DeviceGetCapabilities()` method.

Parameter	Values	I/O	Description

`LONG nMode`
`NFCMode mode`
`ByVal mode As NFCMode`
	0	I	The NFC reader is turned off; after a restart, it is in standard operating mode again.

	1	I	The NFC reader is turned on; after a restart, it is in standard operating mode again.
	2	I	The NFC reader is turned off; the standard operating mode is likewise set to 'off'.
	3	I	The NFC reader is turned on; the standard operating mode is likewise set to 'on'.

`LONG nIndex`
`int index`
`ByVal index As Integer`
	>= 0	I	Index of the device whose mode is to be changed.

Return value	**Values**	**Description**	
`LONG`	0	Method was executed successfully	
	< 0	Error	

8.13.1 STPadCapt.ocx

Available from Version 8.4.0. The status described is available from Version 8.4.1010.

```
LONG DeviceSetNFCMode(LONG nMode, LONG nIndex)
```

8.13.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.DeviceSetNFCMode(1, 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.13.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.DeviceSetNFCMode(1, 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.13.2 STPadLib.dll

Available from Version 8.4.0. The status described is available from Version 8.4.1010.

```
LONG STDeviceSetNFCMode(LONG nMode, LONG nIndex)
```

The following values defined in the header file can be used for the `nMode` parameter:

```
#define STPAD_NFC_OFF               0
#define STPAD_NFC_ON                1
#define STPAD_NFC_PERMANENTLYOFF    2
#define STPAD_NFC_PERMANENTLYON     3
```

8.13.2.1 Implementation in C++

```
LONG nRc = STDeviceSetNFCMode(STPAD_NFC_ON, 0);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.13.3 STPadLibNet.dll

Available from Version 8.4.0. The status described is available from Version 8.4.1010.

```
void DeviceSetNFCMode(signotec.STPadLibNet.NFCMode mode, int index)
```

```
Sub DeviceSetNFCMode(ByVal mode As signotec.STPadLibNet.NFCMode, ByVal index As
Integer)
```

The `NFCMode` enumeration is defined as follows:

```
Off = 0,
On = 1,
PermanentlyOff = 2,
PermanentlyOn = 3
```

8.13.3.1 Implementation in C#

```
try
{
    stPad.DeviceSetNFCMode(NFCMode.On, 0);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.13.3.2 Implementation in Visual Basic

```
Try
    STPad.DeviceSetNFCMode(NFCMode.On, 0)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.14 DeviceStartService method

This method starts one of the configuration dialog boxes on the signature device. The device cannot be reached as long as the dialog is displayed. The device restarts if the type of connection or the IP configuration is adjusted in the configuration dialog box.

Parameter	Values	I/O	Description
LONG nType int type ByVal type As Integer	0	I	Starts the service menu in which the type of connection, the IP configuration and the screen brightness can be changed; this is only supported by the Sigma model from firmware 2.10, the Zeta model, the Omega model from firmware 2.0, the Gamma model from firmware 1.6, the Delta model and the Alpha model, provided that the device has a backlit screen
	1	I	Starts screen calibration
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.14.1 STPadCapt.ocx

Available from Version 8.1.2. The status described is available from Version 8.4.0.

```
LONG DeviceStartService(LONG nType)
```

8.14.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.DeviceStartService(1);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.14.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.DeviceStartService(1)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.14.2 STPadLib.dll

Available from Version 8.1.2. The status described is available from Version 8.4.0.

```
LONG STDeviceStartService(LONG nType)
```

8.14.2.1 Implementation in C++

```
LONG nRc = STDeviceStartService(1);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.14.3 STPadLibNet.dll

Available from Version 8.1.2. The status described is available from Version 8.4.0.

```
void DeviceStartService(int type)
```

```
Sub DeviceStartService(ByVal type As Integer)
```

8.14.3.1 Implementation in C#

```
try
{
    stPad.DeviceStartService(1);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.14.3.2 Implementation in Visual Basic

```
Try
    STPad.DeviceStartService(1)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.15 SensorGetSampleRateMode method

This method returns the configured sample rate with which the signature is captured.

Parameter	Values	I/O	Description
–	–	-	-

Return value	Values	Description
LONG SampleRate	3	280 Hz
	2	500 Hz
	1	250 Hz
	0	125 Hz
	< 0	Error (not STPadLibNet.dll)

8.15.1 STPadCapt.ocx

Available from Version 8.0.1. The status described is available from Version 8.1.1.

```
LONG SensorGetSampleRateMode()
```

8.15.1.1 Implementation in C#

```
int nMode = axSTPadCapt1.SensorGetSampleRateMode();
switch (nMode)
{
    case 0:
        MessageBox.Show("Sample rate is 125 Hz.");
        break;
    case 1:
        MessageBox.Show("Sample rate is 250 Hz.");
        break;
    case 2:
        MessageBox.Show("Sample rate is 500 Hz.");
        break;
    case 3:
        MessageBox.Show("Sample rate is 280 Hz.");
        break;
    default:
        MessageBox.Show(String.Format("Error {0}", nMode);
        break;
}
```

8.15.1.2 Implementation in Visual Basic

```
Dim nMode As Integer = AxSTPadCapt1.SensorGetSampleRateMode
Select Case nMode
    Case 0
        MsgBox("Sample rate is 125 Hz.")
    Case 1
        MsgBox("Sample rate is 250 Hz.")
    Case 2
        MsgBox("Sample rate is 500 Hz.")
    Case 3
        MsgBox("Sample rate is 280 Hz.")
    Case Else
        MsgBox("Error " & CStr(nMode))
End Select
```

8.15.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.1.1.

```
LONG STSensorGetSampleRateMode()
```

8.15.2.1 Implementation in C++

```
LONG nMode = STSensorGetSampleRateMode();
switch (nMode)
{
    case 0:
        wprintf(L"Sample rate is 125 Hz.");
        break;
    case 1:
        wprintf(L"Sample rate is 250 Hz.");
        break;
    case 2:
        wprintf(L"Sample rate is 500 Hz.");
        break;
    case 3:
        wprintf(L"Sample rate is 280 Hz.");
        break;
    default:
        wprintf(L"Error %d", nMode);
        break;
}
```

8.15.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.1.1.

`signotec.STPadLibNet.SampleRate SensorGetSampleRateMode()`

`Function SensorGetSampleRateMode() As signotec.STPadLibNet.SampleRate`

The `SampleRate` enumeration is defined as follows:

```
Hz125 = 0,
Hz250 = 1,
Hz500 = 2,
Hz280 = 3
```

8.15.3.1 Implementation in C#

```
try
{
    switch (stPad.SensorGetSampleRateMode())
    {
        case SampleRate.Hz125:
            MessageBox.Show("Sample rate is 125 Hz.");
            break;
        case SampleRate.Hz250:
            MessageBox.Show("Sample rate is 250 Hz.");
            break;
        case SampleRate.Hz500:
            MessageBox.Show("Sample rate is 500 Hz.");
            break;
        case SampleRate.Hz280:
            MessageBox.Show("Sample rate is 280 Hz.");
            break;
    }
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.15.3.2 Implementation in Visual Basic

```
Try
    Select Case STPad.SensorGetSampleRateMode()
        Case SampleRate.Hz125
            MsgBox("Sample rate is 125 Hz.")
        Case SampleRate.Hz250
            MsgBox("Sample rate is 250 Hz.")
        Case SampleRate.Hz500
            MsgBox("Sample rate is 500 Hz.")
        Case SampleRate.Hz280
            MsgBox("Sample rate is 280 Hz.")

    End Select

Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.16 SensorSetSampleRateMode method

This method sets the sample rate with which the signature is captured. The default setting is mode 1 (250 Hz) or mode 3 (280 Hz) when using the Alpha model. This mode provides high-quality signature data while at the same time ensures that the data record is of moderate size. When using the Sigma, Zeta, Gamma and Omega models, this value can easily be set to 2 (500 Hz) for high-speed data lines.

Parameter	Values	I/O	Description
LONG nMode SampleRate mode ByVal mode As SampleRate	0	I	125 Hz (Sigma, Zeta, Omega, Gamma and Delta only)
	1	I	250 Hz (Sigma, Zeta, Omega, Gamma and Delta only)
	2	I	500 Hz (Sigma, Zeta, Omega, Gamma and Delta only)
	3	I	280 Hz (Alpha only)
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.16.1 STPadCapt.ocx

Available from Version 8.0.1. The status described is available from Version 8.1.1.

```
LONG SensorSetSampleRateMode(LONG nMode)
```

8.16.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.SensorSetSampleRateMode(1);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.16.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.SensorSetSampleRateMode(1)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.16.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.1.1.

```
LONG STSensorSetSampleRateMode(LONG nMode)
```

8.16.2.1 Implementation in C++

```
LONG nRc = STSensorSetSampleRateMode(1);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.16.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.1.1.

```
void SensorSetSampleRateMode(signotec.STPadLibNet.SampleRate mode)
```

```
Sub SensorSetSampleRateMode(ByVal mode As signotec.STPadLibNet.SampleRate)
```

The `SampleRate` enumeration is defined as follows:

```
Hz125 = 0,
Hz250 = 1,
Hz500 = 2,
Hz280 = 3
```

8.16.3.1 Implementation in C#

```
try
{
    stPad.SensorSetSampleRateMode(SampleRate.Hz250);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.16.3.2 Implementation in Visual Basic

```
Try

    STPad.SensorSetSampleRateMode(SampleRate.Hz250)

Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.17 SensorSetSignRect method

This method defines the rectangle in which the signature is captured. If the rectangle overlaps with one of the fixed hotspots (see `SensorAddHotSpot()`), an error is returned.

Parameter	Values	I/O	Description

`LONG nLeft`
`int left`
`ByVal left As Integer`
	`>= 0`	I	Left boundary; 0 is on the far left of the display

`LONG nTop`
`int top`
`ByVal top As Integer`
	`>= 0`	I	Upper boundary; 0 is at the top of the display

`LONG nWidth`
`int width`
`ByVal width As Integer`
	`> 3`	I	Width; `DisplayWidth` holds the width of the LCD used

	0	I	Right boundary is automatically set to the maximum value (right margin of the LCD)

`LONG nHeight`
`int height`
`ByVal height As Integer`
	`> 3`	I	Height; `DisplayHeight` holds the height of the LCD used

	0	I	Lower boundary is automatically set to the maximum value (lower margin of the LCD)

Return value	Values	Description
`LONG`	0	Method was executed successfully
	`< 0`	Error

8.17.1 STPadCapt.ocx

Available from Version 8.0.1. The status described is available from Version 8.4.1.5.

```
LONG SensorSetSignRect(LONG nLeft, LONG nTop, LONG nWidth, LONG nHeight)
```

8.17.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.SensorSetSignRect(0, 40, 0, 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.17.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.SensorSetSignRect(0, 40, 0, 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.17.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.4.1.5.

```
LONG STSensorSetSignRect(LONG nLeft, LONG nTop, LONG nWidth, LONG nHeight)
```

8.17.2.1 Implementation in C++

```
LONG nRc = STSensorSetSignRect(0, 40, 0, 0);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.17.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.4.1.5.

```
void SensorSetSignRect(int left, int top, int width, int height)
```

```
Sub SensorSetSignRect(ByVal left As Integer, ByVal top As Integer, ByVal width
As Integer, ByVal height As Integer)
```

8.17.3.1 Implementation in C#

```
try
{
    stPad.SensorSetSignRect(0, 40, 0, 0);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.17.3.2 Implementation in Visual Basic

```
Try
    STPad.SensorSetSignRect(0, 40, 0, 0)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.18 SensorClearSignRect method

This method erases the signature window.

Parameter	Values	I/O	Description
-	-	-	-
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.18.1 STPadCapt.ocx

Available from Version 8.0.1.

```
LONG SensorClearSignRect()
```

8.18.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.SensorClearSignRect();
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.18.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.SensorClearSignRect
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.18.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STSensorClearSignRect()
```

8.18.2.1 Implementation in C++

```
LONG nRc = STSensorClearSignRect();
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.18.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void SensorClearSignRect()
```

```
Sub SensorClearSignRect()
```

8.18.3.1 Implementation in C#

```
try
{
    stPad.SensorClearSignRect();
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.18.3.2 Implementation in Visual Basic

```
Try
    STPad.SensorClearSignRect()
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.19 SensorSetScrollArea method

This method defines a rectangular subarea of the non-volatile memory whose content can be scrolled. The subarea must be at least as big as the display and fully encompass the displayed area. Once a connection has been opened, the entire memory is set as the scroll area.

Parameter	Values	I/O	Description

`LONG nLeft`
`int left`
`ByVal left As Integer`
	`>= 0`	I	Left boundary; 0 is on the far left of the memory

`LONG nTop`
`int top`
`ByVal top As Integer`
	`>= 0`	I	Upper boundary; 0 is at the top of the memory

`LONG nWidth`
`int width`
`ByVal width As Integer`
	`> 0`	I	Width, must be >= `DisplayWidth`; `DisplayTargetWidth` contains the width of the currently set memory

	`0`	I	Right boundary is automatically set to the maximum value (right margin of the memory)

`LONG nHeight`
`int height`
`ByVal height As Integer`
	`> 0`	I	Height must be >= `DisplayHeight`; `DisplayTargetHeight` contains the height of the currently set memory

	`0`	I	Lower boundary is automatically set to the maximum value (lower margin of the memory)
Return value	**Values**	**Description**	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	

8.19.1 STPadCapt.ocx

Available from Version 8.3.1.

```
LONG SensorSetScrollArea(LONG nLeft, LONG nTop, LONG nWidth, LONG nHeight)
```

8.19.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.SensorSetScrollArea(0, 0, 0, 960);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.19.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.SensorSetScrollArea(0, 0, 0, 960)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.19.2 STPadLib.dll

Available from Version 8.3.1.

```
LONG STSensorSetScrollArea(LONG nLeft, LONG nTop, LONG nWidth, LONG nHeight)
```

8.19.2.1 Implementation in C++

```
LONG nRc = STSensorSetScrollArea(0, 0, 0, 960);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.19.3 STPadLibNet.dll

Available from Version 8.3.1.

```
void SensorSetScrollArea(int left, int top, int width, int height)
```

```
Sub SensorSetScrollArea(ByVal left As Integer, ByVal top As Integer, ByVal
width As Integer, ByVal height As Integer)
```

8.19.3.1 Implementation in C#

```
try
{
    stPad.SensorSetScrollArea(0, 0, 0, 960);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.19.3.2 Implementation in Visual Basic

```
Try
    STPad.SensorSetScrollArea(0, 0, 0, 960)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.20 SensorSetPenScrollingEnabled method

This method activates scrolling with the pen. In this mode, the memory contents can be offset by moving the pen on the display within the area set via SensorSetScrollArea. The application is informed about this via the `DisplayScrollPosChanged()` event.

This method only works with the Delta model.

Parameter	Values	I/O	Description
VARIANT_BOOL bEnable BOOL bEnable bool enable ByVal enable As Boolean	true	I	Scrolling with the pen is activated; signature capture must not have been started.
	false	I	Deactivates scrolling with the pen
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.20.1 STPadCapt.ocx

Available from Version 8.3.1. The status described is available from Version 8.4.1.5.

```
LONG SensorSetPenScrollingEnabled(VARIANT_BOOL bEnable)
```

8.20.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.SensorSetPenScrollingEnabled(true);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.20.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.SensorSetPenScrollingEnabled(True)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.20.2 STPadLib.dll

Available from Version 8.3.1. The status described is available from Version 8.4.1.5.

```
LONG STSensorSetPenScrollingEnabled(BOOL bEnable)
```

8.20.2.1 Implementation in C++

```
LONG nRc = STSensorSetPenScrollingEnabled(TRUE);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.20.3 STPadLibNet.dll

Available from Version 8.3.1. The status described is available from Version 8.4.1.5.

```
void SensorSetPenScrollingEnabled(bool enable)
```

```
Sub SensorSetPenScrollingEnabled(ByVal enable As Boolean)
```

8.20.3.1 Implementation in C#

```
try
{
    stPad.SensorSetPenScrollingEnabled(true);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.20.3.2 Implementation in Visual Basic

```
Try
    STPad.SensorSetPenScrollingEnabled(True)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.21 SensorAddHotSpot method

This method defines a rectangular subarea of the sensor surface that responds to user clicks. See also `SensorHotSpotPressed()`. If a scroll hotspot has already been defined, scrolling has been performed or pen-controlled scrolling (see also `SensorSetPenScrollingEnabled()`) is active, the rectangle must be in the area defined via `DisplaySetOverlayRect()`. It should not overlap with the defined signature window (see `SensorSetSignRect()`) or a hotspot that was previously set.

Parameter	Values	I/O	Description

`LONG nLeft`
`int left`
`ByVal left As Integer`
	`>= 0`	I	Left boundary; 0 is on the far left of the display

`LONG nTop`
`int top`
`ByVal top As Integer`
	`>= 0`	I	Upper boundary; 0 is at the top of the display

`LONG nWidth`
`int width`
`ByVal width As Integer`
	`> 0`	I	Width; `DisplayWidth` holds the width of the LCD used

	`0`	I	Right boundary is automatically set to the maximum value (right edge of the LCD)

`LONG nHeight`
`int height`
`ByVal height As Integer`
	`> 0`	I	Height; `DisplayHeight` holds the height of the LCD used

	`0`	I	Lower boundary is automatically set to the maximum value (lower edge of the LCD)
Return value	**Values**	**Description**	
`LONG`	`>= 0`	ID of the hotspot that was generated	
	`< 0`	Error (not STPadLibNet.dll)	

8.21.1 STPadCapt.ocx

Available from Version 8.0.1. The status described is available from Version 8.3.1.

```
LONG SensorAddHotSpot(LONG nLeft, LONG nTop, LONG nWidth, LONG nHeight)
```

8.21.1.1 Implementation in C#

```
int nHotspotId = axSTPadCapt1.SensorAddHotSpot(0, 0, 0, 40);
if (nHotspotId < 0)
    MessageBox.Show(String.Format("Error {0}", nHotspotId);
```

8.21.1.2 Implementation in Visual Basic

```
Dim nHotpotId As Integer = AxSTPadCapt1.SensorAddHotSpot(0, 0, 0, 40)
If nHotspotId < 0 Then
    MsgBox("Error " & CStr(nHotspotId))
End If
```

8.21.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STSensorAddHotSpot(LONG nLeft, LONG nTop, LONG nWidth, LONG nHeight)
```

8.21.2.1 Implementation in C++

```
LONG nRc = STSensorAddHotSpot(0, 0, 0, 40);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.21.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void SensorAddHotSpot(int left, int top, int width, int height)
```

```
Sub SensorAddHotSpot(ByVal left As Integer, ByVal top As Integer, ByVal width
As Integer, ByVal height As Integer)
```

8.21.3.1 Implementation in C#

```
try
{
    STPad.SensorAddHotSpot(0, 0, 0, 40);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.21.3.2 Implementation in Visual Basic

```
Try
    STPad.SensorAddHotSpot(0, 0, 0, 40)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.22 SensorAddScrollHotSpot method

This method defines a rectangular subarea of the sensor surface that responds to user clicks. Depending on the option that is used, the subarea is either created as a scroll hotspot or as a scrollable hotspot.

If a scroll hotspot is activated by the user, the screen content is scrolled to the left, to the right, up or down at the speed defined by the `DisplayScrollSpeed` property and the `DisplayScrollPosChanged()` event is called. A scroll hotspot only responds to clicks if it lies in the area defined by `DisplaySetOverlayRect()`; otherwise it is inactive.

The rectangle should not overlap with the defined signature window (see `SensorSetSignRect()`) or a fixed hotspot that was previously set.

A scrollable hotspot behaves like a normal hotspot (also see `SensorAddHotSpot()`), however, it is moved with the displayed content during scrolling. A scrollable hotspot does not respond to clicks if it lies in the area defined by `DisplaySetOverlayRect()`.

The rectangle should not overlap with a scrollable hotspot that was previously set. The rectangle should not overlap with the defined signature window (see `SensorSetSignRect()`) while a signature capture process is currently running.

This method only works with the Omega, Gamma and Delta models as well as with Alpha models with firmware 1.8 or a newer version.

Parameter	Values	I/O	Description

`LONG nLeft`
`int left`
`ByVal left As Integer`
	`>= 0`	I	Left boundary; 0 is on the far left of the display

`LONG nTop`
`int top`
`ByVal top As Integer`
	`>= 0`	I	Upper boundary; 0 is at the top of the display

`LONG nWidth`
`int width`
`ByVal width As Integer`
	`> 0`	I	Width; `DisplayWidth` holds the width of the LCD used

	`0`	I	Right boundary is automatically set to the maximum value (right edge of the LCD)

`LONG nHeight`
`int height`
`ByVal height As Integer`
	`> 0`	I	Height; `DisplayHeight` holds the height of the LCD used

	`0`	I	Lower boundary is automatically set to the maximum value (lower edge of the LCD)

LONG nType HOTSPOTTYPE nType ScrollOption type ByVal type As ScrollOption	0	I	Scroll hotspot: Pressing the hotspot moves the screen content up (scrolls down)
	1	I	Scroll hotspot: Pressing the hotspot moves the screen content down (scrolls up)
	2	I	Scroll hotspot: Pressing the hotspot moves the screen content to the left (scrolls right); only available with the Alpha model
	3	I	Scroll hotspot: Pressing the hotspot moves the screen content to the right (scrolls left); only available with the Alpha model
	4	I	Scrollable hotspot; only available with the Delta model
Return value	**Values**	**Description**	
LONG	>= 0	ID of the hotspot that was generated	
	< 0	Error (not STPadLibNet.dll)	

8.22.1 STPadCapt.ocx

Available from Version 8.0.17. The status described is available from Version 8.4.1.5.

```
LONG SensorAddScrollHotSpot(LONG nLeft, LONG nTop, LONG nWidth, LONG nHeight,
LONG nTYPE)
```

8.22.1.1 Implementation in C#

```
int nHotspotId = axSTPadCapt1.SensorAddScrollHotSpot(0, 0, 0, 40, 0);
if (nHotspotId < 0)
    MessageBox.Show(String.Format("Error {0}", nHotspotId);
```

8.22.1.2 Implementation in Visual Basic

```
Dim nHotpotId As Integer
nHotpotId = AxSTPadCapt1.SensorAddScrollHotSpot(0, 0, 0, 40, 0)
If nHotspotId < 0 Then
    MsgBox("Error " & CStr(nHotspotId))
End If
```

8.22.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.4.1.5.

```
LONG STSensorAddScrollHotSpot(LONG nLeft, LONG nTop, LONG nWidth, LONG nHeight,
HOTSPOTTYPE nType)
```

The `HOTSPOTTYPE` enumeration is defined as follows:

```
kScrollDown = 0,
kScrollUp = 1,
kScrollRight = 2,
kScrollLeft = 3,
kScrollable = 4
```

8.22.2.1 Implementation in C++

```
LONG nRc = STSensorAddScrollHotSpot(0, 0, 0, 40, kScrollDown);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.22.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.4.1.5.

```
void SensorAddScrollHotSpot(int left, int top, int width, int height,
signotec.STPadLibNet.ScrollOption type)
```

```
Sub SensorAddScrollHotSpot(ByVal left As Integer, ByVal top As Integer, ByVal
width As Integer, ByVal height As Integer, signotec.STPadLibNet.ScrollOption
type)
```

The `ScrollOption` enumeration is defined as follows:

```
ScrollDown = 0,
ScrollUp = 1,
ScrollRight = 2,
ScrollLeft = 3,
Scrollable = 4
```

8.22.3.1 Implementation in C#

```
try
{
    stPad.SensorAddScrollHotSpot(0, 0, 0, 40, ScrollOption.ScrollDown);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.22.3.2 Implementation in Visual Basic

```
Try
    STPad.SensorAddScrollHotSpot(0, 0, 0, 40, ScrollOption.ScrollDown)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.23 SensorAddKeypadHotSpot method

This method defines a rectangular subarea of the sensor surface that responds to user clicks. This hotspot type is subject to a special security aspect: Although the `SensorHotSpotPressed()` event is triggered when clicked, the ID of the clicked hotspot is not transmitted but is instead kept in a list in the signature device. This list can be read in encrypted form using the `SensorGetKeypadEntries()` method or deleted using the `SensorClearKeypadEntries()` method. This hotspot type is therefore suitable for sensitive input such as PINs.

Depending on the device properties, the list in the device may hold up to eight or 32 entries before an overflow occurs; hotspots can still be clicked when a list is filled, however, the information will be lost. The application must therefore read out the list in good time if more than eight or 32 characters need to be entered. See also `DeviceGetCapabilities()`.

If a scroll hotspot has already been defined, scrolling has been performed or pen-controlled scrolling (see also `SensorSetPenScrollingEnabled()`) is active, the rectangle must be in the area defined via `DisplaySetOverlayRect()` . It should not overlap with the defined signature window (see `SensorSetSignRect()`) or a hotspot that was previously set.

Please note that adding a keypad hotspot empties the list in the device. Therefore, if necessary, make sure to read out the list first using the `SensorGetKeypadEntries()` method.

This method works with the Sigma model from firmware 2.10, the Zeta model from firmware 1.0, the Omega model from firmware 2.14, the Gamma model from firmware 1.20 and the Delta model from firmware 1.22. For the Sigma model up to firmware 2.14, the Omega model up to firmware 2.18, the Gamma model up to firmware 1.33 and the Delta model up to firmware 1.35, the restriction applies that these methods only work if the memory defined with the `DisplaySetTarget()` method is the foreground memory.

Parameter	Values	I/O	Description

`LONG nLeft`
`int left`
`ByVal left As Integer`
	`>= 0`	I	Left boundary; 0 is on the far left of the display

`LONG nTop`
`int top`
`ByVal top As Integer`
	`>= 0`	I	Upper boundary; 0 is at the top of the display

`LONG nWidth`
`int width`
`ByVal width As Integer`
	`> 0`	I	Width; `DisplayWidth` holds the width of the LCD used

	`0`	I	Right boundary is automatically set to the maximum value (right edge of the LCD)

`LONG nHeight`
`int height`
`ByVal height As Integer`
	`> 0`	I	Height; `DisplayHeight` holds the height of the LCD used

	`0`	I	Lower boundary is automatically set to the maximum value (lower edge of the LCD)

`WCHAR* szChars`
`LPCWSTR szChars`
`ByVal chars As SecureString`
	`!= NULL`	I	Character string that is stored for the hotspot and returned when `SensorGetKeypadEntries()` is called

Return value	**Values**	**Description**	
`LONG`	`>= 0`	ID of the hotspot that was generated	
	`< 0`	Error (not STPadLibNet.dll)	

8.23.1 STPadCapt.ocx

Available from Version 8.4.3.

`SensorAddKeypadHotSpot(long nLeft, long nTop, long nWidth, long nHeight, WCHAR* szChars)`

8.23.1.1 Implementation in C#

```
int nRc = axSTPadCapt1.SensorAddKeypadHotSpot(0, 0, 0, 40, "1");
if (nRc < 0)
    MessageBox.Show(String.Format("Error {0}", nRc);
```

8.23.1.2 Implementation in Visual Basic

```
Dim nResult As Integer
nRc = AxSTPadCapt1.SensorAddKeypadHotSpot(0, 0, 0, 40, "1")
If nHotspotId < 0 Then
    MsgBox("Error " & CStr(nRc))
End If
```

8.23.2 STPadLib.dll

Available from Version 8.4.3.

```
LONG STSensorAddKeypadHotSpot(LONG nLeft, LONG nTop, LONG nWidth, LONG nHeight,
LPCWSTR szChars)
```

8.23.2.1 Implementation in C++

```
LONG nRc = STSensorAddKeypadHotSpot(0, 0, 0, 40, L"1");
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.23.3 STPadLibNet.dll

Available from Version 8.4.3.

```
long SensorAddKeypadHotSpot(long left, long top, long width, long height,
System.Security.SecureString chars)

Sub SensorAddKeypadHotSpot(ByVal left As Integer, ByVal top As Integer, ByVal
width As Integer, ByVal height As Integer, ByVal chars As
System.Security.SecureString)
```

8.23.3.1 Implementation in C#

```
SecureString keypadName = new SecureString();
keypadName.AppendChar('1');
try
{
    stPad.SensorAddKeypadHotSpot(0, 0, 0, 40, keypadName);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.23.3.2 Implementation in Visual Basic

```
Dim keypadName As SecureString = New SecureString()
keypadName.AppendChar('1')
Try
    STPad.SensorAddKeypadHotSpot(0, 0, 0, 40, keypadName)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.24 SensorGetKeypadEntries method

With this method, any number of keypad entries signalled by the `SensorHotSpotPressed()` event can be read out of the signature device in encrypted form and retrieved as contiguous text consisting of the characters stored when calling `SensorAddKeyPadHotSpot()`. The entries that are read out are deleted from the signature device.

This method works with the Sigma model from firmware 2.10, the Zeta model from firmware 1.0, the Omega model from firmware 2.14, the Gamma model from firmware 1.20 and the Delta model from firmware 1.22.

Parameter	Values	I/O	Description

`BSTR* pbstrEntries`
`out SecureString entry`
`ByRef entry As String`
	`!= NULL`	O	Character string containing the characters of the clicked keypad hotspots; the oldest entries are processed first

`LPCWSTR szEntries`	`NULL`	I	The method returns the length of the character string in the `pnStringLength` parameter
	`!= NULL`	I/O	Array in which the character string is to be written
`LONG* pnStringLength`	`>= 0`	I/O	Length of the character string incl. terminated 0 or size of the `szEntries` array in bytes

`LONG nMaxEntries`
`int maxEntries`
`ByVal maxEntries As Integer`
	`0`	I	All existing list elements are read out

	`1 - 32`	I	Maximum number of list elements to be read from the signature device
Return value	**Values**	**Description**	

`LONG`
`int`
`Integer`
	`>= 0`	Number of list items that have been read out	

	`< 0`	Error (not STPadLibNet.dll)	

8.24.1 STPadCapt.ocx

Available from Version 8.4.3.

```
LONG SensorGetKeypadEntries(BSTR* pbstrEntries, LONG nMaxEntries)
```

8.24.1.1 Implementation in C#

```
string keypadEntries = String.Empty;
int nResult = axSTPadCapt1.SensorGetKeypadEntries(ref keypadEntries, 1);
if (nResult == 0)
    MessageBox.Show("No button has been pressed since last call"));
else if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult));
```

8.24.1.2 Implementation in Visual Basic

```
Dim keypadEntries As String = String.Empty
Dim nResult As Integer
nResult = AxSTPadCapt1.SensorGetKeypadEntries(keypadEntries, 1)
ElseIf nResult = 0 Then
    MsgBox("No button has been pressed since last call")
ElseIf nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.24.2 STPadLib.dll

Available from Version 8.4.3.

```
LONG STSensorGetKeypadEntries(LPCWSTR szEntries, LONG* pnStringLength, LONG
nMaxEntries)
```

8.24.2.1 Implementation in C++

```
LONG nLen = 0;
LONG nRc = STSensorGetKeypadEntries(NULL, &nLen, 1);
if (nRc == 0)
    wprintf(L"No button has been pressed since last call");
else if (nRc > 0)
{
    WCHAR* szKeypadEntries = new WCHAR[nLen / sizeof(WCHAR)];
    nRc = STSensorGetKeypadEntries(szKeypadEntries, &nLen, 1);
    if (nRc > 0)
        // process entries...
    delete [] szKeypadEntries;
}
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.24.3 STPadLibNet.dll

Available from Version 8.4.3.

```
int SensorGetKeypadEntries(out System.Security.SecureString entry, int
maxEntries)
```

```
Function SensorGetKeypadEntries(ByRef entry As System.Security.SecureString,
ByVal maxEntries As Integer) As Integer
```

8.24.3.1 Implementation in C#

```
try
{
    SecureString keypadEntries;
    int count = stPad.SensorGetKeypadEntries(out keypadEntries, 1);
    if (count == 0)
        MessageBox.Show("No button has been pressed since last call");
    else
        // process entries...
    keypadEntries.Dispose();
{
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.24.3.2 Implementation in Visual Basic

```
Try
    Dim keypadNames As SecureString
    Dim count As Integer = STPad.SensorGetKeypadEntries(keypadNames, 1)
    If count = 0 Then
        MsgBox("No button has been pressed since last call")
    Else
        ' process entries...
    EndIf
    keypadEntries.Dispose();
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.25 SensorSetHotspotMode method

This method defines the behaviour of a monitored area (hotspot).

Parameter	Values	I/O	Description

`LONG nMode`
`HOTSPOTMODE nMode`
`HotSpotMode mode`
`ByVal mode As HotSpotMode`
	0	I	Deactivates the monitored area

	1	I	Activates the monitored area (default after calling `SensorAddHotSpot()` or `SensorAddScrollHotSpot()`, `SensorAddKeypadHotSpot()`); scrollable hotspots can only be activated during an ongoing signature capture process if they do not overlap with the defined signature window (see `SensorSetSignRect()`).
	2	I	Activates the monitored area but disables the automatic inverting when the area is clicked

`LONG nHotSpotId`
`int hotSpotId`
`ByVal hotSpotId As Integer`
	>= 0	I	ID of the hotspot that is to be changed

Return value	**Values**	**Description**	
`LONG`	0	Method was executed successfully	
	< 0	Error	

8.25.1 STPadCapt.ocx

Available from Version 8.0.16. The status described is available from Version 8.4.1.5.

```
LONG SensorSetHotSpotMode(LONG nMode, LONG nHotSpotId)
```

8.25.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.SensorSetHotspotMode(1, 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.25.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.SensorSetHotspotMode(1, 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.25.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.4.1.5.

```
LONG STSensorSetHotSpotMode(HOTSPOTMODE nMode, LONG nHotSpotId)
```

The `HOTSPOTMODE` enumeration is defined as follows:

```
kInactive = 0,
kActive = 1,
kInvertOff = 2
```

8.25.2.1 Implementation in C++

```
LONG nRc = STSensorSetHotspotMode(kActive, 0);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.25.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.4.1.5.

```
void SensorSetHotSpotMode(signotec.STPadLibNet.HotSpotMode mode, int hotSpotId)
```

```
Sub SensorSetHotspotMode(ByVal mode As signotec.STPadLibNet.HotSpotMode, ByVal
hotSpotId As Integer)
```

The `HotSpotMode` enumeration is defined as follows:

```
Inactive = 0,
Active = 1,
InvertOff = 2
```

8.25.3.1 Implementation in C#

```
try
{
    stPad.SensorSetHotspotMode(HotSpotMode.Active, 0);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.25.3.2 Implementation in Visual Basic

```
Try
    STPad.SensorSetHotspotMode(HotSpotMode.Active, 0)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.26 SensorClearHotSpots method

This method removes all monitored areas (hotspots). However, it does not delete the list of actuated keypad hotspots in the signature device so that it can still be read out.

Parameter	Values	I/O	Description
-	-	-	-
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error (not STPadLibNet.dll)	

8.26.1 STPadCapt.ocx

Available from Version 8.0.1.

```
LONG SensorClearHotSpots()
```

8.26.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.SensorClearHotSpots();
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.26.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.SensorClearHotSpots()
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.26.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STSensorClearHotSpots()
```

8.26.2.1 Implementation in C++

```
LONG nRc = STSensorClearHotSpots();
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.26.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void SensorClearHotSpots()
```

```
Sub SensorClearHotSpots()
```

8.26.3.1 Implementation in C#

```
try
{
    stPad.SensorClearHotSpots();
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.26.3.2 Implementation in Visual Basic

```
Try
    STPad.SensorClearHotSpots()
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.27 SensorClearKeypadEntries method

This method deletes the list of actuated keypad hotspots in the signature device, but does not remove the hotspots themselves. To do this, please use the `SensorClearHotSpots()` method.

Keypad hotspots work with the Sigma model from firmware 2.10, the Zeta model from firmware 1.0, the Omega model from firmware 2.14, the Gamma model from firmware 1.20 and the Delta model from firmware 1.22.

Parameter	Values	I/O	Description
–	–	-	-

Return value	Values	Description
LONG	0	Method was executed successfully
	< 0	Error

8.27.1 STPadCapt.ocx

Available from Version 8.4.3.

```
LONG SensorClearKeypadEntries()
```

8.27.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.SensorClearKeypadEntries();
```

```
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.27.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.SensorClearKeypadEntries()
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.27.2 STPadLib.dll

Available from Version 8.4.3.

```
LONG STSensorClearKeypadEntries()
```

8.27.2.1 Implementation in C++

```
LONG nRc = STSensorClearKeypadEntries();
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.27.3 STPadLibNet.dll

Available from Version 8.4.3.

```
void SensorClearKeypadEntries()
```

```
Sub SensorClearKeypadEntries()
```

8.27.3.1 Implementation in C#

```
try
{
    stPad.SensorClearKeypadEntries();
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.27.3.2 Implementation in Visual Basic

```
Try
    STPad.SensorClearKeypadEntries()
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.28 SensorStartTimer method

This method starts a Timer, which starts a defined function, if there was no interaction on the sensor of the pad for the given time periods. This Functionality is intended primarily to capture a signature without user interaction, but it can also be used to get a confirmation for a displayed text, if the belonging hotspot is not pressed for a given time period.

Parameter	Values	I/O	Description
LONG nWaitBeforeAction int waitBeforeAction ByVal waitBeforeAction As Integer	0	I	No timer waiting for the first interaction is started
	> 0	I	Maximum time to wait for the first interaction (in milliseconds) before the defined function is triggered (for example, before the start of a signature); the timer is restarted with this value after the calling of SignatureRetry().
LONG nWaitAfterAction int waitAfterAction ByVal waitAfterAction As Integer	0	I	After the first interaction no timer waiting for the next interaction is started
	> 0	I	Maximum time to wait after the last interaction was noticed in millisecond. If the given time period elapsed without a new interaction, the wanted function will be called (usually this takes places after the signing is finished).
LONG nOptions TimerOption options ByVal options As TimerOption	0	I	If the timer has expired, the SensorTimeoutOccured() event is called.
	1	I	If the time period of nWaitBeforeAction has elapsed, SignatureCancel() is called; if the time period of nWaitAfterAction has elapsed, SignatureConfirm() is called.
	2	I	If the timer has expired, SignatureCancel() is called.
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.28.1 STPadCapt.ocx

Available from Version 8.0.11. The status described is available from Version 8.3.1.

```
LONG SensorStartTimer(LONG nWaitBeforeAction, LONG nWaitAfterAction, LONG
nOptions)
```

8.28.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.SensorStartTimer(10000, 1000, 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.28.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.SensorStartTimer(10000, 1000, 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.28.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STSensorStartTimer(LONG nWaitBeforeAction, LONG nWaitAfterAction, LONG
nOptions)
```

8.28.2.1 Implementation in C++

```
LONG nRc = STSensorStartTimer(10000, 1000, 0);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.28.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void SensorStartTimer(int waitBeforeAction, int waitAfterAction,
signotec.STPadLibNet.TimerOption options)
```

```
Sub SensorStartTimer(ByVal waitBeforeAction As Integer, ByVal waitAfterAction
As Integer, ByVal options As signotec.STPadLibNet.TimerOption)
```

The `TimerOption` enumeration is defined as follows:

```
CallEvent = 0,
CallCancelOrConfirm = 1,
CallCancel = 2
```

8.28.3.1 Implementation in C#

```
try
{
    stPad.SensorStartTimer(10000, 1000, TimerOption.CallEvent);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.28.3.2 Implementation in Visual Basic

```
Try
    STPad.SensorStartTimer(10000, 1000, TimerOption.CallEvent)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.29 SensorStopTimer method

This method stops a timer started with `SensorStartTimer()` without triggering the function defined there. The method is called automatically if `SignatureConfirm()` or `SignatureCancel()` is called.

Parameter	Values	I/O	Description
–	–	-	-

Return value	Values	Description
LONG	0	Method was executed successfully (will be returned also, if no timer was set before)
	< 0	Error

8.29.1 STPadCapt.ocx

Available from Version 8.0.11.

```
LONG SensorStopTimer()
```

8.29.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.SensorStopTimer();
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.29.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.SensorStopTimer()
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.29.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STSensorStopTimer()
```

8.29.2.1 Implementation in C++

```
LONG nRc = STSensorStopTimer();
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.29.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void SensorStopTimer()
```

```
Sub SensorStopTimer()
```

8.29.3.1 Implementation in C#

```
try
{
    stPad.SensorStopTimer();
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.29.3.2 Implementation in Visual Basic

```
Try
    STPad.SensorStopTimer()
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.30 SignatureSetSecureMode method

This method places the signature device in the secure sign mode. In this mode, the biometric data can only be read using the `RSAGetSignData()` method. The `SignatureGetSignData()`

and `SignatureGetIsoData()` methods cannot be used; the `SignatureDataReceived()` event is called with 0 for all parameters.

The method must be called before `SignatureStart()` is called; the mode is retained until the device is closed.

It is possible to activate secure signature mode permanently inside the device. Please refer to your contact at signotec as required. This can be checked by attempting to deactivate it (see below)

This method only works if a public key for encrypting the biometric data is stored in the device (see also `RSASetEncryptionCert()`).

Parameter	Values	I/O	Description

`VARIANT_BOOL bEnable`
`BOOL bEnable`
`bool enable`
`ByVal enable As Boolean`
	`true`	I	The secure signature mode is activated; the call fails if no certificate for the encryption is stored on the device (see also `RSASetEncryptionCert()`).

	`false`	I	Secure signature mode is deactivated; if this mode has been activated permanently inside the device, error -36 will occur
Return value	**Values**	**Description**	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	

8.30.1 STPadCapt.ocx

Available from Version 8.0.30. The status described is available from Version 8.1.2.

```
LONG SignatureSetSecureMode(VARIANT_BOOL bEnable)
```

8.30.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.SignatureSetSecureMode(true);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.30.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.SignatureSetSecureMode(True)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.30.2 STPadLib.dll

Available from Version 8.0.30. The status described is available from Version 8.1.2.

```
LONG STSignatureSetSecureMode(BOOL bEnable)
```

8.30.2.1 Implementation in C++

```
LONG nResult = STSignatureSetSecureMode(TRUE);
if (nResult < 0)
    wprintf(L"Error %d", nResult);
```

8.30.3 STPadLibNet.dll

Available from Version 8.0.30. The status described is available from Version 8.1.2.

```
void SignatureSetSecureMode(bool enable)
```

```
Sub SignatureSetSecureMode(ByVal enable As Boolean)
```

8.30.3.1 Implementation in C#

```
try
{
    stPad.SignatureSetSecureMode(true);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.30.3.2 Implementation in Visual Basic

```
Try
    STPad.SignatureSetSecureMode(True)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.31 SignatureStart method

This method starts the signature capture process provided a connection has been opened via `DeviceOpen()`. The entire sensor is used as a writing surface provided no signature window has been defined. Signature data is only received, if a signature is actually entered on the pad. The method also sets the colour of the LED to green unless the `DeviceLedDefaultFlag` property is set to FALSE. This method automatically restores the previous content of the LCD unless this has been explicitly erased by calling `DisplayErase()`.

The method cannot be called if an active scrollable hotspot overlaps with the defined signature window (see `SensorSetSignRect()`).

If a hash 1 has been previously set using the `RSASetHash()` method, it is now transferred to the signature device. If, before this method is called, a hash 1 was generated using the `RSACreateDisplayHash()` method, the content that was used to calculate the hash is displayed (if it is not yet being displayed), and content signing is started. Outputs on the screen are no longer possible in this state. This allows the signature device to ensure that hash 1 was calculated using the screen content visible during signing. In both cases, hash 1 is inseparably linked with the signature captured subsequently. Once a signature has been captured (see also `SignatureStop()` and/or `SignatureConfirm()`) hash 1, hash 2 generated using the biometric data or the combination of both can be digitally signed using the `RSASign()` method.

Parameter	Values	I/O	Description
-	-	-	-
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.31.1 STPadCapt.ocx

Available from Version 8.0.1. The status described is available from Version 8.4.1.5.

```
LONG SignatureStart()
```

8.31.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.SignatureStart();
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.31.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.SignatureStart()
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.31.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.4.1.5.

```
LONG STSignatureStart()
```

8.31.2.1 Implementation in C++

```
LONG nRc = STSignatureStart();
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.31.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.4.1.5.

```
void SignatureStart()
```

```
Sub SignatureStart()
```

8.31.3.1 Implementation in C#

```
try
{
    stPad.SignatureStart();
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.31.3.2 Implementation in Visual Basic

```
Try
    STPad.SignatureStart()
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.32 SignatureStop method

This method terminates the signature capture process that is currently running, and caches the captured signature data. Unlike the `SignatureConfirm()` method, it does not change the display content. The `SignatureStop()` method sets the colour of the LED to yellow unless the `DeviceLedDefaultFlag` property is set to FALSE.

The hash 1, hash 2 or the combination of hash 1 and hash 2 can be signed in the signature device using the `RSASign()` method (see also `SignatureStart()`).

Parameter	Values	I/O	Description
-	-	-	-
Return value	**Values**	**Description**	
LONG int Integer	>= 0	Number of points captured	
	< 0	Error (not STPadLibNet.dll)	

8.32.1 STPadCapt.ocx

Available from Version 8.0.14. The status described is available from Version 8.0.26.

```
LONG SignatureStop()
```

8.32.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.SignatureStop();
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
else
    MessageBox.Show(String.Format("{0} points captured.", nResult);
```

8.32.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.SignatureStop()
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
Else
    MsgBox(CStr(nResult) & " points captured.")
End If
```

8.32.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.0.26.

```
LONG STSignatureStop()
```

8.32.2.1 Implementation in C++

```
LONG nRc = STSignatureStop();
if (nRc < 0)
    wprintf(L"Error %d", nRc);
else
    wprintf(L"%d points captured.", nRc);
```

8.32.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.0.26.

```
int SignatureStop()
```

```
Function SignatureStop() As Integer
```

8.32.3.1 Implementation in C#

```
try
{
    int nResult = stPad.SignatureStop();
    MessageBox.Show(String.Format("{0} points captured.", nResult));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.32.3.2 Implementation in Visual Basic

```
Try
    Dim nResult As Integer = STPad.SignatureStop()
    MsgBox(CStr(nResult) & " points captured.")
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.33 SignatureConfirm method

This methods terminates the signature capture process that is currently running (if any), caches the captured signature data and, unlike `SignatureStop()`, erases the entire LCD. The displayed signature is retained in the control element for visual inspection. To delete the signature from the control element, please use `ControlErase()`. `SignatureConfirm()` sets the colour of the LED to yellow unless the `DeviceLedDefaultFlag` property is set to FALSE.

The hash 1, hash 2 or the combination of hash 1 and hash 2 can be signed digitally in the signature device using the `RSASign()` method (see also `SignatureStart()`).

Parameter	Values	I/O	Description
-	-	-	-

Return value	Values	Description
LONG int Integer	>= 0	Number of points captured
	< 0	Error (not STPadLibNet.dll)

8.33.1 STPadCapt.ocx

Available from Version 8.0.1. The status described is available from Version 8.0.26.

```
LONG SignatureConfirm()
```

8.33.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.SignatureConfirm();
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
else
    MessageBox.Show(String.Format("{0} points captured.", nResult);
```

8.33.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.SignatureConfirm()
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
Else
    MsgBox(CStr(nResult) & " points captured.")
End If
```

8.33.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.0.26.

```
LONG STSignatureConfirm()
```

8.33.2.1 Implementation in C++

```
LONG nRc = STSignatureConfirm();
if (nRc < 0)
    wprintf(L"Error %d", nRc);
else
    wprintf(L"%d points captured.", nRc);
```

8.33.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.0.26.

```
int SignatureConfirm()
```

```
Function SignatureConfirm() As Integer
```

8.33.3.1 Implementation in C#

```
try
{
    int nResult = stPad.SignatureConfirm();
    MessageBox.Show(String.Format("{0} points captured.", nResult));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.33.3.2 Implementation in Visual Basic

```
Try
    Dim nResult As Integer = STPad.SignatureConfirm()
    MsgBox(CStr(nResult) & " points captured.")
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.34 SignatureRetry method

This method discards the signature data without ending the signature capture process, and deletes the rendered signature in the control element and on the LCD. This method will start a new capture process if the signature capture process was terminated beforehand with `SignatureStop()`.

Parameter	Values	I/O	Description
-	-	-	-
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.34.1 STPadCapt.ocx

Available from Version 8.0.1.

```
LONG SignatureRetry()
```

8.34.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.SignatureRetry();
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.34.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.SignatureRetry()
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.34.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STSignatureRetry()
```

8.34.2.1 Implementation in C++

```
LONG nRc = STSignatureRetry();
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.34.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void SignatureRetry()
```

```
Sub SignatureRetry()
```

8.34.3.1 Implementation in C#

```
try
{
    int nResult = stPad.SignatureRetry();
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.34.3.2 Implementation in Visual Basic

```
Try
    Dim nResult As Integer = STPad.SignatureRetry()
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.35 SignatureCancel method

This method ends the capture process, discards the signature data and deletes the entire LCD or just the signature. It also sets the colour of the LED to yellow unless the `DeviceLedDefaultFlag` property is set to FALSE. This method is called automatically when `DeviceClose()` is used.

Calling this method aborts content signing (see also `SignatureStart()`). The signature device can then be used as normal again.

Parameter	Values	I/O	Description

`VARIANT nErase`
`ERASEOPTION nErase`
`EraseOption erase`
`ByVal erase As EraseOption`
	0	I	The entire LCD will be deleted (Default)

	1	I	Only the signature will be deleted (Optional)
Return value	**Values**	**Description**	
`LONG`	0	Method was executed successfully	
	< 0	Error	

8.35.1 STPadCapt.ocx

Available from Version 8.0.1. The status described is available from Version 8.0.24.

`LONG SignatureCancel([optional]VARIANT nErase)`

Note: The `nErase` parameter is optional and must contain a number if it is transferred.

8.35.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.SignatureCancel();
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.35.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.SignatureCancel()
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.35.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.0.24.

```
LONG STSignatureCancel(ERASEOPTION nErase=kComplete)
```

The `ERASEOPTION` enumeration is defined as follows:

```
kComplete = 0,
kSignature = 1
```

8.35.2.1 Implementation in C++

```
LONG nRc = STSignatureCancel();
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.35.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.0.24.

```
void SignatureCancel()
```

```
void SignatureCancel(signotec.STPadLibNet.EraseOption erase)
```

```
Sub SignatureCancel()
```

```
Sub SignatureCancel(ByVal erase As signotec.STPadLibNet.EraseOption)
```

The `EraseOption` enumeration is defined as follows:

```
Complete = 0,
Signature = 1
```

8.35.3.1 Implementation in C#

```
try
{
    int nResult = stPad.SignatureCancel();
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.35.3.2 Implementation in Visual Basic

```
Try
    Dim nResult As Integer = STPad.SignatureCancel()
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.36 SignatureGetSignData method

This method returns the digitalised signature in SignData format. In order to keep the biometric data in an RSA-encrypted format, please use the `RSAGetSignData()` method.

Parameter	Values	I/O	Description
`BYTE* pbtSignData`	`NULL`	I	The method returns the required size of the array in the `pnSize` parameter.
	other	I/O	Array (in the required size) in which the SignData is written; `pnSize` must correspond to the value returned for the previous call.
`LONG* pnSize`	`> 0`	I/O	Size of the array (in bytes) in which the SignData is to be written

Return value	Values	Description
`VARIANT`	empty	Error
	other	Signature in SignData format as a byte array
`LONG`	`0`	Method was executed successfully
	`< 0`	Error
`byte[]` `Byte()`	`!= NULL`	Signature in SignData format

8.36.1 STPadCapt.ocx

Available from Version 8.0.19.

```
VARIANT SignatureGetSignData()
```

8.36.1.1 Implementation in C#

```
byte[] btSignData = (byte[])axSTPadCapt1.SignatureGetSignData();
if (btSignData == null)
{
    MessageBox.Show(String.Format("Error"));
    return;
}
```

8.36.1.2 Implementation in Visual Basic

```
Dim btSignData As Byte() = AxSTPadCapt1.SignatureGetSignData()
If btSignData Is Nothing Then
    MsgBox("Error")
    Exit Sub
End If
```

8.36.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STSignatureGetSignData(BYTE* pbtSignData, LONG* pnSize)
```

8.36.2.1 Implementation in C++

```
LONG nSize = 0;
LONG nRc = STSignatureGetSignData(NULL, &nSize);
BYTE* pbtSignData = NULL;
if (nRc == 0)
{
    pbtSignData = new BYTE[nSize];
    nRc = STSignatureGetSignData(pbtSignData, &nSize);
}
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.36.3 STPadLibNet.dll

Available from Version 8.0.19.

```
byte[] SignatureGetSignData()
```

```
Function SignatureGetSignData() As Byte()
```

8.36.3.1 Implementation in C#

```
byte[] signData;
try
{
    signData = stPad.SignatureGetSignData();
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.36.3.2 Implementation in Visual Basic

```
Dim signData As Byte()
Try
    signData = STPad.SignatureGetSignData()
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.37 SignatureGetIsoData method

This method returns the digitised signature in a standardised format in accordance with ISO/IEC 19794-7. In order to keep the biometric data in an RSA-encrypted format, please use the `RSAGetSignData()` method.

Parameter	Values	I/O	Description
`BYTE* pbtIsoData`	`NULL`	I	The method returns the required size of the array in the `pnSize` parameter.
	other	I/O	Array (in the required size) in which the ISO data is written; `pnSize` must correspond to the value returned for the previous call.
`LONG* pnSize`	`> 0`	I/O	Size of the array (in bytes) in which the ISO data is to be written

LONG nOptions IsoDataFlag options ByVal options As IsoDataFlag	Bitmask containing one or more hexadecimal values from the following list:		
	0x01	I	An 'Extended Data' block is added with data specific to signotec; if you would like more details, please speak with your contact at signotec.
VARIANT &vaCustomData BYTE* pbtCustomData byte[] customData ByVal customData As Byte()	NULL	I	None
	other	I	Additional data to be added to the 'Extended Data' block; speak with your contact at signotec for more details
LONG nCustomDataSize	>= 0	I	Size of the array referenced by pbtCustomData
LONG nCustomDataBlocks int customDataBlocks ByVal customDataBlocks As Integer	>= 0	I	Number of data blocks contained in 'Custom Data'
Return value	**Values**	**Description**	
VARIANT	empty	Error	
	other	Signature in ISO format as a byte array	
LONG	0	Method was executed successfully	
	< 0	Error	
byte[] Byte()	!= NULL	Signature in ISO format	

8.37.1 STPadCapt.ocx

Available from Version 8.0.30.

```
VARIANT SignatureGetIsoData(LONG nOptions, VARIANT &vaCustomData, LONG
nCustomDataBlocks)
```

8.37.1.1 Implementation in C#

```
byte[] btIsoData = (byte[])axSTPadCapt1.SignatureGetIsoData(0, null, 0);
if (btIsoData == null)
{
    MessageBox.Show(String.Format("Error"));
    return;
}
```

8.37.1.2 Implementation in Visual Basic

```
Dim btIsoData As Byte() = AxSTPadCapt1.SignatureGetIsoData(0, Nothing, 0)
If btIsoData Is Nothing Then
    MsgBox("Error")
    Exit Sub
End If
```

8.37.2 STPadLib.dll

Available from Version 8.0.30.

```
LONG STSignatureGetIsoData(BYTE* pbtIsoData, LONG* pnSize, LONG nOptions, BYTE*
pbtCustomData, LONG nCustomDataSize, LONG nCustomDataBlocks)
```

The following values defined in the header file can be used for the `nOptions` parameter:

```
#define STPAD_ISO_EXTENDEDDATA      0x01
```

8.37.2.1 Implementation in C++

```
LONG nSize = 0;
LONG nRc = STSignatureGetIsoData(NULL, &nSize, 0, NULL, 0, 0);
BYTE* pbtIsoData = NULL;
if (nRc == 0)
{
    pbtIsoData = new BYTE[nSize];
    nRc = STSignatureGetIsoData(pbtIsoData, &nSize, 0, NULL, 0, 0);
}
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.37.3 STPadLibNet.dll

Available from Version 8.0.19.

```
byte[] SignatureGetIsoData(IsoDataFlag options, byte[]customData, int
customDataBlocks)
```

```
Function SignatureGetIsoData(ByVal options As IsoDataFlag, ByVal customData As
Byte(), ByVal customDataBlocks As Integer) As Byte()
```

The IsoDataFlag enumeration is defined as follows:

```
None = 0x00,
ExtendedData = 0x01
```

8.37.3.1 Implementation in C#

```
byte[] btIsoData = null;
try
{
    btIsoData = stPad.SignatureGetIsoData(IsoDataFlag.None, null, 0);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.37.3.2 Implementation in Visual Basic

```
Dim btIsoData As Byte() = Nothing
Try
    btIsoData = STPad.SignatureGetIsoData(IsoDataFlag.None, Nothing, 0)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.38 SignatureSaveAsStream/SignatureSaveAsFile method

These methods are obsolete and are only included for compatibility reasons. Please use `SignatureSaveAsStreamEx()` or `SignatureSaveAsFileEx()` instead.

8.39 SignatureSaveAsStreamEx / SignatureSaveAsFileEx method

This method can be used to request a captured signature as image data in the memory or save it as an image file on the hard drive. The colour depth depends on the file type, the device used and the settings. If no further settings are made (see `nOptions` parameter), the image is created with the aspect ratio of the rectangle that surrounds the signature.

Parameter	Values	I/O	Description
`BYTE* pbtImage`	`NULL`	I	The method calculates the image, caches it and returns the required size of the array in the `pnSize` parameter.
	other	I/O	Array (in the required size) in which the cached image data is written; `pnSize` must correspond to the value returned for the previous call; all other parameters are ignored.
`LONG* pnSize`	`> 0`	I/O	Size of the array (in bytes) in which the image data is to be written

`BSTR bstrPath`
`LPCWSTR szPath`
`string path`
`ByVal path As String`
	`!= NULL`	I	Storage location for the image file as a full path that includes the file name

`LONG nResolution`
`int resolution`
`ByVal resolution As Integer`
	`>=75 <=600`	I	Resolution of the image file in pixels per inch (ppi); for the signature to be displayed in its original size, this value must be identical to the resolution of the document, in which the signature is to be integrated

`LONG nWidth`
`int width`
`ByVal width As Integer`
	`0`	I	The image will be created in original size; the `nHeight` or `height` parameter is ignored

	`> 0`	I	Maximum width of the image in pixels

`LONG nHeight`
`int height`
`ByVal height As Integer`
	`0`	I	The image will be created in original size; the `nWidth` or `width` parameter is ignored

	`> 0`	I	Maximum height of the image in pixels

LONG nFileType FILETYPE nFileType	0	I	Use TIFF with CCITT4 compression (b/w image) or LZW compression (colour image) as the file format (recommended)
	1	I	Use PNG file format
	2	I	Use BMP file format
	3	I	Use JPEG with a quality setting of 75 as the file format
	4	I	Use GIF file format (the resolution will always be 96 ppi)
	200 – 204	I	Image data is not returned as binary data, but as Base64 encoded data; otherwise as values 0–4 (only STPadCapt.ocx for the `SignatureSaveAsStreamEx()` method).
LONG nPenWidth int penWidth ByVal penWidth As Integer	< 0	I	Fixed pen width in pixels (absolute value); the pressure values are visualised by drawing in variable brightness
	0	I	A variable pen width is used that is dependent on the resolution and the pressure values
	> 0	I	Fixed pen width in pixels
OLE_COLOR clrPen COLORREF clrPen Color penColor ByVal penColor As Color	>= 0	I	Signature colour
LONG nOptions SignatureImageFlag options ByVal options As SignatureImageFlag	Bitmask containing one or more hexadecimal values from the following list:		
	0x0001	I	A visual timestamp is added to the image beneath the signature
	0x0002	I	The signature is rendered in the image that was displayed during the capture process; the image always has the aspect ratio of the display that is used, `nWidth` or `width` or `nHeight` or `height` may be ignored
	0x0004	I	The defined hotspot areas are whitened in the image (only if `0x0002` is set).
	0x0008	I	White areas at the sides of the signature are not removed; if `nWidth` or `width` and `nHeight` or `height` are greater than 0, the signature is scaled to the defined height or width depending on the aspect ratio and the image to be created has the exact size defined (only if `0x0002` is not set)
	0x0010	I	The signature will be aligned to the left (only if `0x0008` is set).
	0x0020	I	The signature will be aligned to the right (only if `0x0008` is set)
	0x0040	I	The signature will be aligned to the top (only if `0x0008` is set)
	0x0080	I	The signature will be aligned to the bottom (only if `0x0008` is set)

	0x0100	I	The timestamp size is relative to the height of the created image, not to the height of the display; this setting is useful if the signature is scaled to a given image size to make sure that the timestamp size is independent from the size of the actual signature (only if 0x0001 is set)
	0x0200	I	The signature is never smoothed independent from all other settings; this will create small files
	0x0400	I	The signature is always smoothed independent from all other settings
	0x0800	I	The image includes the overlay rectangle if displayed (only if 0x0002 is set)
	0x1000	I	White areas are stored as transparent (only if 0x0002 is not set and PNG is selected as the file format)
	0x2000	I	The current display content and not the content displayed during capture is used as the background image (only if 0x0002 is set)
	0x4000	I	The pen width will vary by the value indicated depending on the pressure values

Return value	Values	Description
VARIANT	empty	Error
	other	Image data as array of Bytes or Base64-coded String
LONG	0	Method was executed successfully
	< 0	Error
Bitmap	!= NULL	Image as System.Drawing.Bitmap

8.39.1 STPadCapt.ocx

Available from Version 8.0.14. The status described is available from Version 8.3.1.

```
VARIANT SignatureSaveAsStreamEx(LONG nResolution, LONG nWidth, LONG nHeight,
LONG nFileType, LONG nPenWidth, OLE_COLOR clrPen, LONG nOptions)
```

```
LONG SignatureSaveAsFileEx(BSTR bstrPath, LONG nResolution, LONG nWidth, LONG
nHeight, LONG nFileType, LONG nPenWidth, OLE_COLOR clrPen, LONG nOptions)
```

8.39.1.1 Implementation in C#

Work in the memory:

```
byte[] btSignature = (byte[])axSTPadCapt1.SignatureSaveAsStreamEx(300, 0,
     0, 1, 0, (uint)ColorTranslator.ToOle(Color.FromArgb(0, 0, 255)), 0);
if (btSignature == null)
{
    MessageBox.Show(String.Format("Error"));
    return;
}
MemoryStream memoryStream = new MemoryStream(btSignature);
Image image = Image.FromStream(memoryStream);
```

Work with files:

```
int nResult;
nResult = axSTPadCapt1.SignatureSaveAsFileEx(@"C:\Signature.png", 300, 0,
    0, 1, 0, (uint)ColorTranslator.ToOle(Color.FromArgb(0, 0, 255)), 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.39.1.2 Implementation in Visual Basic

Work in the memory:

```
Dim btSignature As Byte() = AxSTPadCapt1.SignatureSaveAsStreamEx(300, _
                                                0, 0, 1, 0, RGB(0, 0, 255), 0)
If btSignature Is Nothing Then
    MsgBox("Error")
    Exit Sub
End If
Dim memoryStream As MemoryStream = New MemoryStream(btSignature)
Dim image As Image = Image.FromStream(memoryStream)
```

Work with files:

```
Dim nResult As Integer
nResult = AxSTPadCapt1.SignatureSaveAsFileEx("C:\Signature.png", 300, _
                                                0, 0, 1, 0, RGB(0, 0, 255), 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.39.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.0.24.

```
LONG STSignatureSaveAsStreamEx(BYTE* pbtImage, LONG* pnSize, LONG nResolution,
LONG nWidth, LONG nHeight, FILETYPE nFileType, LONG nPenWidth, COLORREF clrPen,
LONG nOptions)
```

```
LONG STSignatureSaveAsFileEx(LPCWSTR szPath, LONG nResolution, LONG nWidth,
LONG nHeight, FILETYPE nFileType, LONG nPenWidth, COLORREF clrPen, LONG
nOptions)
```

The `FILETYPE` enumeration is defined as follows:

```
kTiff = 0,
kPng = 1,
kBmp = 2,
kJpeg = 3,
kGif = 4
```

The following values defined in the header file can be used for the `nOptions` parameter:

```
#define STPAD_SIMG_TIMESTAMP        0x0001
#define STPAD_SIMG_BACKIMAGE        0x0002
#define STPAD_SIMG_HOTSPOTS         0x0004
#define STPAD_SIMG_NOCROPPING       0x0008
#define STPAD_SIMG_ALIGNLEFT        0x0010
#define STPAD_SIMG_ALIGNRIGHT       0x0020
#define STPAD_SIMG_ALIGNTOP         0x0040
#define STPAD_SIMG_ALIGNBOTTOM      0x0080
#define STPAD_SIMG_TIMESTAMPIMGREL  0x0100
#define STPAD_SIMG_DONTSMOOTH       0x0200
#define STPAD_SIMG_SMOOTH           0x0400
#define STPAD_SIMG_OVERLAYIMAGE     0x0800
#define STPAD_SIMG_TRANSPARENT      0x1000
#define STPAD_SIMG_CURRENTIMAGES    0x2000
#define STPAD_SIMG_VARIABLEPENWIDTH 0x4000
```

8.39.2.1 Implementation in C++

Work in the memory:

```
LONG nSize = 0;
LONG nRc = STSignatureSaveAsStreamEx(NULL, &nSize, 300, 0, 0, kBmp, 0,
                                     RGB(0, 0, 255), 0);
BYTE* pbtImage = NULL;
BITMAP bitmap;
if (nRc == 0)
{
    pbtImage = new BYTE[nSize];
    nRc = STSignatureSaveAsStreamEx(pbtImage, &nSize, 300, 0, 0, kBmp, 0,
                                    RGB(0, 0, 255), 0);
}
if (nRc == 0)
{
    BITMAPFILEHEADER bmfh = (*(BITMAPFILEHEADER*)pbtImage);
    BITMAPINFO bmi = (*(BITMAPINFO*)(pbtImage +
                                     sizeof(BITMAPFILEHEADER)));
    bitmap.bmType = 0;
    bitmap.bmWidth = bmi.bmiHeader.biWidth;
    bitmap.bmHeight = bmi.bmiHeader.biHeight;
    bitmap.bmPlanes = bmi.bmiHeader.biPlanes;
    bitmap.bmBitsPixel = bmi.bmiHeader.biBitCount;
    bitmap.bmWidthBytes = ((bitmap.bmWidth * bitmap.bmBitsPixel + 31)
                           >> 5) << 2;
    bitmap.bmBits = new BYTE[bitmap.bmHeight * bitmap.bmWidthBytes];
    memcpy(bitmap.bmBits, pbtImage + bmfh.bfOffBits, bitmap.bmHeight *
           bitmap.bmWidthBytes);
    delete [] pbtImage;
}
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

Work with files:

```
LONG nRc = STSignatureSaveAsFileEx(L"C:\\Signature.png", 300, 0, 0, kPng,
                                   0, RGB(0, 0, 255), 0);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.39.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.0.24.

```
System.Drawing.Bitmap SignatureSaveAsStreamEx(int resolution, int width, int
height, int penWidth, System.Drawing.Color penColor,
signotec.STPadLibNet.SignatureImageFlag options)

void SignatureSaveAsFileEx(string path, int resolution, int width, int height,
System.Drawing.Imaging.ImageFormat fileType, int penWidth, System.Drawing.Color
penColor, signotec.STPadLibNet.SignatureImageFlag options)

Function SignatureSaveAsFileEx(ByVal resolution As Integer, ByVal width As
Integer, ByVal height As Integer, ByVal penWidth As Integer, ByVal penColor As
System.Drawing.Color, ByVal options As signotec.STPadLibNet.SignatureImageFlag)
As System.Drawing.Bitmap

Sub SignatureSaveAsFileEx(ByVal path As String, ByVal resolution As Integer,
ByVal width As Integer, ByVal height As Integer, ByVal fileType As
System.Drawing.Imaging.ImageFormat, ByVal penWidth As Integer, ByVal penColor
As System.Drawing.Color, ByVal options As
signotec.STPadLibNet.SignatureImageFlag)
```

The `SignatureImageFlag` enumeration is defined as follows:

```
None = 0x0000,
Timestamp = 0x0001,
BackImage = 0x0002,
ExcludeHotSpots = 0x0004,
DontCrop = 0x0008,
AlignLeft = 0x0010,
AlignRight = 0x0020,
AlignTop = 0x0040,
AlignBottom = 0x0080,
TimestampRelToImage = 0x0100,
DontSmooth = 0x0200,
Smooth = 0x0400,
OverlayImage = 0x0800,
TransparentBack = 0x1000,
CurrentImages = 0x2000,
VariablePenWidth = 0x4000
```

8.39.3.1 Implementation in C#

Work in the memory:

```
Bitmap bitmap;
try
{
    bitmap = stPad.SignatureSaveAsStreamEx(300, 0, 0, 0,
                    Color.FromArgb(0, 0, 255), SignatureImageFlag.None);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

Work with files:

```
try
{
    stPad.SignatureSaveAsFileEx(@"C:\Signature.png", 300, 0, 0,
                            ImageFormat.Png, 0, Color.FromArgb(0, 0, 255),
                            SignatureImageFlag.None);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.39.3.2 Implementation in Visual Basic

Work in the memory:

```
Dim bitmap As Bitmap
Try
    bitmap = STPad.SignatureSaveAsStreamEx(300, 0, 0, 0, _
                        Color.FromArgb(0, 0, 255), SignatureImageFlag.None)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

Work with files:

```
Try
    STPad.SignatureSaveAsFileEx("C:\Signature.png", 300, 0, 0, _
                            ImageFormat.Png, 0, Color.FromArgb(0, 0, 255), _
                            SignatureImageFlag.None)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.40 SignatureGetBounds method

This method delivers the coordinates of the rectangle in which the captured signature is given.

Parameter	Values	I/O	Description
SHORT* pnLeft LONG* pnLeft out int left ByRef left As Integer	>= 0	O	Left border of the signature rectangle
SHORT* pnTop LONG* pnTop out int top ByRef top As Integer	>= 0	O	Upper border of the signature rectangle

SHORT* pnRight LONG* pnRight out int right ByRef right As Integer	>= 0	O	Right border of the signature rectangle
SHORT* pnBottom LONG* pnBottom out int bottom ByRef bottom As Integer	>= 0	O	Bottom border of the signature rectangle
LONG nOptions SignatureBoundsOption options ByVal options As SignatureBoundsOption	0	I	The coordinates will be delivered relative to the size of the used LCD
	1	I	The coordinates are returned relative to the defined signature window (see SensorSetSignRect())
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.40.1 STPadCapt.ocx

Available from Version 8.0.11. The status described is available from Version 8.3.1.

```
LONG SignatureGetBounds(SHORT* pnLeft, SHORT* pnTop, SHORT* pnRight, SHORT*
pnBottom, LONG nOptions)
```

8.40.1.1 Implementation in C#

```
short nLeft, nTop, nRight, nBottom;
int nResult = axSTPadCapt1.SignatureGetBounds(ref nLeft, ref nTop, ref
                                                nRight, ref nBottom, 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error"));
else
    MessageBox.Show(String.Format("The Bounds of the Signature are: " +
                                  "{0} (left), {1} (top), {2} (right) " +
                                  "&& {3} (bottom).", nLeft, nTop,
                                  nRight, nBottom));
```

8.40.1.2 Implementation in Visual Basic

```
Dim nLeft, nTop, nRight, nBottom As Integer
Dim nResult As Integer = AxSTPadCapt1.SignatureGetBounds(nLeft, nTop, _
                                                    nRight, nBottom, 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
Else
    MsgBox("The Bounds of the Signature are:" + CStr(nLeft) + _
           " (left), " + CStr(nTop) + " (top), " + CStr(nRight) + _
           " (right) && " + CStr(nBottom) + " (bottom).")
End If
```

8.40.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STSignatureGetBounds(LONG* pnLeft, LONG* pnTop, LONG* pnRight, LONG*
pnBottom, LONG nOptions)
```

The following values defined in the header file can be used for the `nOptions` parameter:

```
#define STPAD_BOUNDS_DISPLAY   0
#define STPAD_BOUNDS_SIGNRECT  1
```

8.40.2.1 Implementation in C++

```
LONG nLeft, nTop, nRight, nBottom;
LONG nRc = STSignatureGetBounds(&nLeft, &nTop, &nRight, &nBottom,
                                STPAD_BOUNDS_DISPLAY);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
else
{
    wprintf(L"The Bounds of the Signature are: %d (left), %d (top), %d
            (right) & %d (bottom).", nLeft, nTop, nRight, nBottom);
}
```

8.40.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.3.1.

```
void SignatureGetBounds(out int left, out int top, out int right, out int
bottom, signotec.STPadLibNet.SignatureBoundsOption options)
```

```
Sub SignatureGetBounds(ByRef left As Integer, ByRef top As Integer, ByRef right
As Integer, ByRef bottom As Integer, ByVal options As
signotec.STPadLibNet.SignatureBoundsOption)
```

The `SignatureBoundsOption` enumeration is defined as follows:

```
DisplayRelative = 0,
SignRectRelative = 1
```

8.40.3.1 Implementation in C#

```
try
{
    int nLeft, nTop, nRight, nBottom;
    stPad.SignatureGetBounds(out nLeft, out nTop, out nRight, out
                        nBottom, SignatureBoundsOption.DisplayRelative);
    MessageBox.Show(String.Format("The Bounds of the Signature are: " +
                    "{0} (left), {1} (top), {2} (right) " +
                    "&& {3} (bottom).", nLeft, nTop, nRight, nBottom));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.40.3.2 Implementation in Visual Basic

```
Try
    Dim nLeft, nTop, nRight, nBottom As Integer
    STPad.SignatureGetBounds(nLeft, nTop, nRight, nBottom,
                             SignatureBoundsOption.DisplayRelative)
    MsgBox("The Bounds of the Signature are:" + CStr(nLeft) + _
           " (left), " + CStr(nTop) + " (top), " + CStr(nRight) + _
           " (right) && " + CStr(nBottom) + " (bottom).")
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.41 SignatureScaleToDisplay method

This method converts the sensor coordinates delivered by the `SignatureDataReceived()` event into display coordinates.

Parameter	Values	I/O	Description

`LONG nSensorValue`
`int sensorValue`
`ByVal sensorValue As Integer`
	>= 0	I	x or y value of a sensor coordinate

Return value	**Values**	**Description**	

`LONG`
`int`
`Integer`
	0	x or y value of a display coordinate	

	< 0	Error (not STPadLibNet.dll)	

8.41.1 STPadCapt.ocx

Available from Version 8.0.19.

`LONG SignatureScaleToDisplay(LONG nSensorValue)`

8.41.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.SignatureScaleToDisplay(1000);
if (nResult < 0)
    MessageBox.Show(String.Format("Error"));
else
    MessageBox.Show(String.Format("Display Value: {0}", nResult));
```

8.41.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.SignatureScaleToDisplay(1000)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
Else
    MsgBox("Display Value:" + CStr(nResult))
End If
```

8.41.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STSignatureScaleToDisplay(LONG nSensorValue)
```

8.41.2.1 Implementation in C++

```
LONG nRc = STSignatureScaleToDisplay(1000);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
else
    wprintf(L"Display Value: %d", nRc);
```

8.41.3 STPadLibNet.dll

Available from Version 8.0.19.

```
int SignatureScaleToDisplay(int sensorValue)
```

```
Function SignatureScaleToDisplay(ByVal sensorValue As Integer)
```

8.41.3.1 Implementation in C#

```
try
{
    MessageBox.Show(String.Format("Display Value: {0}",
                    stPad.SignatureScaleToDisplay(1000)));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.41.3.2 Implementation in Visual Basic

```
Try
    MsgBox("Display Value:" + CStr(STPad.SignatureScaleToDisplay(1000)))
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.42 DisplayErase method

This method erases both the foreground and the background memory and removes the overlay rectangle if set. Thus the entire contents of the LCD are erased. Depending on the value of the `ControlMirrorDisplay` property, this content is also erased from the control element. To erase only parts of the memory defined with `DisplaySetTarget()`, please use `DisplayEraseRect()`.

Parameter	Values	I/O	Description
-	-	-	-
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.42.1 STPadCapt.ocx

Available from Version 8.0.1.

```
LONG DisplayErase()
```

8.42.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.DisplayErase();
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.42.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.DisplayErase()
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.42.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDisplayErase()
```

8.42.2.1 Implementation in C++

```
LONG nRc = STDisplayErase();
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.42.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void DisplayErase()
```

```
Sub DisplayErase()
```

8.42.3.1 Implementation in C#

```
try
{
    stPad.DisplayErase();
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.42.3.2 Implementation in Visual Basic

```
Try
    STPad.DisplayErase()
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.43 DisplayEraseRect method

This method erases a rectangle in the memory defined with `DisplaySetTarget()`. If erasing is performed on the display and if the `ControlMirrorDisplay` property is set to 2, the rectangle is also erased in the control element.

Parameter	Values	I/O	Description
LONG nLeft int left ByVal left As Integer	>= 0	I	Left boundary; 0 is on the far left of the display
LONG nTop int top ByVal top As Integer	>= 0	I	Upper boundary; 0 is at the top of the display
LONG nWidth int width ByVal width As Integer	> 0	I	Width; DisplayWidth holds the width of the LCD used
	0	I	Right boundary is automatically set to the maximum value (right edge of the LCD)
LONG nHeight int height ByVal height As Integer	> 0	I	Height; DisplayHeight holds the height of the LCD used
	0	I	Lower boundary is automatically set to the maximum value (lower edge of the LCD)
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.43.1 STPadCapt.ocx

Available from Version 8.0.16. The status described is available from Version 8.3.1.

```
LONG DisplayEraseRect(LONG nLeft, LONG nTop, LONG nWidth, LONG nHeight)
```

8.43.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.DisplayEraseRect(10, 50, 30, 20);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.43.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.DisplayEraseRect(10, 50, 30, 20)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.43.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDisplayEraseRect(LONG nLeft, LONG nTop, LONG nWidth, LONG nHeight)
```

8.43.2.1 Implementation in C++

```
LONG nRc = STDisplayEraseRect(10, 50, 30, 20);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.43.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void DisplayEraseRect(int left, int top, int width, int height)
```

```
Sub DisplayEraseRect(ByVal left As Integer, ByVal top As Integer, ByVal width
As Integer, ByVal height As Integer)
```

8.43.3.1 Implementation in C#

```
try
{
    stPad.DisplayEraseRect(10, 50, 30, 20);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.43.3.2 Implementation in Visual Basic

```
Try
    STPad.DisplayEraseRect(10, 50, 30, 20)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.44 DisplayConfigPen method

This method sets the pen width and colour used to display a signature on the LCD. The pen width is always stored permanently in the device; the pen colour is stored permanently only on Omega devices with firmware 1.4 or later.

Parameter	Values	I/O	Description

`LONG nWidth`
`int width`
`ByVal width As Integer`
	`1 - 3`	I	Width in pixels

`OLE_COLOR clrPen`
`COLORREF clrPen`
`Color penColor`
`ByVal penColor As Color`
	`>= 0`	I	Colour; this parameter is ignored for the Sigma and Zeta model

Return value	**Values**	**Description**	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	

8.44.1 STPadCapt.ocx

Available from Version 8.0.11. The status described is available from Version 8.3.1.

```
LONG DisplayConfigPen(LONG nWidth, OLE_COLOR clrPen)
```

8.44.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.DisplayConfigPen(2,
                  (uint)ColorTranslator.ToOle(Color.FromArgb(0, 0, 255)));
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.44.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.DisplayConfigPen(2, RGB(0, 0, 255))
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.44.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDisplayConfigPen(LONG nWidth, COLORREF clrPen)
```

8.44.2.1 Implementation in C++

```
LONG nRc = STDisplayConfigPen(2, RGB(0, 0, 255));
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.44.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void DisplayConfigPen(int width, System.Drawing.Color penColor)
```

```
Sub DisplayConfigPen(ByVal width As Integer, ByVal penColor As
System.Drawing.Color)
```

8.44.3.1 Implementation in C#

```
try
{
    stPad.DisplayConfigPen(2, Color.FromArgb(0, 0, 255));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.44.3.2 Implementation in Visual Basic

```
Try
    STPad.DisplayConfigPen(2, Color.FromArgb(0, 0, 255))
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.45 DisplaySetFont method

This method permanently sets the font that is used to output text to the LCD. Text that has already been output is not modified. Arial 20 pt (Sigma and Zeta models) or 40 pt (Omega, Gamma, Delta and Alpha models) is set when `DeviceOpen()` is called.

Parameter	Values	I/O	Description
`BSTR bstrName` `LPCWSTR szName`	`!= NULL`	I	Full name of the font, which must be installed on the PC
`LONG nSize`	`12 - 200`	I	Font size
`LONG nOptions`	Bitmask containing one or more hexadecimal values from the following list:		
	`0x01`	I	Bold
	`0x02`	I	Underline
	`0x04`	I	Italic
`Font font` `ByVal font As Font`	`!= NULL`	I	Font; must be installed on the PC
Return value	**Values**	**Description**	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	

8.45.1 STPadCapt.ocx

Available from Version 8.0.1. The status described is available from Version 8.3.1.

```
LONG DisplaySetFont(BSTR bstrName, LONG nSize, LONG nOptions)
```

8.45.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.DisplaySetFont("Arial", 20, 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.45.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.DisplaySetFont("Arial", 20, 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.45.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDisplaySetFont(LPCWSTR szName, LONG nSize, LONG nOptions)
```

The following values defined in the header file can be used for the `nOptions` parameter:

```
#define STPAD_FONT_NORMAL     0x00
#define STPAD_FONT_BOLD       0x01
#define STPAD_FONT_UNDERLINE  0x02
#define STPAD_FONT_ITALIC     0x04
```

8.45.2.1 Implementation in C++

```
LONG nRc = STDisplaySetFont(L"Arial", 20, STPAD_FONT_NORMAL);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.45.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void DisplaySetFont(System.Drawing.Font font)
```

```
Sub DisplaySetFont(ByVal font As System.Drawing.Font)
```

8.45.3.1 Implementation in C#

```
try
{
    stPad.DisplaySetFont(new Font("Arial", 20, FontStyle.Regular));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.45.3.2 Implementation in Visual Basic

```
Try
    STPad.DisplaySetFont(New Font("Arial", 20, FontStyle.Regular))
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.46 DisplaySetFontColor method

This method permanently sets the colour in which the text is displayed on the LCD. Text that has already been output is not modified. The given values will be ignored, if a pad without a colour LCD is used. The colour black is set when the component is initialised.

Parameter	Values	I/O	Description

`OLE_COLOR clrFont`
`COLORREF clrFont`
`Color fontColor`
`ByVal fontColor As Color`
	`>= 0`	I	Text colour

Return value	**Values**	**Description**	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	

8.46.1 STPadCapt.ocx

Available from Version 8.0.11.

```
LONG DisplaySetFontColor(OLE_COLOR clrFont)
```

8.46.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.DisplaySetFontColor
              ((uint)ColorTranslator.ToOle(Color.FromArgb(238, 121, 0)));
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.46.1.2 Implementation in Visual Basic

```
Dim nResult As Integer
nResult = AxSTPadCapt1.DisplaySetFontColor(RGB(238, 121, 0))
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.46.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDisplaySetFontColor(COLORREF clrFont)
```

8.46.2.1 Implementation in C++

```
LONG nRc = STDisplaySetFontColor(RGB(238, 121, 0));
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.46.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void DisplaySetFontColor(System.Drawing.Color fontColor)
```

```
Sub DisplaySetFontColor(ByVal fontColor As System.Drawing.Color)
```

8.46.3.1 Implementation in C#

```
try
{
    stPad.DisplaySetFontColor(Color.FromArgb(238, 121, 0));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.46.3.2 Implementation in Visual Basic

```
Try
    STPad.DisplaySetFontColor(Color.FromArgb(238, 121, 0))
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.47 DisplaySetTarget method

This method defines the device memory that is used by the following methods and properties: `DisplayEraseRect()`, `DisplaySetText()`, `DisplaySetTextInRect()`, `DisplaySetImage()`,

DisplaySetImageFromFile(), DisplaySetImageFromStore(), DisplaySetScrollPos(), DisplayGetScrollPos(), DisplayTargetWidth and DisplayTargetHeight. The set memory remains valid until the next call of this method or of DeviceClose(). Contents stored in a non-visible memory can be displayed with the DisplaySetImageFromStore() method. For more details, see section 7.

After the calling of DeviceOpen(), the methods specified above are all executed directly on the LCD (foreground memory) as long as DisplaySetTarget() is not called.

Parameter	Values	I/O	Description
LONG nTarget DisplayTarget target ByVal target As DisplayTarget	-2	I	A permanent memory that can hold an image of the same width and double the height of the display is reserved inside the device; the memory can be used for writing from now on; if there is no permanent memory available, the return value will be 1 (see below); the value -2 is handled as -1 for the Gamma, Delta and Alpha models (see there for details)
	-1	I	A permanent memory that can hold an image in the size of the display (Omega, Gamma and Delta models) or up to a size of 2048 x 2048 pixels (Alpha model) is reserved inside the device; the memory can be used for writing from now on; if there is no permanent memory available, the return value will be 1 (see below)
	0	I	All content is displayed directly on the LCD and stored in the foreground memory; the content is lost if the device is switched off or if DisplayErase() or DeviceClose() is called
	1	I	All content is written to the non-visible background memory; the background memory is used internally during the signature process, so content is no longer available after SignatureStart() has been called; the content is also lost if the device is switched off or if DisplayErase() or DeviceClose() is called.
	2	I	All content is written to the overlay memory; it is visible immediately if an overlay rectangle has already been defined; the content is lost if the device is switched off or if DisplayErase() or DeviceClose() is called; this value cannot be used for the Sigma and Zeta models.
	1000	I	A virtual memory is used in the API; RSACreateDisplayHash() can then be called in order to transfer the content of the memory to the signature device in an optimal manner.
	other	I	Enables direct access to a permanent storage; The storage must be reserved before it can be used (see above for value -1 and -2)

Return value	Values	Description
LONG DisplayTarget	>= 0	ID of the store which is now selected; all the above referred methods will now be applied to this store; this ID can be used when calling this method again to specifically address this store
	< 0	Error (not STPadLibNet.dll)

8.47.1 STPadCapt.ocx

Available from Version 8.0.11. The status described is available from Version 8.1.0.

```
LONG DisplaySetTarget(LONG nTarget)
```

8.47.1.1 Implementation in C#

```
int nStoreId = axSTPadCapt1.DisplaySetTarget(-1);
if (nStoreId < 0)
    MessageBox.Show(String.Format("Error {0}", nStoreId);
```

8.47.1.2 Implementation in Visual Basic

```
Dim nStoreId As Integer = AxSTPadCapt1.DisplaySetTarget(-1)
If nStoreId < 0 Then
    MsgBox("Error " & CStr(nStoreId))
End If
```

8.47.2 STPadLib.dll

Available from Version 8.0.19. The described state is available as of Version 8.1.0.

```
LONG STDisplaySetTarget(LONG nTarget)
```

The following values defined in the header file can be used for the `nTarget` parameter:

```
#define STPAD_TARGET_LARGESTORE    -2
#define STPAD_TARGET_STANDARDSTORE -1
#define STPAD_TARGET_FOREGROUND     0
#define STPAD_TARGET_BACKGROUND     1
#define STPAD_TARGET_OVERLAY        2
#define STPAD_TARGET_DISPLAYHASH 1000
```

8.47.2.1 Implementation in C++

```
LONG nStoreId = STDisplaySetTarget(STPAD_TARGET_STANDARDSTORE);
if (nStoreId < 0)
    wprintf(L"Error %d", nStoreId);
```

8.47.3 STPadLibNet.dll

Available from Version 8.0.19. The described state is available as of Version 8.1.0.

```
signotec.STPadLibNet.DisplayTarget
DisplaySetTarget(signotec.STPadLibNet.DisplayTarget target)
```

```
Function DisplaySetTarget(ByVal target As signotec.STPadLibNet.DisplayTarget)
As signotec.STPadLibNet.DisplayTarget
```

The `DisplayTarget` enumeration is defined as follows:

```
NewLargeStore = -2,
NewStandardStore = -1,
ForegroundBuffer = 0,
BackgroundBuffer = 1,
OverlayBuffer = 2,
Reserved1 = 3,
Reserved2 = 4,
Reserved3 = 5,
Reserved4 = 6,
Reserved5 = 7,
Reserved6 = 8,
Reserved7 = 9,
Reserved8 = 10,
Reserved9 = 11,
Reserved10 = 12,
Reserved11 = 13,
DisplayHashBuffer = 1000
```

8.47.3.1 Implementation in C#

```
try
{
    DisplayTarget nStoreId = DisplayTarget.NewStandardStore;
    nStoreId = stPad.DisplaySetTarget(nStoreId);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.47.3.2 Implementation in Visual Basic

```
Try
    Dim nStoreId As DisplayTarget = DisplayTarget.NewStandardStore
    nStoreId = STPad.DisplaySetTarget(nStoreId)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.48 DisplaySetText method

This method can be used to write any text to the memory defined with the `DisplaySetTarget()` method. The rectangle enclosing the text overlays existing information in the memory. The text is also displayed in the control element if the output is made on the display and if the `ControlMirrorDisplay` property is set to 2. The text can also appear outside of the display and it is not wrapped unless it already contains breaks. Arial 20 pt (Sigma and Zeta models) or 40 pt (Omega, Gamma, Delta and Alpha models) is used, unless another font has been set using `DisplaySetFont()`. The font colour will be black unless another colour has been set using `DisplaySetFontColor()`.

Parameter	Values	I/O	Description

`LONG nXPos`
`int xPos`
`ByVal xPos As Integer`
	all	I	X coordinate of the starting point; 0 is on the far left of the display; `DisplayWidth` holds the point on the far right of the display

LONG nYPos int yPos ByVal yPos As Integer	all	I	Y coordinate of the starting point; 0 is at the top of the display; DisplayHeight holds the point at the very bottom of the display
LONG nAlignment ALIGN nAlignment TextAlignment alignment ByVal alignment As TextAlignment	0	I	Text is aligned to the right of the starting point
	1	I	Text is centred horizontally at the starting point
	2	I	Text is aligned to the left of the starting point
BSTR bstrText LPCWSTR szText string text ByVal text As String	!= NULL	I	Text to be output
Return value	**Values**	**Description**	
LONG int Integer	>= 0	Width of the rectangle enclosing the text	
	< 0	Error (not STPadLibNet.dll)	

8.48.1 STPadCapt.ocx

Available from Version 8.0.1. The status described is available from Version 8.3.1.

```
LONG DisplaySetText(LONG nXPos, LONG nYPos, LONG nAlignment, BSTR bstrText)
```

8.48.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.DisplaySetText(50, 20, 0, "Text");
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.48.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.DisplaySetText(50, 20, 0, "Text")
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.48.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDisplaySetText(LONG nXPos, LONG nYPos, ALIGN nAlignment, LPCWSTR szText)
```

The ALIGN enumeration is defined as follows:

```
kLeft = 0,
kCenter = 1,
kRight = 2,
kLeftCenteredVertically = 3,
kCenterCenteredVertically = 4,
kRightCenteredVertically = 5
```

8.48.2.1 Implementation in C++

```
LONG nRc = STDisplaySetText(50, 20, kLeft, L"Text");
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.48.3 STPadLibNet.dll

Available from Version 8.0.19.

```
int DisplaySetText(int xPos, int yPos, signotec.STPadLibNet.TextAlignment
alignment, string text)
```

```
Function DisplaySetText(ByVal xPos As Integer, ByVal yPos As Integer, ByVal
alignment As signotec.STPadLibNet.TextAlignment, ByVal text As String) As
Integer
```

The `TextAlignment` enumeration is defined as follows:

```
Left = 0,
Center = 1,
Right = 2,
LeftCenteredVertically = 3,
CenterCenteredVertically = 4,
RightCenteredVertically = 5
```

8.48.3.1 Implementation in C#

```
try
{
    stPad.DisplaySetText(50, 20, TextAlignment.Left, "Text");
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.48.3.2 Implementation in Visual Basic

```
Try
    STPad.DisplaySetText(50, 20, TextAlignment.Left, "Text")
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.49 DisplaySetTextInRect method

This method can be used to write any text to the memory defined with the `DisplaySetTarget()` method. The specified rectangle overlays existing information in the memory. The text is also displayed in the control element if the output is made on the display and if the `ControlMirrorDisplay` property is set to 2. The text is placed in the rectangle and can optionally be wrapped automatically. No check is made regarding whether the rectangle is within the display. Arial 20 pt (Sigma and Zeta models) or 40 pt (Omega, Gamma, Delta and Alpha models) is used, unless another font has been set using `DisplaySetFont()`. If the text is too long, the font size is automatically reduced to a minimum of 12 pt (see also options parameter). The font colour will be black unless another colour has been set using `DisplaySetFontColor()`.

Parameter	Values	I/O	Description

`LONG nLeft`
`int left`
`ByVal left As Integer`
	all	I	Left boundary; 0 is on the far left of the display

`LONG nTop`
`int top`
`ByVal top As Integer`
	all	I	Upper boundary; 0 is at the top of the display

`LONG nWidth`
`int width`
`ByVal width As Integer`
	`> 0`	I	Width; `DisplayWidth` holds the width of the LCD used

	0	I	Right boundary is automatically set to the maximum value (right margin of the LCD)

`LONG nHeight`
`int height`
`ByVal height As Integer`
	`> 0`	I	Height; `DisplayHeight` holds the height of the LCD used

	0	I	Lower boundary is automatically set to the maximum value (lower margin of the LCD)

`LONG nAlignment`
`ALIGN nAlignment`
`TextAlignment alignment`
`ByVal alignment As TextAlignment`
	0	I	Text is left-aligned and wrapped automatically

	1	I	Text is centred and wrapped automatically
	2	I	Text is right-aligned and wrapped automatically
	3	I	Text is left-aligned and centred vertically in the rectangle with no wrapping (breaks are ignored)
	4	I	Text is centred vertically and horizontally in the rectangle with no wrapping (breaks are ignored); this setting is ideal for button text
	5	I	Text is right-aligned and centred vertically in the rectangle with no wrapping (breaks are ignored)
	6	I	Text is left-aligned and not wrapped automatically (breaks are retained)
	7	I	Text is centred and not wrapped automatically (breaks are retained)
	8	I	Text is right-aligned and not wrapped automatically (breaks are retained)

`BSTR bstrText`
`LPCWSTR szText`
`string text`
`ByVal text As String`
	`!= NULL`	I	Text to be output

`VARIANT nOptions`
`LONG nOptions`
`TextFlag options`
`ByVal options As TextFlag`
	Bitmask containing one or more hexadecimal values from the following list (optional):		

	`0x01`	I	Instead of the font size, the height of the text block in pixels is returned; if the text does not fit in the given rectangle, it is not output, and the height that is necessary to output the text in the desired font size is returned; an error is returned if the longest word in the text does not fit in a line of the given rectangle.

Return value	Values	Description
LONG int Integer	>=0	The font size that is actually used or the height of the text in pixels (see also options parameter)
	< 0	Error (not STPadLibNet.dll)

8.49.1 STPadCapt.ocx

Available from Version 8.0.1. The status described is available from Version 8.4.1.9.

```
LONG DisplaySetTextInRect(LONG nLeft, LONG nTop, LONG nWidth, LONG nHeight,
LONG nAlignment, BSTR bstrText, [optional]VARIANT nOptions)
```

8.49.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.DisplaySetTextInRect(0, 0, 20, 40, 4, "Text");
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.49.1.2 Implementation in Visual Basic

```
Dim nResult As Integer
nResult =AxSTPadCapt1.DisplaySetTextInRect(0, 0, 20, 40, 4, "Text")
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.49.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.4.1.9.

```
LONG STDisplaySetTextInRect(LONG nLeft, LONG nTop, LONG nWidth, LONG nHeight,
ALIGN nAlignment, LPCWSTR szText, LONG nOptions=0)
```

The ALIGN enumeration is defined as follows:

```
kLeft = 0,
kCenter = 1,
kRight = 2,
kLeftCenteredVertically = 3,
kCenterCenteredVertically = 4,
kRightCenteredVertically = 5,
kLeftNoWrap = 6,
kCenterNoWrap = 7,
kRightNoWrap = 8
```

The following values defined in the header file can be used for the nOptions parameter:

```
#define STPAD_TEXT_NORESIZE   0x01
```

8.49.2.1 Implementation in C++

```
LONG nRc = STDisplaySetTextInRect(0, 0, 20, 40, kLeft, L"Text");
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.49.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.4.1.9.

```
int DisplaySetTextInRect(int left, int top, int width, int height,
signotec.STPadLibNet.TextAlignment alignment, string text)
```

```
int DisplaySetTextInRect(int left, int top, int width, int height,
signotec.STPadLibNet.TextAlignment alignment, string text,
signotec.STPadLibNet.TextFlag options)
```

```
Function DisplaySetTextInRect(ByVal left As Integer, ByVal top As Integer,
ByVal width As Integer, ByVal height As Integer, ByVal alignment As
signotec.STPadLibNet.TextAlignment, ByVal text As String) As Integer
```

```
Function DisplaySetTextInRect(ByVal left As Integer, ByVal top As Integer,
ByVal width As Integer, ByVal height As Integer, ByVal alignment As
signotec.STPadLibNet.TextAlignment, ByVal text As String, ByVal options As
signotec.STPadLibNet.TextFlag) As Integer
```

The `TextAlignment` enumeration is defined as follows:

```
Left = 0,
Center = 1,
Right = 2,
LeftCenteredVertically = 3,
CenterCenteredVertically = 4,
RightCenteredVertically = 5,
LeftNoWrap = 6,
CenterNoWrap = 7,
RightNoWrap = 8
```

The `TextFlag` enumeration is defined as follows:

```
None = 0x00,
NoResize = 0x01
```

8.49.3.1 Implementation in C#

```
try
{
    stPad.DisplaySetTextInRect(0, 0, 20, 40, TextAlignment.Left, "Text");
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.49.3.2 Implementation in Visual Basic

```
Try
    STPad.DisplaySetTextInRect(0, 0, 20, 40, TextAlignment.Left, "Text")
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.50 DisplaySetImage / DisplaySetImageFromFile method

This method can be used to write an image to the memory defined with the `DisplaySetTarget()` method. Although the colour depth is automatically adjusted to the

connected LCD, it is still advisable to correctly generate the image beforehand (for example, a 1-bit monochrome image is required for the Sigma and Zeta models). If there is an existing Alpha channel, it is ignored. The transfer time for the Omega, Gamma, Delta and Alpha models depends on the image material; the best pictures have few colours, so they can be compressed well. The image overlays the existing information in the memory and any signature that is present is completely erased. The text is also displayed in the control element if the output is made on the display and if the `ControlMirrorDisplay` property is set to 2. The image may also be positioned outside of the display.

Parameter	Values	I/O	Description

`LONG nXPos`
`int xPos`
`ByVal xPos As Integer`
	all	I	X coordinate of the point from which the bitmap is output to the right; 0 is on the far left of the display; `DisplayWidth` holds the point on the far right of the display

`LONG nYPos`
`int yPos`
`ByVal yPos As Integer`
	all	I	Y coordinate of the point from which the bitmap is output downwards; 0 is at the top of the display; `DisplayHeight` holds the point at the very bottom of the display

`LONG nImageHandle`
`HBITMAP hBitmap`
`Bitmap bitmap`
`ByVal bitmap As Bitmap`
	!= NULL	I	`HBITMAP` or `System.Drawing.Bitmap` to be output

`BSTR bstrPath`
`LPCWSTR szPath`
`string path`
`ByVal path As String`
	!= NULL	I	Full path or URL of the image; BMP, GIF, JPEG, PNG & TIFF can be used as file formats

Return value	**Values**	**Description**	
`LONG`	0	Method was executed successfully	
	< 0	Error	

8.50.1 STPadCapt.ocx

Available from Version 8.0.1. The status described is available from Version 8.3.1.

```
LONG DisplaySetImage(LONG nXPos, LONG nYPos, LONG nImageHandle)
```

```
LONG DisplaySetImageFromFile(LONG nXPos, LONG nYPos, BSTR bstrPath)
```

8.50.1.1 Implementation in C#

Work in the memory:

```
Bitmap bitmap = (Bitmap)Bitmap.FromFile(@"C:\Image.png");
IntPtr hBitmap = bitmap.GetHbitmap();
int nResult = axSTPadCapt1.DisplaySetImage(0, 0, hBitmap);
DeleteObject(hBitmap);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

Work with files:

```
int nResult = axSTPadCapt1.DisplaySetImageFromFile(0, 0, "C:\Image.png");
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.50.1.2 Implementation in Visual Basic

Work in the memory:

```
Dim bitmap As Bitmap = Bitmap.FromFile("C:\Image.png")
Dim hBitmap As IntPtr = bitmap.GetHbitmap
Dim nResult As Integer = AxSTPadCapt1.DisplaySetImage(0, 0, hBitmap)
DeleteObject(hBitmap)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

Work with files:

```
Dim nResult As Integer
nResult = AxSTPadCapt1.DisplaySetImageFromFile(0, 0, "C:\Image.png")
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.50.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDisplaySetImage(LONG nXPos, LONG nYPos, HBITMAP hBitmap)
```

```
LONG STDisplaySetImageFromFile(LONG nXPos, LONG nYPos, LPCWSTR szPath)
```

8.50.2.1 Implementation in C++

Work in the memory:

```
HBITMAP hBm = (HBITMAP)LoadImage(0, L"C:\\Image.bmp", IMAGE_BITMAP, 0, 0,
                                 LR_LOADFROMFILE | LR_CREATEDIBSECTION);
LONG nRc = STDisplaySetImage(0, 0, hBm);
DeleteObject(hBm);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

Work with files:

```
LONG nRc = STDisplaySetImageFromFile(0, 0, L"C:\\Image.png");
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.50.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void DisplaySetImage(int xPos, int yPos, System.Drawing.Bitmap bitmap)
```

```
void DisplaySetImageFromFile(int xPos, int yPos, string path)
```

```
Sub DisplaySetImage(ByVal xPos As Integer, ByVal yPos As Integer, ByVal bitmap
As System.Drawing.Bitmap)
```

```
Sub DisplaySetImageFromFile(ByVal xPos As Integer, ByVal yPos As Integer, ByVal
path As String)
```

8.50.3.1 Implementation in C#

Work in the memory:

```
try
{
    Bitmap bitmap = (Bitmap)Bitmap.FromFile(@"C:\Image.png");
    stPad.DisplaySetImage(0, 0, bitmap);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

Work with files:

```
try
{
    stPad.DisplaySetImageFromFile(0, 0, @"C:\Image.png");
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.50.3.2 Implementation in Visual Basic

Work in the memory:

```
Try
    Dim bitmap As Bitmap = Bitmap.FromFile("C:\Image.png")
    STPad.DisplaySetImage(0, 0, bitmap)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

Work with files:

```
Try
    STPad.DisplaySetImageFromFile(0, 0, "C:\Image.png")
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.51 DisplaySetPDF method

This method can be used to write a page of a PDF document or an excerpt thereof to the memory defined with the `DisplaySetTarget()` method. The image overlays the existing information in the memory and any signature that is present is completely erased. The text is also displayed in the control element if the output is made on the display and if the

`ControlMirrorDisplay` property is set to 2. The image may also be positioned outside of the display.

Please also note the `PDFLoad()` and `PDFSelectRect()` methods.

Parameter	Values	I/O	Description

`LONG nXPos`
`int xPos`
`ByVal xPos As Integer`
	all	I	X coordinate of the point from which the bitmap is output to the right; 0 is on the far left of the display; `DisplayWidth` holds the point on the far right of the display

`LONG nYPos`
`int yPos`
`ByVal yPos As Integer`
	all	I	Y coordinate of the point from which the bitmap is output downwards; 0 is at the top of the display; `DisplayHeight` holds the point at the very bottom of the display

`LONG nPage`
`int page`
`ByVal page As Integer`
	> 0	I	Number of the page to be output (starting at 1)

`DOUBLE dScale`
`double scale`
`ByVal scale As Double`
	> 0	I	Page is scaled; a value of 1 produces a display in original size

`LONG nOptions`
`PdfFlag options`
`ByVal options As PdfFlag`
	Bitmask containing one or more hexadecimal values from the following list:		

	0x01	I	The image transferred to the signature device is stored in the main memory of the PC and therefore does not have to be rendered again for repeated display.
Return value	**Values**	**Description**	
`LONG`	0	Method was executed successfully	
	< 0	Error	

8.51.1 STPadCapt.ocx

Available from Version 8.1.4.

`LONG DisplaySetPDF(LONG nXPos, LONG nYPos, LONG nPage, DOUBLE dScale, LONG nOptions)`

8.51.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.DisplaySetPDF(0, 0, 1, 1.0, 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.51.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.DisplaySetPDF(0, 0, 1, 1R, 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.51.2 STPadLib.dll

Available from Version 8.1.4.

```
LONG STDisplaySetPDF(LONG nXPos, LONG nYPos, LONG nPage, DOUBLE dScale, LONG
nOptions)
```

The following values defined in the header file can be used for the `nOptions` parameter:

```
#define STPAD_PDF_CACHE             0x01
```

8.51.2.1 Implementation in C++

```
LONG nRc = STDisplaySetPDF(0, 0, 1, 1., 0);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.51.3 STPadLibNet.dll

Available from Version 8.1.4.

```
void DisplaySetPDF(int xPos, int yPos, int page, double scale, PdfFlag options)
```

```
Sub DisplaySetPDF(ByVal xPos As Integer, ByVal yPos As Integer, ByVal page As
Integer, ByVal scale As Double, ByVal options As PdfFlag)
```

The `PdfFlag` enumeration is defined as follows:

```
None = 0x00,
Cache = 0x01
```

8.51.3.1 Implementation in C#

```
try
{
    stPad.DisplaySetPDF(0, 0, 1, 1., PdfFlag.None);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.51.3.2 Implementation in Visual Basic

```
Try
    STPad.DisplaySetPDF(0, 0, 1, 1R, PdfFlag.None)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.52 DisplaySetImageFromStore method

This method allows an image that has been stored in a device memory to be written to the memory defined by `DisplaySetTarget()`. The content to copy will overlay the content which is currently stored in the target storage. For more details, see section 7.

If the memory defined by `nStoreId` was not reserved beforehand by calling `DisplaySetTarget()`, the content is copied as desired; however, it is not available within the component for display in the control element or for storing with the `DisplaySaveImage…()` or

`SignatureSave…()` method. To differentiate this case from a call with a reserved `nStoreId`, `nStoreId` is returned instead of 0.

The scroll position of the source memory will be assigned to the destination memory, if both have the same size, else it will be set to 0 / 0.

The image is also displayed in the control element if the output is made to the display and if the `ControlMirrorDisplay` property is set to 2.

Parameter	Values	I/O	Description

`LONG nStoreId`
`DisplayTarget storeId`
`ByVal storeId As DisplayTarget`
	`>= 0`	I	ID of the memory from which the image is to be read; the ID is the value returned by `DisplaySetTarget()`

Return value	**Values**	**Description**	

`LONG`
`int`
`Integer`
	`> 2`	The memory defined by `nStoreId` has not been reserved beforehand; the content was successfully copied, but is not available within the component; the returned value is identical to the value of `nStoreId`	

	0	Method was executed successfully	
	`< 0`	Error (not STPadLibNet.dll)	

8.52.1 STPadCapt.ocx

Available from Version 8.0.11. The described state is available as of Version 8.0.19.

```
LONG DisplaySetImageFromStore(LONG nStoreId)
```

8.52.1.1 Implementation in C#

```
int nReturn = axSTPadCapt1.DisplaySetImageFromStore(1);
if (nReturn < 0)
    MessageBox.Show(String.Format("Error {0}", nReturn);
```

8.52.1.2 Implementation in Visual Basic

```
Dim nReturn As Integer = AxSTPadCapt1.DisplaySetImageFromStore(1)
If nReturn < 0 Then
    MsgBox("Error " & CStr(nReturn))
End If
```

8.52.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDisplaySetImageFromStore(LONG nStoreId)
```

The following values defined in the header file or the ID of a reserved, non-volatile memory can be used for the `nStoreId` parameter:

```
#define STPAD_TARGET_FOREGROUND      0
#define STPAD_TARGET_BACKGROUND      1
```

8.52.2.1 Implementation in C++

```
LONG nRc = STDisplaySetImageFromStore(STPAD_TARGET_BACKGROUND);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.52.3 STPadLibNet.dll

Available from Version 8.0.19.

```
int DisplaySetImageFromStore(signotec.STPadLibNet.DisplayTarget storeId)
```

```
Function DisplaySetImageFromStore(ByVal storeId As
signotec.STPadLibNet.DisplayTarget) As Integer
```

The `DisplayTarget` enumeration is defined as follows:

```
ForegroundBuffer = 0,
BackgroundBuffer = 1,
OverlayBuffer = 2,
Reserved1 = 3,
Reserved2 = 4,
Reserved3 = 5,
Reserved4 = 6,
Reserved5 = 7,
Reserved6 = 8,
Reserved7 = 9,
Reserved8 = 10,
Reserved9 = 11,
Reserved10 = 12,
Reserved11 = 13
```

8.52.3.1 Implementation in C#

```
try
{
    stPad.DisplaySetImageFromStore(DisplayTarget.BackgroundBuffer);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.52.3.2 Implementation in Visual Basic

```
Try
    STPad.DisplaySetImageFromStore(DisplayTarget.BackgroundBuffer);
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.53 DisplaySetOverlayRect method

This method overlays a rectangular section of the overlay memory on the content of the foreground memory. The foreground memory is covered within this rectangle until it is removed again or until `DisplayErase()`, `SignatureConfirm()` or `SignatureCancel()` is called. This functionality is ideal for a toolbar that displays hotspots, for example, to scroll.

If the storage defined with `DisplaySetTarget()` is not the foreground memory, the rectangle is not set until `DisplaySetImageFromStore()` (with the foreground memory as the destination) is called to synchronize the display.

The parameters must be multiples of eight when using Omega (with firmware 1.x) and Alpha models and are rounded if necessary.

This method cannot be called when a both a standard hotpot lying outside of the given rectangle and a scroll hotspot has been defined previously.

This method only works with the Omega, Gamma, Delta and Alpha models!

Parameter	Values	I/O	Description

`LONG nLeft`
`int left`
`ByVal left As Integer`
	>= 0	I	Left boundary; 0 is on the far left of the display

`LONG nTop`
`int top`
`ByVal top As Integer`
	>= 0	I	Upper boundary; 0 is at the top of the display

`LONG nWidth`
`int width`
`ByVal width As Integer`
	>= 8	I	Width; `DisplayWidth` holds the width of the LCD used

	0	I	The overlay rectangle is removed; the complete contents of the foreground memory will be visible again

`LONG nHeight`
`int height`
`ByVal height As Integer`
	>= 8	I	Height; `DisplayHeight` holds the height of the LCD used

	0	I	The overlay rectangle is removed; the complete contents of the foreground memory will be visible again
Return value	**Values**	**Description**	
`LONG`	0	Method was executed successfully	
	< 0	Error	

8.53.1 STPadCapt.ocx

Available from Version 8.0.17. The status described is available from Version 8.4.0.

```
LONG DisplaySetOverlayRect(LONG nLeft, LONG nTop, LONG nWidth, LONG nHeight)
```

8.53.1.1 Implementation in C#

```
int nReturn = axSTPadCapt1.DisplaySetOverlayRect(0, 400, 640, 80);
if (nReturn < 0)
    MessageBox.Show(String.Format("Error {0}", nReturn);
```

8.53.1.2 Implementation in Visual Basic

```
Dim nReturn As Integer
nReturn = AxSTPadCapt1.DisplaySetOverlayRect(0, 400, 640, 80)
If nReturn < 0 Then
    MsgBox("Error " & CStr(nReturn))
End If
```

8.53.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.4.0.

```
LONG STDisplaySetOverlayRect(LONG nLeft, LONG nTop, LONG nWidth, LONG nHeight)
```

8.53.2.1 Implementation in C++

```
LONG nRc = STDisplaySetOverlayRect(0, 400, 640, 80);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.53.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.4.0.

```
void DisplaySetOverlayRect(int left, int top, int width, int height)
```

```
Sub DisplaySetOverlayRect(ByVal left As Integer, ByVal top As Integer, ByVal
width As Integer, ByVal height As Integer)
```

8.53.3.1 Implementation in C#

```
try
{
    stPad.DisplaySetOverlayRect(0, 400, 640, 80);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.53.3.2 Implementation in Visual Basic

```
Try
    STPad.DisplaySetOverlayRect(0, 400, 640, 80);
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.54 DisplaySetScrollPos method

This method defines the X/Y position where the contents of the storage defined with `DisplaySetTarget()` will be displayed. This method only works for image memories with a size larger than the display size. Please refer to the descriptions of the properties `DisplayTargetWidth` and `DisplayTargetHeight`.

Parameter	Values	I/O	Description

`LONG nXPos`
`int xPos`
`ByVal xPos As Integer`
	`>= 0`	I	Horizontal offset of the memory contents to the left, in pixels; while the maximum possible value is calculated from `DisplayTargetWidth - DisplayWidth`, it is possible to impose limits on this value by calling the `SensorSetScrollArea()` method.

LONG nYPos int yPos ByVal yPos As Integer	>= 0	I	Vertical offset of the memory contents to the top, in pixels; the maximum possible value is calculated from DisplayTargetHeight – DisplayHeight, it is possible to impose limits on this value by calling the SensorSetScrollArea() method.
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.54.1 STPadCapt.ocx

Available from Version 8.0.17. The status described is available from Version 8.4.1.5.

```
LONG DisplaySetScrollPos(LONG nXPos, LONG nYPos)
```

8.54.1.1 Implementation in C#

```
int nReturn = axSTPadCapt1.DisplaySetScrollPos(0, 100);
if (nReturn < 0)
    MessageBox.Show(String.Format("Error {0}", nReturn);
```

8.54.1.2 Implementation in Visual Basic

```
Dim nReturn As Integer = AxSTPadCapt1.DisplaySetScrollPos(0, 100)
If nReturn < 0 Then
    MsgBox("Error " & CStr(nReturn))
End If
```

8.54.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.4.1.5.

```
LONG STDisplaySetScrollPos(LONG nXPos, LONG nYPos)
```

8.54.2.1 Implementation in C++

```
LONG nRc = STDisplaySetScrollPos(0, 100);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.54.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.4.1.5.

```
void DisplaySetScrollPos(int xPos, int yPos)
```

```
Sub DisplaySetScrollPos(ByVal xPos As Integer, ByVal yPos As Integer)
```

8.54.3.1 Implementation in C#

```
try
{
    stPad.DisplaySetScrollPos(0, 100);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.54.3.2 Implementation in Visual Basic

```
Try
    STPad.DisplaySetScrollPos(0, 100)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.55 DisplayGetScrollPos method

This method returns the X/Y position where the content of the memory defined using the `DisplaySetTarget()` method are output on the screen.

Parameter	Values	I/O	Description

`LONG* pnXPos`
`out int xPos`
`ByRef xPos As Integer`
	`!= NULL`	O	Horizontal offset of the memory contents to the left, in pixels

`LONG* pnYPos`
`out int yPos`
`ByRef yPos As Integer`
	`!= NULL`	O	Vertical offset of the memory contents to the top, in pixels

Return value	**Values**	**Description**	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	

8.55.1 STPadCapt.ocx

Available from Version 8.0.17.

```
LONG DisplayGetScrollPos(LONG* pnXPos, LONG* pnYPos)
```

8.55.1.1 Implementation in C#

```
int nXPos, nYPos;
int nReturn = axSTPadCapt1.DisplayGetScrollPos(ref nXPos, ref nYPos);
if (nReturn < 0)
    MessageBox.Show(String.Format("Error {0}", nReturn);
else
    MessageBox.Show(String.Format("Scroll pos: {0} / {1}", nXpos, nYPos);
```

8.55.1.2 Implementation in Visual Basic

```
Dim nXPos, nYPos As Integer
Dim nReturn As Integer = AxSTPadCapt1.DisplayGetScrollPos(nXpos, nYPos)
If nReturn < 0 Then
    MsgBox("Error " & CStr(nReturn))
Else
    MsgBox("Scroll pos: " & CStr(nXPos) & " / " & CStr(nYPos))
End If
```

8.55.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDisplayGetScrollPos(LONG* pnXPos, LONG* pnYPos)
```

8.55.2.1 Implementation in C++

```
LONG nXPos, nYPos;
LONG nRc = STDisplayGetScrollPos(&nXPos, &nYPos);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
else
    wprintf(L"Scroll pos: %d / %d", nXPos, nYPos);
```

8.55.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.3.1.

```
void DisplayGetScrollPos(out int xPos, out int yPos)
```

```
Sub DisplayGetScrollPos(ByRef xPos As Integer, ByRef yPos As Integer)
```

8.55.3.1 Implementation in C#

```
try
{
    int nXPos, nYPos;
    stPad.DisplayGetScrollPos(out nXPos, out nYPos);
    MessageBox.Show(String.Format("Scroll pos: {0} / {1}", nXPos,
                                  nYPos));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.55.3.2 Implementation in Visual Basic

```
Try
    Dim nXPos, nYPos As Integer
    STPad.DisplayGetScrollPos(nXpos, nYPos)
    MsgBox("Scroll pos: " & CStr(nXPos) & " / " & CStr(nYPos))
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.56 DisplaySaveImageAsStream / DisplaySaveImageAsFile method

This method can be used to retrieve the content of an image memory as image data in the memory or save it as an image file on the hard drive. Any existing signature will be ignored for saving. The image has the size and resolution of the screen on the device used and, depending on the option, the size of the screen or image memory. The colour depth depends on the file type and the device used.

Parameter	Values	I/O	Description
`BYTE* pbtImage`	`NULL`	I	The method calculates the image, caches it and returns the required size of the array in the `pnSize` parameter.
	other	I/O	Array (in the required size) in which the cached image data is written; `pnSize` must correspond to the value returned for the previous call; all other parameters are ignored.
`LONG* pnSize`	`> 0`	I/O	Size of the array (in bytes) in which the image data is to be written

`BSTR bstrPath`
`LPCWSTR szPath`
`string path`
`ByVal path As String`
	`!= NULL`	I	Storage location for the image file as a full path that includes the file name

`LONG nFileType` `FILETYPE nFileType`	`0`	I	Use TIFF with CCITT4 compression (b/w image) or LZW compression (colour image) as the file format (recommended)
	`1`	I	Use PNG file format
	`2`	I	Use BMP file format
	`3`	I	Use JPEG with a quality setting of 75 as the file format
	`4`	I	Use GIF file format (the resolution will always be 96 ppi)
	`200 – 204`	I	Image data is not returned as binary data, but as Base64 encoded data; otherwise as values 0–4 (STPadCapt.ocx and `DisplaySaveImageAsStream()` only)

`LONG nOptions`
`DisplayImageFlag options`
`ByVal options As DisplayImageFlag`
	Bitmask containing one or more hexadecimal values from the following list:		

	`0x01`	I	The whole content of the display will be stored; The hotspot areas (buttons) stay white in this mode.
	`0x02`	I	Instead of the current display content the content of the whole foreground memory without the overlay rectangle is saved
	`0x04`	I	Instead of the current display content, the content of the entire memory defined beforehand with `DisplaySetTarget()` is saved.

Return value	Values	Description
VARIANT	empty	Error
	other	Image data as array of Bytes or Base64-coded String
LONG	0	Method was executed successfully
	< 0	Error
Bitmap	!= NULL	Image as System.Drawing.Bitmap

8.56.1 STPadCapt.ocx

Available from Version 8.0.11. The status described is available from Version 8.3.2.

```
LONG DisplaySaveImageAsFile(BSTR bstrPath, LONG nFileType, LONG nOptions)
VARIANT DisplaySaveImageAsStream(LONG nFileType, LONG nOptions)
```

8.56.1.1 Implementation in C#

Work in the memory:

```
byte[] btImage = (byte[])axSTPadCapt1.DisplaySaveImageAsStream(1, 0);
if (btImage == null)
{
    MessageBox.Show(String.Format("Error"));
    return;
}
MemoryStream memoryStream = new MemoryStream(btImage);
Image image = Image.FromStream(memoryStream);
```

Work with files:

```
int nResult = axSTPadCapt1.DisplaySaveImageAsFile(@"C:\Image.png", 1, 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.56.1.2 Implementation in Visual Basic

Work in the memory:

```
Dim btImage As Byte() = AxSTPadCapt1.DisplaySaveImageAsStream(1, 0)
If btImage Is Nothing Then
    MsgBox("Error")
    Exit Sub
End If
Dim memoryStream As MemoryStream = New MemoryStream(btImage)
Dim image As Image = Image.FromStream(memoryStream)
```

Work with files:

```
Dim nResult As Integer
nResult = AxSTPadCapt1.DisplaySaveImageAsFile("C:\Image.png", 1, 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.56.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.3.2.

```
LONG STDisplaySaveImageAsFile(LPCWSTR szPath, FILETYPE nFileType, LONG
nOptions)

LONG STDisplaySaveImageAsStream(BYTE* pbtImage, LONG* pnSize, FILETYPE
nFileType, LONG nOptions)
```

The `FILETYPE` enumeration is defined as follows:

```
kTiff = 0,
kPng = 1,
kBmp = 2,
kJpeg = 3,
kGif = 4
```

The following values defined in the header file can be used for the `nOptions` parameter:

```
#define STPAD_DIMG_HOTSPOTS         0x01
#define STPAD_DIMG_BUFFER           0x02
#define STPAD_DIMG_CURRENTTARGET    0x04
```

8.56.2.1 Implementation in C++

Work in the memory:

```
LONG nSize = 0;
LONG nRc = STDisplaySaveImageAsStream(NULL, &nSize, kBmp, 0);
BYTE* pbtImage = NULL;
BITMAP bitmap;
if (nRc == 0)
{
    pbtImage = new BYTE[nSize];
    nRc = STDisplaySaveImageAsStream(pbtImage, &nSize, kBmp, 0);
}
if (nRc == 0)
{
    BITMAPFILEHEADER bmfh = (*(BITMAPFILEHEADER*)pbtImage);
    BITMAPINFO bmi = (*(BITMAPINFO*)(pbtImage +
                                     sizeof(BITMAPFILEHEADER)));
    bitmap.bmType = 0;
    bitmap.bmWidth = bmi.bmiHeader.biWidth;
    bitmap.bmHeight = bmi.bmiHeader.biHeight;
    bitmap.bmPlanes = bmi.bmiHeader.biPlanes;
    bitmap.bmBitsPixel = bmi.bmiHeader.biBitCount;
    bitmap.bmWidthBytes = ((bitmap.bmWidth * bitmap.bmBitsPixel + 31)
                           >> 5) << 2;
    bitmap.bmBits = new BYTE[bitmap.bmHeight * bitmap.bmWidthBytes];
    memcpy(bitmap.bmBits, pbtImage + bmfh.bfOffBits, bitmap.bmHeight *
           bitmap.bmWidthBytes);
    delete [] pbtImage;
}
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

Work with files:

```
LONG nRc = STDisplaySaveImageAsFile(L"C:\\Image.png", kPng, 0);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.56.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.3.2.

```
void DisplaySaveImageAsFile(string path, System.Drawing.Imaging.ImageFormat
fileType, signotec.STPadLibNet.DisplayImageFlag options)

System.Drawing.Bitmap
DisplaySaveImageAsStream(signotec.STPadLibNet.DisplayImageFlag options)

Sub DisplaySaveImageAsFile(ByVal path As String, ByVal fileType As
System.Drawing.Imaging.ImageFormat, ByVal options As
signotec.STPadLibNet.DisplayImageFlag)

Function DisplaySaveImageAsStream(ByVal options As
signotec.STPadLibNet.DisplayImageFlag) As System.Drawing.Bitmap
```

The `DisplayImageFlag` enumeration is defined as follows:

```
None = 0x00,
ExcludeHotSpots = 0x01,
CompleteBuffer = 0x02,
CurrentTarget = 0x04
```

8.56.3.1 Implementation in C#

Work in the memory:

```
Bitmap bitmap;
try
{
    bitmap = stPad.DisplaySaveImageAsStream(DisplayImageFlag.None);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

Work with files:

```
try
{
    stPad.DisplaySaveImageAsFile(@"C:\Image.png", ImageFormat.Png,
                                 DisplayImageFlag.None);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.56.3.2 Implementation in Visual Basic

Work in the memory:

```
Dim bitmap As Bitmap
Try
    bitmap = STPad.DisplaySaveImageAsStream(DisplayImageFlag.None)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

Work with files:

```
Try
    STPad.DisplaySaveImageAsFile("C:\Image.png", ImageFormat.Png, _
                                 DisplayImageFlag.None)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.57 DisplaySetStandbyImage / DisplaySetStandbyImageFromFile method

This method permanently stores an image in the selected device. The image is automatically displayed when a connection to the device has not yet been opened (while a connection is being established, for example). Although the colour depth is automatically adjusted to the connected LCD, it is still advisable to correctly generate the image beforehand (for example, a 1-bit monochrome image is required for the Sigma and Zeta models). If there is an existing Alpha channel, it is ignored. If the image is too small, it is centred. If the image is too large, it is cropped on the right and at the bottom.

The image is only transmitted when the memory management determines that the image is not yet stored in the device. A slide show configuration is removed by calling this method.

Parameter	Values	I/O	Description
`LONG nImageHandle`	`NULL`	I	The stored image is deleted

`HBITMAP hBitmap`
`Bitmap bitmap`
`ByVal bitmap As Bitmap`
	`!= NULL`	I	Handle of the image or `System.Drawing.Bitmap` to be stored

`BSTR bstrPath`
`LPCWSTR szPath`
`string path`
`ByVal path As String`
	`!= NULL`	I	Full path or URL of the image; BMP, GIF, JPEG, PNG & TIFF can be used as file formats

Return value	**Values**	**Description**	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	

8.57.1 STPadCapt.ocx

Available from Version 8.0.3.

```
LONG DisplaySetStandbyImage(LONG nImageHandle)
LONG DisplaySetStandbyImageFromFile(BSTR bstrPath)
```

8.57.1.1 Implementation in C#

Work in the memory:

```
Bitmap bitmap = (Bitmap)Bitmap.FromFile(@"C:\Image.png");
IntPtr hBitmap = bitmap.GetHbitmap();
int nResult = axSTPadCapt1.DisplaySetStandbyImage(hBitmap);
DeleteObject(hBitmap);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

Work with files:

```
int nResult;
nResult = axSTPadCapt1.DisplaySetStandbyImageFromFile(@"C:\Image.png");
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.57.1.2 Implementation in Visual Basic

Work in the memory:

```
Dim bitmap As Bitmap = Bitmap.FromFile("C:\Image.png")
Dim hBitmap As IntPtr = bitmap.GetHbitmap
Dim nResult As Integer = AxSTPadCapt1.DisplaySetStandbyImage(hBitmap)
DeleteObject(hBitmap)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

Work with files:

```
Dim nResult As Integer
nResult = AxSTPadCapt1.DisplaySetStandbyImageFromFile("C:\Image.png")
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.57.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDisplaySetStandbyImage(HBITMAP hBitmap)
```

```
LONG STDisplaySetStandbyImageFromFile(LPCWSTR szPath)
```

8.57.2.1 Implementation in C++

Work in the memory:

```
HBITMAP hBm = (HBITMAP)LoadImage(0, L"C:\\Image.bmp", IMAGE_BITMAP, 0, 0,
                                 LR_LOADFROMFILE | LR_CREATEDIBSECTION);
LONG nRc = STDisplaySetStandbyImage(hBm);
DeleteObject(hBm);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

Work with files:

```
LONG nRc = STDisplaySetStandbyImageFromFile(L"C:\\Image.png");
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.57.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void DisplaySetStandbyImage(System.Drawing.Bitmap bitmap)
void DisplaySetStandbyImageFromFile(string path)
Sub DisplaySetStandbyImage(ByVal bitmap As System.Drawing.Bitmap)
Sub DisplaySetStandbyImageFromFile(ByVal path As String)
```

8.57.3.1 Implementation in C#

Work in the memory:

```
try
{
    Bitmap bitmap = (Bitmap)Bitmap.FromFile(@"C:\Image.png");
    stPad.DisplaySetStandbyImage(bitmap);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

Work with files:

```
try
{
    stPad.DisplaySetStandbyImageFromFile(@"C:\Image.png");
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.57.3.2 Implementation in Visual Basic

Work in the memory:

```
Try
    Dim bitmap As Bitmap = Bitmap.FromFile("C:\Image.png")
    STPad.DisplaySetStandbyImage(bitmap)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

Work with files:

```
Try
    STPad.DisplaySetStandbyImageFromFile("C:\Image.png")
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.58 DisplaySetStandbyImageEx / DisplaySetStandbyImageFromFileEx method

This method permanently stores an image in the selected device. The image is automatically displayed when a connection to the device has not yet been opened (while a connection is being established, for example). Unlike `DisplaySetStandbyImage()` and `DisplaySetStandbyImageFromFile()`, the backlight is switched off and the image is removed from the display after a previously specified time period has expired.

The colour depth of the image is automatically adjusted to suit the connected LCD. Although, it is still advisable to correctly generate the image beforehand (for example, a 1-bit monochrome image is required for the Sigma and Zeta models). If there is an existing Alpha channel, it is ignored. If the image is too small, it is centred. If the image is too large, it is cropped on the right and at the bottom.

The image is only transmitted when the memory management determines that the image is not yet stored in the device.

A slide show configuration is removed by calling this method.

The time-controlled switching off is not supported by the models Sigma, Omega up to firmware 1.40 or Alpha. The time value is ignored in these devices.

Parameter	Values	I/O	Description

`LONG nImageHandle`
`HBITMAP hBitmap`
`Bitmap bitmap`
`ByVal bitmap As Bitmap`
	`NULL`	I	The stored image is deleted

	`!= NULL`	I	Handle of the image or `System.Drawing.Bitmap` to be stored

`BSTR bstrPath`
`LPCWSTR szPath`
`string path`
`ByVal path As String`
	`!= NULL`	I	Full path or URL of the image; BMP, GIF, JPEG, PNG & TIFF can be used as file formats

`LONG nDuration`
`int duration`
`ByVal duration As Integer`
	`0, 100 – 300000`	I	Time in milliseconds that the image is displayed until the display is switched off; a maximum value of `255000` can be transmitted for the Gamma and Delta models. If the value is 0, no switching off will occur.

Return value	**Values**	**Description**	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	

8.58.1 STPadCapt.ocx

Available from Version 8.0.3.

```
LONG DisplaySetStandbyImageEx(LONG nImageHandle, LONG nDuration)
```

```
LONG DisplaySetStandbyImageFromFileEx(BSTR bstrPath, LONG nDuration)
```

8.58.1.1 Implementation in C#

Work in the memory:

```
Bitmap bitmap = (Bitmap)Bitmap.FromFile(@"C:\Image.png");
IntPtr hBitmap = bitmap.GetHbitmap();
int nResult = axSTPadCapt1.DisplaySetStandbyImageEx(hBitmap, 5000);
DeleteObject(hBitmap);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

Work with files:

```
int nResult;
nResult = axSTPadCapt1.DisplaySetStandbyImageFromFileEx(@"C:\Image.png",
                                                        5000);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.58.1.2 Implementation in Visual Basic

Work in the memory:

```
Dim bitmap As Bitmap = Bitmap.FromFile("C:\Image.png")
Dim hBitmap As IntPtr = bitmap.GetHbitmap
Dim nResult As Integer = AxSTPadCapt1.DisplaySetStandbyImageEx(hBitmap,
                                                               5000)
DeleteObject(hBitmap)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

Work with files:

```
Dim nResult As Integer
nResult = AxSTPadCapt1.DisplaySetStandbyImageFromFileEx("C:\Image.png",
                                                        5000)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.58.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDisplaySetStandbyImageEx(HBITMAP hBitmap, LONG nDuration)
```

```
LONG STDisplaySetStandbyImageFromFileEx(LPCWSTR szPath, LONG nDuration)
```

8.58.2.1 Implementation in C++

Work in the memory:

```
HBITMAP hBm = (HBITMAP)LoadImage(0, L"C:\\Image.bmp", IMAGE_BITMAP, 0, 0,
                                 LR_LOADFROMFILE | LR_CREATEDIBSECTION);
LONG nRc = STDisplaySetStandbyImageEx(hBm, 5000);
DeleteObject(hBm);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

Work with files:

```
LONG nRc = STDisplaySetStandbyImageFromFileEx(L"C:\\Image.png", 5000);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.58.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void DisplaySetStandbyImageEx(System.Drawing.Bitmap bitmap, int duration)
```

```
void DisplaySetStandbyImageFromFileEx(string path, int duration)
```

```
Sub DisplaySetStandbyImageEx(ByVal bitmap As System.Drawing.Bitmap, ByVal
duration As Integer)
```

```
Sub DisplaySetStandbyImageFromFileEx(ByVal path As String, ByVal duration As
Integer)
```

8.58.3.1 Implementation in C#

Work in the memory:

```
try
{
    Bitmap bitmap = (Bitmap)Bitmap.FromFile(@"C:\Image.png");
    stPad.DisplaySetStandbyImageEx(bitmap, 5000);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

Work with files:

```
try
{
    stPad.DisplaySetStandbyImageFromFileEx(@"C:\Image.png", 5000);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.58.3.2 Implementation in Visual Basic

Work in the memory:

```
Try
    Dim bitmap As Bitmap = Bitmap.FromFile("C:\Image.png")
    STPad.DisplaySetStandbyImageEx(bitmap, 5000)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

Work with files:

```
Try
    STPad.DisplaySetStandbyImageFromFileEx("C:\Image.png", 5000)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.59 DisplayConfigSlideShow method

With this method, a slide show of permanently stored images can be configured to be played automatically on the target device, if the device is not in use. A possibly saved standby image is removed.

This method only works with the Omega, Gamma, Delta and Alpha models. It is necessary to use the `DisplayConfigSlideShowEx()` method in order to fully exploit the functionality of the Gamma and Delta models.

Parameter	Values	I/O	Description

`BSTR bstrSlideList`
`LPCWSTR szSlideList`
`string slideList`
`ByVal slideList As String`
	`NULL`, `""`	I	The slide show will be disabled

	other	I	A list of up to 16 (Gamma up to firmware 1.9 and Delta up to firmware 1.7) or 32 (Omega, Gamma from firmware 1.10, Delta from firmware 1.8 and Alpha) IDs of image stores separated by semicolons; these IDs must be reserved previously by the `DisplaySetTarget()` method and must be filled with text or images; an ID of an image store can be included multiple times; the ID 35 cannot be used for a slide show; the slide show will be displayed in the given order.

`LONG nDuration`
`int duration`
`ByVal duration As Integer`
	`100 - 300000`	I	Time in milliseconds that each image is displayed; a maximum value of `255000` can be passed for the Gamma and Delta models.

Return value	**Values**	**Description**	

`LONG`
`int`
`Integer`
	`>= 0`	Number of images in the slide show	

	`< 0`	Error (not STPadLibNet.dll)	

8.59.1 STPadCapt.ocx

Available from Version 8.0.11. The status described is available from Version 8.4.1.10.

`LONG DisplayConfigSlideShow(BSTR bstrSlideList, LONG nDuration)`

8.59.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.DisplayConfigSlideShow("5;6;8;5;7", 2000);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.59.1.2 Implementation in Visual Basic

```
Dim nResult As Integer
nResult = AxSTPadCapt1.DisplayConfigSlideShow("5;6;8;5;7", 2000)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.59.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.4.1.10.

```
LONG STDisplayConfigSlideShow(LPCWSTR szSlideList, LONG nDuration)
```

8.59.2.1 Implementation in C++

```
LONG nRc = STDisplayConfigSlideShow(L"5;6;8;5;7", 2000);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.59.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.4.1.10.

```
int DisplayConfigSlideShow(string slideList, int duration)
```

```
Function DisplayConfigSlideShow(ByVal slideList As String, ByVal duration As Integer)
```

8.59.3.1 Implementation in C#

```
try
{
    stPad.DisplayConfigSlideShow("5;6;8;5;7", 2000);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.59.3.2 Implementation in Visual Basic

```
Try
    STPad.DisplayConfigSlideShow("5;6;8;5;7", 2000);
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.60 DisplayConfigSlideShowEx method

With this method, a slide show of permanently stored images can be configured to be played automatically on the target device, if the device is not in use. A possibly saved standby image is removed.

This method only offers the functionality of the `DisplayConfigSlideShow()` method with the Omega (up to firmware 1.40) and Alpha models.

Parameter	Values	I/O	Description

`BSTR bstrSlideList`
`LPCWSTR szSlideList`
`string slideList`
`ByVal slideList As String`
	`NULL, ""`	I	The slide show will be disabled

	other	I	A list of up to 16 (Gamma up to firmware 1.9 and Delta up to firmware 1.7) or 32 (Omega, Gamma from firmware 1.10, Delta from firmware 1.8 and Alpha) IDs of image stores separated by semicolons; these IDs must have been previously reserved through the `DisplaySetTarget()` method and must have been filled with text or images; an ID of an image store can be included multiple times; the ID 35 cannot be used for a slide show; the slide show will be displayed in the given order; a negative number can also be included when using the Omega (from firmware 2.0), Gamma and Delta models, the image from the Store ID that corresponds to the absolute value of this number will then only be displayed during the first run of the slide show; a value of 0 can also be included with the Omega (from firmware 2.0), Gamma and Delta models, the slide show will then end at this point and the backlight will be disabled.

`BSTR bstrDurationList`
`LPCWSTR szDurationList`
`string durationList`
`ByVal durationList As String`
	`100 - 300000`	I	A list of a maximum of one (Omega up to firmware 1.40 and Alpha) or 16 (Omega from firmware 2.0, Gamma up to firmware 1.9 and Delta up to firmware 1.7) or 32 (Gamma from firmware 1.10 and Delta from firmware 1.8) times in milliseconds separated by semicolon for which each individual image is to be displayed; if this list contains fewer values than the list of Store IDs, the last value in the list is used for all further images; permitted values are all those from `"100"` to `"255000"` for the Gamma and Delta models and `"300000"` for all other models.

Return value	**Values**	**Description**	

`LONG`
`int`
`Integer`
	`>= 0`	Number of images in the slide show	

	`< 0`	Error (not STPadLibNet.dll)	

8.60.1 STPadCapt.ocx

Available from Version 8.2.0. The status described is available from Version 8.4.1.10.

```
LONG DisplayConfigSlideShowEx(BSTR bstrSlideList, BSTR bstrDurationList)
```

8.60.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.DisplayConfigSlideShowEx("-5;6;8;6;7;0",
                                                    "5000;1000;2000;1000;2000");
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.60.1.2 Implementation in Visual Basic

```
Dim nResult As Integer
nResult = AxSTPadCapt1.DisplayConfigSlideShowEx("-5;6;8;6;7;0",
                                                "5000;1000;2000;1000;2000")
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.60.2 STPadLib.dll

Available from Version 8.2.0. The status described is available from Version 8.4.1.10.

```
LONG STDisplayConfigSlideShowEx(LPCWSTR szSlideList, LPCWSTR szDurationList)
```

8.60.2.1 Implementation in C++

```
LONG nRc = STDisplayConfigSlideShowEx(L"-5;6;8;6;7;0",
                                      L"5000;1000;2000;1000;2000");
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.60.3 STPadLibNet.dll

Available from Version 8.2.0. The status described is available from Version 8.4.1.10.

```
int DisplayConfigSlideShowEx(string slideList, string durationList)
```

```
Function DisplayConfigSlideShowEx(ByVal slideList As String, ByVal durationList
As String) As Integer
```

8.60.3.1 Implementation in C#

```
try
{
    stPad.DisplayConfigSlideShowEx("-5;6;8;6;7;0",
                                   "5000;1000;2000;1000;2000");
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.60.3.2 Implementation in Visual Basic

```
Try
    STPad.DisplayConfigSlideShowEx("-5;6;8;6;7;0",
                                   "5000;1000;2000;1000;2000");
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.61 DisplayGetStandbyId method

This method returns the number of images configured for standby operation, as well as a hexadecimal character string that identifies the standby image currently set or the slide show currently configured. Thus it can be checked, for example, whether the current configuration matches the desired one.

Parameter	Values	I/O	Description
BSTR* pbstrId out string id ByRef id As String	!= NULL	O	Unique string the identifies the current configuration
LPCWSTR szId	NULL	I	The method returns the length of the character string in the pnStringLength parameter
	!= NULL	I/O	Array in which the character string that identifies the current configuration is written; if the array is too small, the end characters are cut off
LONG* pnStringLength	>= 0	I/O	Length of the character string incl. terminated 0 or size of the szId array in bytes
Return value	**Values**	**Description**	
LONG int Integer	>= 0	Number of reserved permanent stores used for the standby image or the slide show	
	< 0	Error	

8.61.1 STPadCapt.ocx

Available from Version 8.0.16. The status described is available from Version 8.2.0.

```
LONG DisplayGetStandbyId(BSTR* pbstrId)
```

8.61.1.1 Implementation in C#

```
string strId = "";
int nResult = axSTPadCapt1.DisplayGetStandbyId(ref strId);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult));
else if (nResult == 0)
    MessageBox.Show(String.Format("No standby mode configured!"));
else
    MessageBox.Show(String.Format("{0} stores configured, ID is: {1}",
                                  nResult, strId));
```

8.61.1.2 Implementation in Visual Basic

```
Dim strId As String = ""
Dim nResult As Integer = AxSTPadCapt1.DisplayGetStandbyId(strId)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
ElseIf nResult = 0 Then
    MsgBox("No standby mode configured!")
Else
    MsgBox(CStr(nResult) & " stores configured, ID is: " & strId)
End If
```

8.61.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.4.1.4.

```
LONG STDisplayGetStandbyId(LPCWSTR szId, LONG* pnStringLength)
```

8.61.2.1 Implementation in C++

```
LONG nLen = 0;
LONG nRc = STDisplayGetStandbyId(NULL, &nLen);
if (nRc == 0)
    wprintf(L"No standby mode configured!");
else if (nRc > 0)
{
    WCHAR* szId = new WCHAR[nLen / sizeof(WCHAR)];
    nRc = STDisplayGetStandbyId(szId, &nLen);
    if (nRc > 0)
        wprintf(L"%d stores configured, ID is: %s", nRc, szId);
    delete [] szId;
}
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.61.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.3.1.

```
int DisplayGetStandbyId(out string id)
```

```
Function DisplayGetStandbyId(ByRef id As String) As Integer
```

8.61.3.1 Implementation in C#

```
try
{
    string strId = "";
    int nCount = stPad.DisplayGetStandbyId(out strId);
    if (nCount == 0)
        MessageBox.Show(String.Format("No standby mode configured!"));
    else
        MessageBox.Show(String.Format("{0} stores configured, ID is: " +
                                      "{1}", nCount, strId));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.61.3.2 Implementation in Visual Basic

```
Try
    Dim strId As String = ""
    Dim nCount As Integer = STPad.DisplayGetStandbyId(strId)
    If nCount = 0 Then
        MsgBox("No standby mode configured!")
    Else
        MsgBox(CStr(nCount) & " stores configured, ID is: " & strId)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.62 DisplaySetBacklight method

This method controls the display backlight. The backlight is always set to the default value when the device is switched on. A default behaviour for opening and closing can be additionally defined in the STPad.ini file (see there for details).

With the Sigma model this method only works from firmware 1.10. With the Omega model, it only works from firmware 1.7. In Omega models with a firmware version that is older than 1.12, the values 1, 2 and 3 all set the default brightness.

Parameter	Values	I/O	Description

`LONG nMode`
`BACKLIGHT nMode`
`BacklightMode mode`
`ByVal mode As BacklightMode`
	0	I	The backlight is switched off

	1	I	The backlight is set to the default value
	2	I	The backlight is set to medium brightness
	3	I	The backlight is set to maximum brightness
Return value	**Values**	**Description**	
`LONG`	0	Method was executed successfully	
	< 0	Error	

8.62.1 STPadCapt.ocx

Available from Version 8.0.16. The status described is available from Version 8.3.1.

```
LONG DisplaySetBacklight(LONG nMode)
```

8.62.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.DisplaySetBacklight(0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.62.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.DisplaySetBacklight(0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.62.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.0.21.

```
LONG STDisplaySetBacklight(BACKLIGHT nMode)
```

The `BACKLIGHT` enumeration is defined as follows:

```
kOff = 0,
kOn = 1,
kMedium = 2,
kMaximum = 3
```

8.62.2.1 Implementation in C++

```
LONG nRc = STDisplaySetBacklight(kOff);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.62.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.0.21.

```
void DisplaySetBacklight(signotec.STPadLibNet.BacklightMode mode)
```

```
Sub DisplaySetBacklight(ByVal mode As signotec.STPadLibNet.BacklightMode)
```

The `BacklightMode` enumeration is defined as follows:

```
Off = 0,
On = 1,
Medium = 2,
Maximum = 3
```

8.62.3.1 Implementation in C#

```
try
{
    stPad.DisplaySetBacklight(BacklightMode.Off);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.62.3.2 Implementation in Visual Basic

```
Try
    STPad.DisplaySetBacklight(BacklightMode.Off)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.63 ControlSetLogDirectory method

This method allows logging to be controlled in any folder regardless of the settings in the STPad.ini file (depending on the component, the LogSTPadCapt, LogSTPadLib or LogSTPadLibNet keys). The component must have write permissions in this folder. Accessing this method always closes the current log file. If a valid path is transferred, information is immediately written to an existing or new log file in the specified folder.

Parameter	Values	I/O	Description
`BSTR bstrPath`	`NULL`	I	Logging is disabled.

`LPCWSTR szPath`
`string path`
`ByVal path As String`
	other	I	Absolute path to an existing folder where the logging is to take place; this path may contain environment variables that must be enclosed with %

VARIANT nOptions LONG nOptions LogFlag options ByVal options As LogFlag	Bitmask containing one or more hexadecimal values from the following list:		
	0x01	I	Activate standard logging
	0x02	I	Activate communication logging; should only be activated on request from signotec, as this has a negative effect on performance
	0x04	I	Activate logging for Virtual Channel (in client)
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.63.1 STPadCapt.ocx

Available from Version 8.0.21.12. The status described is available from Version 8.5.2.0.

```
LONG ControlSetLogDirectory(BSTR bstrPath, [optional]VARIANT nOptions)
```

Note: The parameter `nOptions` is only optional for compatibility reasons and should always be transmitted. It must contain a number.

8.63.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.ControlSetLogDirectory("%USERPROFILE%", 1);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.63.1.2 Implementation in Visual Basic

```
Dim nResult As Integer
nResult = AxSTPadCapt1.ControlSetLogDirectory("%USERPROFILE%", 1)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.63.2 STPadLib.dll

Available from Version 8.0.21.12. The status described is available from Version 8.5.2.0.

```
LONG STControlSetLogDirectory(LPCWSTR szPath, LONG nOptions=STPAD_LOG_STPADLIB)
```

The following values defined in the header file can be used for the `nOptions` parameter:

```
#define STPAD_LOG_STPADLIB    0x01
#define STPAD_LOG_STPAD       0x02
#define STPAD_LOG_STVCPAD     0x04
```

8.63.2.1 Implementation in C++

```
LONG nRc = STControlSetLogDirectory(L"%USERPROFILE%");
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.63.3 STPadLibNet.dll

Available from Version 8.0.21.12. The status described is available from Version 8.5.2.0.

```
void ControlSetLogDirectory(string path)
void ControlSetLogDirectory(string path, LogFlag options)
Sub ControlSetLogDirectory(ByVal path As String)
Sub ControlSetLogDirectory(ByVal path As String, ByVal options As LogFlag)
```

The `LogFlag` enumeration is defined as follows:

```
Off = 0x00,
STPadLibNet = 0x01,
STPad = 0x02,
STVCPad = 0x04
```

If the parameter `options` is missing, the value will be assumed as `LogFlag.STPadLibNet`.

8.63.3.1 Implementation in C#

```
try
{
    stPad.ControlSetLogDirectory("%USERPROFILE%");
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.63.3.2 Implementation in Visual Basic

```
Try
    STPad.ControlSetLogDirectory("%USERPROFILE%")
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.64 ControlGetVersion method

This method is deprecated. Please use the `ControlVersion` property instead.

8.65 ControlErase method

This method erases the captured signature data and the displayed signature as well as all bitmaps and text in the control element's window.

Parameter	Values	I/O	Description
-	-	-	-
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.65.1 STPadCapt.ocx

Available from Version 8.0.1.

```
LONG ControlErase()
```

8.65.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.ControlErase();
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.65.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.ControlErase()
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.65.2 STPadLib.dll

Not available.

8.65.3 STPadLibNet.dll

Available from Version 8.0.21 (only in the STPadLibControl class).

```
void ControlErase()
```

```
Sub ControlErase()
```

8.65.3.1 Implementation in C#

```
try
{
    stPad.ControlErase();
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.65.3.2 Implementation in Visual Basic

```
Try
    STPad.ControlErase()
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.66 ControlSetHotspotMode method

This method can be used to change the behaviour of a monitored area (hotspot) in the control element if the `ControlMirrorDisplay()` property is set to four (interactive). The mode set by `SensorSetHotSpotMode()` always takes priority; operating a hotspot deactivated there in the control element is therefore never possible.

This method has no effect on hotspots that have been added via the `SensorAddKeypadHotspot()` method. It is never possible to operate these hotspots in the control element.

Parameter	Values	I/O	Description
LONG nMode HOTSPOTMODE nMode HotSpotMode mode ByVal mode As HotSpotMode	0	I	Deactivates the monitored area
	1	I	Activates the monitored area
	2	I	Activates the monitored area but disables the automatic inverting when the area is clicked
LONG nHotSpotId int hotSpotId ByVal hotSpotId As Integer	>= 0	I	ID of the hotspot that is to be changed
Return value	**Values**	**Description**	
LONG	>= 0	ID of the hotspot that was generated	
	< 0	Error	

8.66.1 STPadCapt.ocx

Available from Version 8.1.2. The status described is available from Version 8.3.1.

```
LONG ControlSetHotSpotMode(LONG nMode, LONG nHotSpotId)
```

8.66.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.ControlSetHotspotMode(0, 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.66.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.ControlSetHotspotMode(0, 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.66.2 STPadLib.dll

Not available.

8.66.3 STPadLibNet.dll

Available from Version 8.1.2.

```
void ControlSetHotSpotMode(signotec.STPadLibNet.HotSpotMode mode, int
hotSpotId)
```

```
Sub ControlSetHotspotMode(ByVal mode As signotec.STPadLibNet.HotSpotMode, ByVal
hotSpotId As Integer)
```

The `HotSpotMode` enumeration is defined as follows:

```
Inactive = 0,
Active = 1,
InvertOff = 2
```

8.66.3.1 Implementation in C#

```
try
{
    stPad.SensorSetHotspotMode(HotSpotMode.Inactive, 0);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.66.3.2 Implementation in Visual Basic

```
Try
    STPad.SensorSetHotspotMode(HotSpotMode.Inactive, 0)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.67 ControlGetErrorString method

This method returns an error description in German, English, French or Italian, depending on the system language.

Parameter	Values	I/O	Description
`BSTR* pbstrError`	String	O	Error description.
`LPCWSTR szError`	`NULL`	I	The method returns the length of the error description in the `pnStringLength` parameter
	`!= NULL`	I/O	Array in which the error description is written; if the array is too small, the end characters are cut off
`LONG* pnStringLength`	`>= 0`	I/O	Length of the error description incl. terminated 0 or size of the `szError` array in bytes
`LONG nErrorId`	`0`	I	The description of the last error that occurred will be returned.
	`< 0`	I	Error number, for which the description should be returned.
Return value	**Values**	**Description**	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	

8.67.1 STPadCapt.ocx

Available from Version 8.0.11.

```
LONG ControlGetErrorString(BSTR* pbstrError, LONG nErrorId)
```

8.67.1.1 Implementation in C#

```
string strError = "";
axSTPadCapt1.ControlGetErrorString(ref strError, 0);
MessageBox.Show(strError);
```

8.67.1.2 Implementation in Visual Basic

```
Dim strError As String = ""
AxSTPadCapt1.ControlGetErrorString(strError, 0)
MsgBox(strError)
```

8.67.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STControlGetErrorString(LPCWSTR szError, LONG* pnStringLength, LONG
nErrorId)
```

8.67.2.1 Implementation in C++

```
LONG nLen = 0;
LONG nRc = STControlGetErrorString(NULL, &nLen, 0);
if (nRc == 0)
{
    WCHAR* szError = new WCHAR[nLen / sizeof(WCHAR)];
    nRc = STControlGetErrorString(szError, &nLen, 0);
    if (nRc == 0)
        wprintf(szError);
    delete [] szError;
}
```

8.67.3 STPadLibNet.dll

Not available. Please use the `Message` property of the `STPadException` instead.

8.68 ControlSetSTPadLib method

This method allows an instance of the STPadLib class to be passed that is to be used by the STPadLibControl class to communicate with signotec LCD pads. It can be called at any time. For example, a STPadLibControl object can be used to alternately display the screen content of two different devices that are each opened in a STPadLib object. After assignment, it makes no difference whether the methods and properties of the STPadLib or STPadLibControl instance are used.

If a STPadLib object, which is already assigned to a STPadLibControl object, is assigned to another STPadLibControl object, the assignment to the earlier object is removed.

This method should only be used if a separate instance of the STPadLib class is used.

Parameter	Values	I/O	Description
`STPadLib stPadLib`	`!= NULL`	I	The STPadLib object to be used for communication
	`NULL`	I	The currently used STPadLib object is removed and destroyed (provided it was created internally); the control element will use a new internal instance of the STPadLib class for communication purposes
Return value	**Values**	**Description**	
-	-	-	

8.68.1 STPadCapt.ocx

Not available.

8.68.2 STPadLib.dll

Not available.

8.68.3 STPadLibNet.dll

Available from Version 8.0.21 (only in the STPadLibControl class).

```
void ControlSetSTPadLib(signotec.STPadLibNet.STPadLib stPadLib)
```

```
Sub ControlSetSTPadLib(ByVal stPadLib As signotec.STPadLibNet.STPadLib)
```

8.68.3.1 Implementation in C#

```
STPadLib STPad = new STPadLib();
STPadLibControl STPadCtrl = new STPadLibControl();
STPadCtrl.ControlSetSTPadLib(STPad);
```

8.68.3.2 Implementation in Visual Basic

```
Dim WithEvents STPad As STPadLib = New STPadLib
Dim WithEvents STPadCtrl As STPadLibControl = New STPadLibControl
STPadCtrl.ControlSetSTPadLib(STPad)
```

8.69 ControlSetCallback method

This method defines a callback routine that is called if one of the events is triggered. For more information, see the section 'Events'.

Parameter	Values	I/O	Description
CBPTR pCallback	NULL	I	No callback is used
	!= NULL	I	Pointer to the callback routine
LPVOID pCustomPar	all	I	Any parameter that is passed when the callback routine is called; normally a pointer to the class whose methods are called from the callback routine
Return value	**Values**	**Description**	
-	-	-	

8.69.1 STPadCapt.ocx

Not available.

8.69.2 STPadLib.dll

Available from Version 8.0.19.

```
VOID STControlSetCallback(CBPTR pCallback, LPVOID pCustomPar)
```

The `CBPTR` type is defined as follows:

```
typedef VOID (*CBPTR)(LONG nEvent, LPVOID pData, LONG nDataSize, LPVOID
pCustomPar);
```

Parameter	Values	I/O	Description
`LONG nEvent`	Index of the triggered event from the following list:		
	`0`	I	`DeviceDisconnected()`
	`1`	I	`SensorHotSpotPressed()`
	`2`	I	`SensorTimeoutOccured()`
	`3`	I	`DisplayScrollPosChanged()`
	`4`	I	`SignatureDataReceived()`
`LPVOID pData`	`!= NULL`	I	Array of data that is given as a parameter to the event; please refer to the respective event for a description of the parameters
`LONG nDataSize`	`> 0`	I	Size of the `pData` array in bytes
`LPVOID pCustomPar`	all	I	Parameter that was passed when calling `STControlSetCallback()`; normally a pointer to the class whose methods are called from the callback routine
Return value	**Values**	**Description**	
-	-	-	

The following values defined in the header file can be used for the `nEvent` parameter:

```
#define STPAD_CALLBACK_DISCONNECT   0
#define STPAD_CALLBACK_HOTSPOT      1
#define STPAD_CALLBACK_TIMEOUT      2
#define STPAD_CALLBACK_SCROLL       3
#define STPAD_CALLBACK_SIGNATURE    4
```

8.69.2.1 Implementation in C++

```
VOID Callback(LONG nEvent, LPVOID pData, LONG nDataSize, LPVOID
              pCustomPar)
{
    if (!pCustomPar)
        return;

    CMyClass* pCls = (CMyClass*)pCustomPar;
    switch (nEvent)
    {
        case STPAD_CALLBACK_DISCONNECT:
            if (nDataSize >= sizeof(LONG))
                pCls->DeviceDisconnected(*(LONG*)pData);
            break;
        case STPAD_CALLBACK_HOTSPOT:
            if (nDataSize >= sizeof(LONG))
                pCls->SensorHotSpotPressed(*(LONG*)pData);
            break;
        case STPAD_CALLBACK_TIMEOUT:
            if (nDataSize >= sizeof(LONG))
                pCls->SensorTimeoutOccured(*(LONG*)pData);
            break;
        case STPAD_CALLBACK_SCROLL:
            if (nDataSize >= (2 * sizeof(LONG)))
                pCls->DisplayScrollPosChanged(*(LONG*)pData,
                                              *((LONG*)pData + 1));
            break;
        case STPAD_CALLBACK_SIGNATURE:
            if (nDataSize >= (4 * sizeof(LONG)))
                pCls->SignatureDataReceived(*(LONG*)pData,
                                            *((LONG*)pData + 1),
                                            *((LONG*)pData + 2),
                                            *((LONG*)pData + 3));
            break;
      }
}

CMyClass::CMyClass()
{
    STControlSetCallback(&Callback, (VOID*)this);
}
```

8.69.3 STPadLibNet.dll

Not available.

8.70 ControlExit method

This method releases used resources; it must be called before the component is de-initialised.

Parameter	Values	I/O	Description
–	–	-	-
Return value	**Values**	**Description**	
–	–	-	

8.70.1 STPadCapt.ocx

Not available.

8.70.2 STPadLib.dll

Available from Version 8.0.19.

```
VOID STControlExit()
```

8.70.2.1 Implementation in C++

```
STControlExit();
```

8.70.3 STPadLibNet.dll

Not available.

8.71 RSAGenerateSigningCert/RSAGenerateSigningCertPw method

This method initiates the generation of a unique RSA key pair in the device that is used to sign data. The keys generated are saved permanently together with a public X.509 certificate, which is customised using the serial number of the device as well as a Certificate Signing Request (CSR). If the `RSASignPasswordLength` property has been set in advance, signing cannot take place with the certificate until a password with a respective minimum length has been specified using `RSASetSignPassword()`.

Note: Generation may last several minutes depending on key length!

It is possible to permanently disable the generation of a pair of keys inside the signature device. This function can also be protected with a password. Please refer to your contact at signotec as required.

These methods only work with the Sigma model from firmware 1.16. With the Omega model, they only work from firmware 1.25.

Parameter	Values	I/O	Description

`LONG nKeyLen`
`int keyLen`
`ByVal keyLen As Integer`
	`1024` – `4096`	I	Key length in bits;
			If the device property "Supports4096BitKeys" has been set, the pad will support all multiples of 8 within the specified range as length. Otherwise only the values `1024` and `2048` are supported. See also `DeviceGetCapabilities()`.

	`0`	I	No key figures are generated; only the certificate with the stored key figures is created again; this only works if the key pair saved in the signature device has a length of 1024 or 2048 bits

`LONG nValidity`
`int validity`
`ByVal validity As Integer`
	`0`	I	The generated certificate is valid until 31.12.2049.

	`>0`	I	Duration of validity of the certificate in months from current date

BSTR bstrDevicePassword LPCWSTR szDevicePassword SecureString devicePassword ByVal devicePassword As SecureString	max. 32 characters	I	Password of the device (if it is password protected); please refer to your contact at signotec for details
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.71.1 STPadCapt.ocx

Available from Version 8.0.26. The status described is available from Version 8.4.0.

```
LONG RSAGenerateSigningCert(LONG nKeyLen, LONG nValidity)
```

```
LONG RSAGenerateSigningCertPw(LONG nKeyLen, LONG nValidity, BSTR
bstrDevicePassword)
```

8.71.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.RSAGenerateSigningCert(2048, 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.71.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.RSAGenerateSigningCert(2048, 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.71.2 STPadLib.dll

Available from Version 8.0.26. The status described is available from Version 8.4.0.

```
LONG STRSAGenerateSigningCert(LONG nKeyLen, LONG nValidity)
```

```
LONG STRSAGenerateSigningCertPw(LONG nKeyLen, LONG nValidity, LPCWSTR
szDevicePassword)
```

8.71.2.1 Implementation in C++

```
long nResult = STRSAGenerateSigningCert(2048, 0);
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

8.71.3 STPadLibNet.dll

Available from Version 8.0.26. The status described is available from Version 8.4.2.0.

```
void RSAGenerateSigningCert(int keyLen, int validity)

void RSAGenerateSigningCertPw(int keyLen, int validity,
System.Security.SecureString devicePassword)

Sub RSAGenerateSigningCert(ByVal keyLen As Integer, ByVal validity As Integer)

Sub RSAGenerateSigningCertPw(ByVal keyLen As Integer, ByVal validity As
Integer, ByVal devicePassword As System.Security.SecureString)
```

8.71.3.1 Implementation in C#

```
try
{
    stPad.RSAGenerateSigningCert(2048, 0);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.71.3.2 Implementation in Visual Basic

```
Try
    STPad.RSAGenerateSigningCert(2048, 0)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.72 RSASetSigningCert/RSASetSigningCertPw method

This method imports an X.509 certificate or a PKCS#12 container, the private key of which is intended to be used for signing data and saves the certificate and, if necessary, the key pair permanently in the signature device. If only the public key is transferred, the public key must correspond to the key already generated and saved in the signature device and only the certificate saved in the device will be overwritten. If a PKCS#12 container with a private key and public certificate is transferred, the keys and the certificate will be overwritten in the device, a saved Certificate Signing Request (CSR) will be deleted. If the `RSASignPasswordLength` property has been set in advance, signing cannot take place with the certificate until a password with a respective minimum length has been specified using `RSASetSignPassword()`.

It is possible to permanently disable the storing of a key pair (generated outside the signature device) inside the device. This function can also be protected with a password. Please refer to your contact at signotec as required.

These methods only work with the Sigma model from firmware 1.16. With the Omega model, they only work from firmware 1.25.

Parameter	Values	I/O	Description

`VARIANT& vaCert`
`BYTE* pbtCert`
`X509Certificate2 cert`
`string cert`
`ByVal cert As X509Certificate2`
`ByVal cert As String`
	`!= NULL`	I	X.509- Certificate or PKCS#12 container from the memory or a file. If the device property "Supports4096BitKeys" has been set, the pad will support all multiples of 8 within the range of `1024` to `4096` as length. Otherwise only the values `1024` and `2048` bit are supported. See also `DeviceGetCapabilities()`.

`LONG nSize`	`0`	I	The `pbtCert` pointer is a `WCHAR*` type and points to the certificate file path or URL.
	`> 0`	I	Size of transferred byte array

`BSTR bstrPassword`
`LPCWSTR szPassword`
`SecureString password`
`ByVal password As SecureString`
	all	I	Password used to fetch the private key (ignored if an X.509 certificate is transferred)

`BSTR bstrDevicePassword`
`LPCWSTR szDevicePassword`
`SecureString devicePassword`
`ByVal devicePassword As SecureString`
	max. 32 characters	I	Password of the device (if it is password protected); please refer to your contact at signotec for details

Return value	**Values**	**Description**	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	

8.72.1 STPadCapt.ocx

Available from Version 8.0.26. The status described is available from Version 8.4.0.

`LONG RSASetSigningCert(VARIANT& vaCert, BSTR bstrPassword)`

`LONG RSASetSigningCertPw(VARIANT& vaCert, BSTR bstrPassword, BSTR bstrDevicePassword)`

Note: The `vaCert` parameter must contain a byte array or a string.

8.72.1.1 Implementation in C#

Work in the memory:

```
X509Certificate2 cert = new X509Certificate2(@"C:\Cert.cer");
int nResult = axSTPadCapt1.RSASetSigningCert
                                (cert.Export(X509ContentType.Cert), null);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

Work with files:

```
int nResult = axSTPadCapt1.RSASetSigningCert(@"C:\Cert.cer", null);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.72.1.2 Implementation in Visual Basic

Work in the memory:

```
Dim cert As X509Certificate2 = New X509Certificate2("C:\Cert.cer")
Dim nResult As Integer = AxSTPadCapt1.RSASetSigningCert _
                            (cert.Export(X509ContentType.Cert), Nothing)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

Work with files:

```
Dim nResult As Integer
nResult = AxSTPadCapt1.RSASetSigningCert("C:\Cert.cer", Nothing)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.72.2 STPadLib.dll

Available from Version 8.0.26. The status described is available from Version 8.4.0.

```
LONG STRSASetSigningCert(BYTE* pbtCert, LONG nSize, LPCWSTR szPassword)
```

```
LONG STRSASetSigningCertPw(BYTE* pbtCert, LONG nSize, LPCWSTR szPassword,
LPCWSTR szDevicePassword)
```

8.72.2.1 Implementation in C++

Work in the memory:

```
long nResult;
nResult = STRSASetSigningCert(&btCert, sizeof(btCert), NULL);
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

Work with files:

```
long nResult = STRSASetSigningCert((BYTE*)L"C:\\Cert.cer", 0, NULL);
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

8.72.3 STPadLibNet.dll

Available from Version 8.0.26. The status described is available from Version 8.4.2.

```
void RSASetSigningCert(System.Security.Cryptography.X509Certificates.
X509Certificate2 cert, System.Security.SecureString password)

void RSASetSigningCert(string cert, System.Security.SecureString password)

void RSASetSigningCertPw(System.Security.Cryptography.X509Certificates.
X509Certificate2 cert, System.Security.SecureString password,
System.Security.SecureString devicePassword)

void RSASetSigningCertPw(string cert, System.Security.SecureString password,
System.Security.SecureString devicePassword)

Sub RSASetSigningCert(ByVal cert As System.Security.Cryptography.
X509Certificates.X509Certificate2, ByVal password As
System.Security.SecureString)

Sub RSASetSigningCert(ByVal cert As String, ByVal password As
System.Security.SecureString)

Sub RSASetSigningCertPw(ByVal cert As System.Security.Cryptography.
X509Certificates.X509Certificate2, ByVal password As
System.Security.SecureString, ByVal devicePassword As
System.Security.SecureString)

Sub RSASetSigningCertPw(ByVal cert As String, ByVal password As
System.Security.SecureString, ByVal devicePassword As
System.Security.SecureString)
```

8.72.3.1 Implementation in C#

Work in the memory:

```
try
{
    stPad.RSASetSigningCert(new X509Certificate2(@"C:\Cert.cer"),
                            (SecureString)null);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

Work with files:

```
try
{
    stPad.RSASetSigningCert(@"C:\Cert.cer", (SecureString)null);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.72.3.2 Implementation in Visual Basic

Work in the memory:

```
Try
    STPad.RSASetSigningCert(New X509Certificate2("C:\Cert.cer"),
                            DirectCast(Nothing, SecureString))
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

Work with files:

```
Try
    STPad.RSASetSigningCert("C:\Cert.cer",
                            DirectCast(Nothing, SecureString))
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.73 RSASaveSigningCertAsStream / RSASaveSigningCertAsFile method

Using this method, the public X.509 certificate stored in the signature device or the Certificate Signing Request (CSR) whose corresponding private key is used to sign data can be read out or saved to a file.

With the Sigma model this method only works from firmware 1.16. With the Omega model, it only works from firmware 1.25.

Parameter	Values	I/O	Description
`BYTE* pbtCert`	`NULL`	I	The method returns the required size of the array in the `pnSize` parameter.
	`!= NULL`	I/O	Array in the required size in which the X.509 certificate or CSR is written;`pnSize` must correspond to the value returned for the previous call
`LONG* pnSize`	`> 0`	I/O	Size of the array (in bytes) in which the certificate is to be written

`BSTR bstrPath`
`LPCWSTR szPath`
`string path`
`ByVal path As String`
	`!= NULL`	I	Storage location for the certificate as full path that includes the file name

`LONG nType`
`CERTTYPE nType`
`CertType type`
`ByVal type As CertType`
	0	I	Public certificate is read and DER returned encoded/saved or returned as `X509Certificate2` (`RSASaveSigningCertAsStream()` for STPadLibNet.dll)

	1	I	The CSR is read and the DER returned encoded or saved; a public X.509 certificate can be issued by a certification body with this, which can subsequently be saved in the signature device using the `RSASetSigningCert()` method
	2	I	Public certificate is read and PEM returned encoded or saved
	3	I	The CSR is read and the PEM returned encoded or saved; a public X.509 certificate can be issued by a certification body with this, which can subsequently be saved in the signature device using the `RSASetSigningCert()` method
Return value	**Values**	**Description**	
`VARIANT`	empty	Error	
	other	X.509 certificate or CSR as byte array	

LONG	0	Method was executed successfully
	< 0	Error
X509Certificate2 byte[] Byte()	!= NULL	X.509 certificate as X509Certificate2 or PEM encoded certificate / CSR as byte array

8.73.1 STPadCapt.ocx

Available from Version 8.0.26. The status described is available from Version 8.4.3.

```
VARIANT RSASaveSigningCertAsStream(LONG nType)
```

```
LONG STRSASaveSigningCertAsFile(LPCWSTR szPath, CERTTYPE nType)
```

8.73.1.1 Implementation in C#

Work in the memory:

```
byte[] btCert = (byte[])axSTPadCapt1.RSASaveSigningCertAsStream(0);
if (btCert == null)
{
    MessageBox.Show(String.Format("Error"));
    return;
}
```

Work with files:

```
int nResult = axSTPadCapt1.RSASaveSigningCertAsFile(@"C:\Cert.cer", 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.73.1.2 Implementation in Visual Basic

Work in the memory:

```
Dim btCert As Byte() = AxSTPadCapt1.RSASaveSigningCertAsStream(0)
If btCert Is Nothing Then
    MsgBox("Error")
    Exit Sub
End If
```

Work with files:

```
Dim nResult As Integer
nResult = AxSTPadCapt1.RSASaveSigningCertAsFile("C:\Cert.cer", 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.73.2 STPadLib.dll

Available from Version 8.0.26. The status described is available from Version 8.4.3.

```
LONG STRSASaveSigningCertAsStream(BYTE* pbtCert, LONG* pnSize, CERTTYPE nType)
```

```
LONG STRSASaveSigningCertAsFile(LPCWSTR szPath, CERTTYPE nType)
```

The CERTTYPE enumeration is defined as follows:

```
kCert_DER = 0,
kCSR_DER = 1,
kCert_PEM = 2,
kCSR_PEM = 3
```

8.73.2.1 Implementation in C++

Work in the memory:

```
long nSize = 0;
long nResult = STRSASaveSigningCertAsStream(NULL, &nSize, kCert_DER);
BYTE* pbtCert = NULL;
if (nResult == 0)
{
    pbtCert = new BYTE[nSize];
    nResult = STRSASaveSigningCertAsStream(pbtCert, &nSize, kCert_DER);
}
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

Work with files:

```
long nResult = STRSASaveSigningCertAsFile(L"C:\\Cert.cer", kCert_DER);
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

8.73.3 STPadLibNet.dll

Available from Version 8.0.26. The status described is available from Version 8.4.3.

```
object RSASaveSigningCertAsStream(signotec.STPadLibNet.CertType type)
```

```
void RSASaveSigningCertAsFile(string path, signotec.STPadLibNet.CertType type)
```

```
Function RSASaveSigningCertAsStream(ByVal type As
signotec.STPadLibNet.CertType) As Object
```

```
Sub RSASaveSigningCertAsFile(ByVal path As String, ByVal type As
signotec.STPadLibNet.CertType)
```

The `CertType` enumeration is defined as follows:

```
Cert_DER = 0,
CSR_DER = 1
Cert_PEM = 2,
CSR_PEM = 3
```

8.73.3.1 Implementation in C#

Work in the memory:

```
X509Certificate2 cert;
try
{
    cert = (X509Certificate2)stPad.RSASaveSigningCertAsStream
                                                    (CertType.Cert_DER);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

Work with files:

```
try
{
    stPad.RSASaveSigningCertAsFile(@"C:\Cert.cer", CertType.Cert_DER);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.73.3.2 Implementation in Visual Basic

Work in the memory:

```
Dim cert As X509Certificate2
Try
    cert = STPad.RSASaveSigningCertAsStream(CertType.Cert_DER)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

Work with files:

```
Try
    STPad.RSASaveSigningCertAsFile("C:\Cert.cer", CertType.Cert_DER)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.74 RSASetHash method

This method can be used to transfer a data block (hash 1) that is to be signed in the signature device. Subsequently, `RSASign()` can be called up in order to sign hash 1 directly, or `SignatureStart()`, in order to transfer hash 1 to the signature device and, after capturing the signature, to sign together with hash 2 which is generated using the biometric data.

It is possible to permanently disable the signing of a hash that was generated outside the signature device inside the device. In this case, hash 1 can only be generated using the `RSACreateDisplayHash()` method. Please refer to your contact at signotec as required.

With the Sigma model this method only works from firmware 1.16. With the Omega model, it only works from firmware 1.25.

Parameter	Values	I/O	Description

`VARIANT& vaHash`
`BYTE* pbtHash`
`byte[] hash`
`ByVal hash As Byte()`
	`!= NULL`	I	Byte array with a hash with a length of up to 32 bytes (depending on algorithm), the content of which is to be signed

	`NULL`	I	Only hash 2 is generated on the biometric data; accordingly, `RSASign()` cannot be used to sign Hash 1

LONG nAlgorithm HASHALGO nAlgorithm HashAlgo algorithm ByVal algorithm As HashAlgo	Algorithm that has been used to create the data block to be signed and/or with which Hash 1 and Hash 2 are to be generated:		
	0	I	SHA-1 (hash 1 must be 20 bytes long)
	1	I	SHA-256 (hash 1 must be 32 bytes long)
LONG nOptions HashFlag options ByVal options As HashFlag	Bitmask containing one or more hexadecimal values from the following list:		
	0x01	I	The data block to be signed is in little endian byte order (otherwise in big endian)
	0x02	I	The given byte array does not contain a hash; rather, it contains data with which the hash is to be calculated
LONG nDataSize	= 0	I	Size of the transferred byte array; ignored if nOptions does not contain the value 0x02; the length is then determined using the nAlgorithm parameter
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.74.1 STPadCapt.ocx

Available from Version 8.0.26. The status described is available from Version 8.1.3.

```
LONG RSASetHash(VARIANT& vaHash, LONG nAlgorithm, LONG nOptions)
```

Note: The vaHash parameter must be NULL or contain a byte array.

8.74.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.RSASetHash(hash, 1, 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.74.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.RSASetHash(hash, 1, 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.74.2 STPadLib.dll

Available from Version 8.0.26. The status described is available from Version 8.4.0.

```
LONG STRSASetHash(BYTE* pbtHash, HASHALGO nAlgorithm, LONG nOptions, LONG
nDataSize=0)
```

The HASHALGO enumeration is defined as follows:

```
kSha1 = 0,
kSha256 = 1
```

The following values defined in the header file can be used for the nOptions parameter:

```
#define STPAD_RSA_LITTLEENDIAN      0x01
#define STPAD_RSA_HASHDATA          0x02
```

8.74.2.1 Implementation in C++

```
long nResult = STRSASetHash(pbtHash, kSha256, 0);
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

8.74.3 STPadLibNet.dll

Available from Version 8.0.26.

```
void RSASetHash(signotec.STPadLibNet.HashAlgo algorithm)
```

```
void RSASetHash(byte[] hash, signotec.STPadLibNet.HashAlgo algorithm,
signotec.STPadLibNet.HashFlag options)
```

```
Sub RSASetHash(ByVal algorithm As signotec.STPadLibNet.HashAlgo)
```

```
Sub RSASetHash(ByVal hash As Byte(), ByVal algorithm As
signotec.STPadLibNet.HashAlgo, ByVal options As signotec.STPadLibNet.HashFlag)
```

The `HashAlgo` enumeration is defined as follows:

```
SHA1 = 0,
SHA256 = 1
```

The `HashFlag` enumeration is defined as follows:

```
None = 0x00,
LittleEndian = 0x01,
HashData = 0x02
```

8.74.3.1 Implementation in C#

```
try
{
    stPad.RSASetHash(hash, HashAlgo.SHA256, HashFlag.None);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.74.3.2 Implementation in Visual Basic

```
Try
    STPad.RSASetHash(hash, HashAlgo.SHA256, HashFlag.None)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.75 RSACreateDisplayHash method

This method initiates the calculation of a hash sum in the signature device using the content of the image memory and supplies, as a byte array, the image data stream that was used to calculate hash 1. One of the following methods should then be called: `SignatureStart()` to start content signing or `RSASign()` to enable hash 1 to be signed in the signature device.

Calling `SignatureCancel()` discards the previously generated hash 1. It is sometimes not possible to access other methods.

Content signing is unique in that the screen content displayed during signing in the form of hash 1 can be signed together with the hash based on biometric data (hash 2). Hash 2 can be calculated retrospectively using the biometric data supplied when calling the `RSAGetSignData()` method (see also `RSADecryptSignData()`). As a result, the screen content is inseparably linked with the signature.

The image data stream that is returned can be converted into an image using the `RSACreateHashedImage()` method. If the source memory contains content from a permanent memory that has not been previously reserved, this data is not included in the calculation of the hash sum! If the content of the selected source memory is empty, no image data stream is returned.

Note: The calculation of the hash sum may last several seconds depending on the image data and algorithm.

With the Sigma model this method only works from firmware 1.16. With the Omega model, it only works from firmware 1.25 and with the Alpha model, it only works from firmware 1.8.

Parameter	Values	I/O	Description
`BYTE* pbtImageData`	`NULL`	I	The method returns the required size of the array in the `pnSize` parameter.
	`!= NULL`	I/O	Byte array in the required size that is used to write the image data stream and was used to calculate hash 1; `pnSize` must correspond to the value returned during the last call.
`LONG* pnSize`	`128` `256`	I/O	Size of the byte array that is referenced by `pbtImageData` in bytes

`LONG nAlgorithm`
`HASHALGO nAlgorithm`
`HashAlgo algorithm`
`ByVal algorithm As HashAlgo`
	Algorithm to be used to generate hash 1		

	`0`	I	SHA-1
	`1`	I	SHA-256
	`2`	I	SHA-512

`LONG nSource`
`DisplayTarget source`
`ByVal target As source`
	`0`	I	The content of the foreground buffer corresponding to the size of the display is used for the calculation.

	`1`	I	The content of the background memory corresponding to the size of the display is used for the calculation and subsequently displayed in the visible foreground memory
	`1000`	I	The content of the virtual memory in the API corresponding to the size of the display is used for the calculation and subsequently displayed in the visible foreground memory; as a rule, this memory should be used, as the data is optimised in the API before transfer, meaning it can be displayed faster
	other	I	ID of the permanent memory whose content corresponding to the size of the display is used for the calculation and subsequently displayed in the visible foreground memory

Return value	Values	Description
VARIANT	empty	Error or the content of the source memory is empty
	other	The image data stream as a byte array that was used to calculate hash 1
LONG	0	Method was executed successfully
	< 0	Error
byte[] Byte()	NULL	The content of the source memory is empty

8.75.1 STPadCapt.ocx

Available from Version 8.0.23.9.

```
VARIANT RSACreateDisplayHash(LONG nAlgorithm, LONG nSource)
```

8.75.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.RSACreateDisplayHash(1, 1000);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.75.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.RSACreateDisplayHash(1, 1000)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.75.2 STPadLib.dll

Available from Version 8.0.23.9.

```
LONG STRSACreateDisplayHash(BYTE* pbtImageData, LONG* pnSize, HASHALGO
nAlgorithm, LONG nSource)
```

The `HASHALGO` enumeration is defined as follows:

```
kSha1 = 0,
kSha256 = 1,
kSha512 = 2
```

The following values defined in the header file can be used for the `nTarget` parameter:

```
#define STPAD_TARGET_FOREGROUND     0
#define STPAD_TARGET_BACKGROUND     1
#define STPAD_TARGET_DISPLAYHASH 1000
```

8.75.2.1 Implementation in C++

```
long nSize = 0;
BYTE* pbtImageData = NULL;
long nResult = STRSACreateDisplayHash(NULL, &nSize, kSha256,
                                      STPAD_TARGET_DISPLAYHASH);
if (nResult == 0)
{
    pbtImageData = new BYTE[nSize];
    nResult = STRSACreateDisplayHash(pbtImageData, &nSize, kSha256,
                                     STPAD_TARGET_DISPLAYHASH);
}
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

8.75.3 STPadLibNet.dll

Available from Version 8.0.23.9.

```
byte[] RSACreateDisplayHash(signotec.STPadLibNet.HashAlgo algorithm,
signotec.STPadLibNet.DisplayTarget source)
```

```
Function RSACreateDisplayHash(ByVal algorithm As signotec.STPadLibNet.HashAlgo,
ByVal source As signotec.STPadLibNet.DisplayTarget) As Byte()
```

The `HashAlgo` enumeration is defined as follows:

```
SHA1 = 0,
SHA256 = 1,
SHA512 = 2
```

The `DisplayTarget` enumeration is defined as follows:

```
ForegroundBuffer = 0,
BackgroundBuffer = 1,
Reserved1 = 3,
Reserved2 = 4,
Reserved3 = 5,
Reserved4 = 6,
Reserved5 = 7,
Reserved6 = 8,
Reserved7 = 9,
Reserved8 = 10,
Reserved9 = 11,
Reserved10 = 12,
Reserved11 = 13,
DisplayHashBuffer = 1000
```

8.75.3.1 Implementation in C#

```
byte[] imageData = null;
try
{
    imageData = stPad.RSACreateDisplayHash(HashAlgo.SHA256,
                                           DisplayTarget.DisplayHashBuffer);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.75.3.2 Implementation in Visual Basic

```
Dim imageData() As Byte
Try
    imageData = STPad.RSACreateDisplayHash(HashAlgo.SHA256, _
                                           DisplayTarget.DisplayHashBuffer)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.76 RSACreateHashedImage method

This method generates an image from the image data stream that was returned by the `RSACreateDisplayHash()` method.

Parameter	Values	I/O	Description
`BYTE* pbtImage`	`NULL`	I	The method calculates the image, caches it and returns the required size of the array in the `pnSize` parameter.
	other	I/O	Array (in the required size) in which the cached image data is written; `pnSize` must correspond to the value returned for the previous call; all other parameters are ignored.
`LONG* pnSize`	`> 0`	I/O	Size of the array in which the image data is to be written

`VARIANT& vaImageData`
`BYTE* pbtImageData`
`byte[] imageData`
`ByVal imageData As Byte()`
	`!= NULL`	I	Image data stream, as it was returned by the `RSACreateDisplayHash()` method, as a byte array

`LONG nImageDataSize`	`> 0`	I	Size of the array that is referenced by `pbtImageData` in bytes

`OLE_COLOR clrBack`
`COLORREF clrBack`
`Color backColor`
`ByVal backColor As Color`
	`>= 0`	I	Colour of the pixels that were not included in the image data stream; these pixels may arise if, when `CreateDisplayHash()` is called, the source memory contained image data from a permanent memory that had not been reserved in advance; otherwise, this parameter has no effect.

`LONG nModelType`
`int modelType`
`ByVal modelType As Integer`
	Model type of the signature device from which the image data stream originates; this determines the colour depth and resolution of the image		

	`1, 2`	I	'Sigma' model type
	`5, 6`	I	'Zeta' model type
	`11, 12`	I	'Omega' model type
	`15, 16`	I	'Gamma' model type
	`21, 22, 23`	I	'Delta' model type

	31, 32, 33	I	'Alpha' model type
LONG nFileType FILETYPE nFileType	0	I	Use TIFF with CCITT4 compression (b/w image) or LZW compression (colour image) as the file format (recommended)
	1	I	Use PNG file format
	2	I	Use BMP file format
	3	I	Use JPEG with a quality setting of 75 as the file format
	4	I	Use GIF as the file format
	200 – 204	I	Image data is not returned as binary data, but as Base64 encoded data; otherwise as values 0–4 (only STPadCapt.ocx)
Return value	**Values**	**Description**	
VARIANT	empty	Error	
	other	Image data as array of Bytes or Base64-coded String	
LONG	0	Method was executed successfully	
	< 0	Error	
Bitmap	!= NULL	Image as `System.Drawing.Bitmap`	

8.76.1 STPadCapt.ocx

Available from Version 8.0.23.9. The status described is available from Version 8.3.1.

```
VARIANT RSACreateHashedImage(VARIANT& vaImageData, OLE_COLOR clrBack, LONG
nModelType, LONG nFileType)
```

8.76.1.1 Implementation in C#

```
byte[] btImage = (byte[])axSTPadCapt1.RSACreateHashedImage(btImageData,
                                                           0, 11, 0);
if (btImage == null)
{
    MessageBox.Show(String.Format("Error"));
    return;
}
MemoryStream memoryStream = new MemoryStream(btImage);
Image image = Image.FromStream(memoryStream);
```

8.76.1.2 Implementation in Visual Basic

```
Dim btImage() As Byte
btImage = AxSTPadCapt1.RSACreateHashedImage(btImageData, 0, 11, 0)
If btImage Is Nothing Then
    MsgBox("Error")
    Exit Sub
End If
Dim memoryStream As MemoryStream = New MemoryStream(btImage)
Dim image As Image = Image.FromStream(memoryStream)
```

8.76.2 STPadLib.dll

Available from Version 8.0.23.9. The status described is available from Version 8.3.1.

```
LONG STRSACreateHashedImage(BYTE* pbtImage, LONG* pnSize, BYTE* pbtImageData,
LONG nImageDataSize, COLORREF clrBack, LONG nModelType, FILETYPE nFileType)
```

The `FILETYPE` enumeration is defined as follows:

```
kTiff = 0,
kPng = 1,
kBmp = 2,
kJpeg = 3,
kGif = 4
```

8.76.2.1 Implementation in C++

```
long nSize = 0;
long nResult = STRSACreateHashedImage(NULL, &nSize, btImageData,
                                      sizeof(btImageData), 0, 11, kBmp);
BITMAP bitmap;
if (nResult == 0)
{
    BYTE* pbtImage = new BYTE[nSize];
    nResult = STRSACreateHashedImage(pbtImage, &nSize, btImageData,
                                     sizeof(btImageData), 0, 11, kBmp);
    BITMAPFILEHEADER bmfh = (*(BITMAPFILEHEADER*)pbtImage);
    BITMAPINFO bmi = (*(BITMAPINFO*)(pbtImage +
                                     sizeof(BITMAPFILEHEADER)));
    bitmap.bmType = 0;
    bitmap.bmWidth = bmi.bmiHeader.biWidth;
    bitmap.bmHeight = bmi.bmiHeader.biHeight;
    bitmap.bmPlanes = bmi.bmiHeader.biPlanes;
    bitmap.bmBitsPixel = bmi.bmiHeader.biBitCount;
    bitmap.bmWidthBytes = ((bitmap.bmWidth * bitmap.bmBitsPixel + 31)
                           >> 5) << 2;
    bitmap.bmBits = new BYTE[bitmap.bmHeight * bitmap.bmWidthBytes];
    memcpy(bitmap.bmBits, pbtImage + bmfh.bfOffBits, bitmap.bmHeight *
           bitmap.bmWidthBytes);
    delete [] pbtImage;
}
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

8.76.3 STPadLibNet.dll

Available from Version 8.0.23.9. The status described is available from Version 8.3.1.

```
System.Drawing.Bitmap RSACreateHashedImage(byte[] imageData,
System.Drawing.Color backColor, int modelType)
```

```
Function RSACreateHashedImage(ByVal imageData As Byte(), ByVal backColor As
System.Drawing.Color, ByVal modelType As Integer) As System.Drawing.Bitmap
```

8.76.3.1 Implementation in C#

```
Bitmap bitmap;
try
{
    bitmap = stPad.RSACreateHashedImage(imageData, Color.Black, 11);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.76.3.2 Implementation in Visual Basic

```
Dim bitmap As Bitmap
Try
    bitmap = STPad.RSACreateHashedImage(imageData, Color.Black, 11)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.77 RSASign/RSASignPw method

This method can be used to sign data in the signature device using the private key, which has been generated using the `RSAGenerateSigningCert()` method or saved using the `RSASetSigningCert()` method. If a minimum password length has been set when calling one of these methods, a signing process is only possible after setting a password with `RSASetSignPassword()` . Please also note the `RSASignPasswordLength` property.

With the Sigma model this method only works from firmware 1.16. With the Omega model, it only works from firmware 1.25.

Parameter	Values	I/O	Description
`BYTE* pbtSignature`	`NULL`	I	The method returns the required size of the array in the `pnSize` parameter.
	`!= NULL`	I/O	Byte array in the required size, in which the signed data block is written; `pnSize` must correspond to the value returned for the previous call
`LONG* pnSize`	`128` `256`	I/O	Size of the array that is referenced by `pbtSignature` in bytes

`LONG nScheme`
`RSASCHEME nScheme`
`RSAScheme scheme`
`ByVal scheme As RSAScheme`
	Signature scheme that is to be used		

	`0`	I	Only a padding is added according to RSASSA-PKCS1-V1_5, no hash OID
	`1`	I	The OID of the hash algorithm used and a padding in accordance with RSASSA-PKCS1-V1_5 are added; this scheme cannot be used if the combination of hash 1 and hash 2 is to be signed
	`2`	I	The RSASSA-PSS signature scheme is used; the salt has the same length as hash 1 and hash 2 respectively, and the algorithm using for encoding is the same as the one that has been used to generate hash 1 and hash 2 respectively; MGF 1 is used as mask generation function.

`LONG nHashValue`
`HASHVALUE nHashValue`
`HashValue hashValue`
`ByVal hashValue As HashValue`
	Specifies the data block that is to be signed		

	`0`	I	The combination of hash 1 and hash 2 is signed; hash 1 is placed after hash 2 in big endian byte order; if SHA1 is used, scheme RSASSA_PSS scheme must be used; scheme RSASSA-PKCS1-V1_5 cannot be used; if the SHA-512 algorithm was used, the key saved in the signature device must be at least 2048 bit long
	`1`	I	Hash 1 is signed, which is transferred when `RSASetHash()` is called.
	`2`	I	Hash 2, which has been generated using the biometric data, is signed

`LONG nOptions`
`SignFlag options`
`ByVal options As SignFlag`
	Bitmask containing one or more hexadecimal values from the following list:		

	`0x01`	I	The signature result is returned in little endian byte order (otherwise in big endian)
	`0x02`	I	The signature result is returned as message PKCS#7 (CMS); the value `0x01` is ignored; the RSA scheme must be RSASSA-PKCS1-V1_5 with OID (value 1)
	`0x04`	I	Includes not only the end certificate, but the certificate chain with the exception of the root certificate (only if `0x02` is also set)

`LPCWSTR szSignPassword`
`LPCWSTR bstrSignPassword`
`SecureString signPassword`
`ByVal password As SecureString`
	max. 32 characters	I	Password of the signature certificate (if it is password protected)

Return value	**Values**	**Description**	
`VARIANT`	empty	Error	
	other	Signed data block as byte array	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	
`byte[]` `Byte()`	`!= NULL`	Signed data block as byte array	

8.77.1 STPadCapt.ocx

Available from Version 8.0.26. The status described is available from Version 8.4.3.

```
VARIANT RSASign(LONG nScheme, LONG nHashValue, LONG nOptions)

VARIANT RSASignPw(LONG nScheme, LONG nHashValue, LONG nOptions, LPCWSTR
bstrSignPassword)
```

8.77.1.1 Implementation in C#

```
byte[] btSignature = (byte[])axSTPadCapt1.RSASign(2, 0, 0);
if (btSignature == null)
{
    MessageBox.Show(String.Format("Error"));
    return;
}
```

8.77.1.2 Implementation in Visual Basic

```
Dim btSignature As Byte() = AxSTPadCapt1.RSASign(2, 0, 0)
If btSignature Is Nothing Then
    MsgBox("Error")
    Exit Sub
End If
```

8.77.2 STPadLib.dll

Available from Version 8.0.26. The status described is available from Version 8.4.3.

```
LONG STRSASign(BYTE* pbtSignature, LONG* pnSize, RSASCHEME nScheme, HASHVALUE
nHashValue, LONG nOptions)

LONG STRSASignPw(BYTE* pbtSignature, LONG* pnSize, RSASCHEME nScheme, HASHVALUE
nHashValue, LONG nOptions, LPCWSTR szSignPassword)
```

The `RSASCHEME` enumeration is defined as follows:

```
kNoHashOID = 0,
kPKCS1_V1_5 = 1,
kPSS = 2
```

The `HASHVALUE` enumeration is defined as follows:

```
kCombination = 0,
kHash1 = 1,
kHash2 = 2
```

The following values defined in the header file can be used for the `nOptions` parameter:

```
#define STPAD_RSA_LITTLEENDIAN      0x01
#define STPAD_RSA_PKCS7             0x02
#define STPAD_RSA_INCLUDECHAIN      0x04
```

8.77.2.1 Implementation in C++

```
long nSize = 0;
long nResult = STRSASign(NULL, &nSize, kPSS, kCombination, 0);
BYTE* pbtSignature = NULL;
if (nResult == 0)
{
    pbtSignature = new BYTE[nSize];
    nResult = STRSASign(pbtSignature, &nSize, kPSS, kCombination, 0);
}
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

8.77.3 STPadLibNet.dll

Available from Version 8.0.26. The status described is available from Version 8.4.3.

```
byte[] RSASign(signotec.STPadLibNet.RSAScheme scheme, HashValue hashValue,
signotec.STPadLibNet.SignFlag options)
```

```
byte[] RSASignPw(signotec.STPadLibNet.RSAScheme scheme, HashValue hashValue,
signotec.STPadLibNet.SignFlag options, System.Security.SecureString
signPassword)
```

```
Function RSASign(ByVal scheme As signotec.STPadLibNet.RSAScheme, ByVal
hashValue As signotec.STPadLibNet.HashValue, ByVal options As
signotec.STPadLibNet.SignFlag) As Byte()

Function RSASignPw(ByVal scheme As signotec.STPadLibNet.RSAScheme, ByVal
hashValue As signotec.STPadLibNet.HashValue, ByVal options As
signotec.STPadLibNet.SignFlag, ByVal signPassword as
System.Security.SecureString) As Byte()
```

The `RSAScheme` enumeration is defined as follows:

```
NoOID = 0,
PKCS1_V1_5 = 1,
PSS = 2
```

The `HashValue` enumeration is defined as follows:

```
Combination = 0,
Hash1 = 1,
Hash2 = 2
```

The `SignFlag` enumeration is defined as follows:

```
None = 0x00,
LittleEndian = 0x01,
PKCS7 = 0x02,
IncludeChain = 0x04
```

8.77.3.1 Implementation in C#

```
byte[] signature = null;
try
{
    signature = stPad.RSASign(RSAScheme.PSS, HashValue.Combination,
                              SignFlag.None);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.77.3.2 Implementation in Visual Basic

```
Dim signature() As Byte
Try
    signature = STPad.RSASign(RSAScheme.PSS, HashValue.Combination,
                              SignFlag.None)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.78 RSASetSignPassword method

This method protects against unauthorised signing via `RSASign()` by keeping a password, which is required for the process, protected in the memory of the pad. The `RSASignPasswordLength` property can be used to specify a minimum password length if the certificate is stored by `RSASetSigningCert()` or `RSAGenerateSigningCert()` .

The password protection works with the Sigma model from firmware 2.13, the Zeta model from firmware 1.0, the Omega model from firmware 2.17, the Gamma model from firmware 1.32 and the Delta model from firmware 1.31.

Parameter	Values	I/O	Description

`LPCWSTR bstrNewPassword`
`LPCWSTR szNewPassword`
`SecureString newPassword`
`ByVal newPassword As SecureString`
	`NULL`	I	Password protection is cancelled. Not possible if a minimum password length has been specified.

	max. 32 characters	I	New password in the form of a hexadecimal character string (all digits and the letters a – e are valid; upper and lower case are ignored)

`LPCWSTR bstrOldPassword`
`LPCWSTR szOldPassword`
`SecureString oldPassword`
`ByVal oldPassword As SecureString`
	`NULL`	I	The device is not password protected as yet.

	max. 32 characters	I	Previous password in the form of a hexadecimal character string (all digits and the letters a – e are valid; upper and lower case are ignored)
`LONG nMaxWrongEntries`	`0`	I	Infinite number of wrong entries are possible when calling `RSASignPw()`
	`1 – 10`	I	Number of permitted wrong entries in succession when calling `RSASignPw()` before the signing certificate is blocked. Once exceeded, signing is only possible again after a re-import by `RSASetSigningCert()` or regeneration by `RSAGenerateSigningCert()`. Blank passwords or the use of `RSASign()` do not increment the counter.
Return value	**Values**	**Description**	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	

8.78.1 STPadCapt.ocx

Available from Version 8.4.3.

```
LONG RSASetSignPassword(LPCWSTR bstrNewPassword, LPCWSTR bstrOldPassword, LONG nMaxWrongEntries)
```

8.78.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.RSASetSignPassword("0e", "", 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nReturn);
```

8.78.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1. RSASetSignPassword("0e", "", 0)
If nReturn < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.78.2 STPadLib.dll

Available from Version 8.4.3.

```
LONG STRSASetSignPassword(LPCWSTR szNewPassword, LPCWSTR szOldPassword, LONG
nMaxWrongEntries)
```

8.78.2.1 Implementation in C++

```
LONG nRc = STRSASetSignPassword(L"0e", L"", 0);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
```

8.78.3 STPadLibNet.dll

Available from Version 8.4.3.

```
void RSASetSignPassword(System.Security.SecureString newPassword,
System.Security.SecureString oldPassword, long maxWrongEntries)
```

```
Sub RSASetSignPassword(ByRef newPassword As System.Security.SecureString, ByRef
oldPassword As System.Security.SecureString, ByRef maxWrongEntries As long)
```

8.78.3.1 Implementation in C#

```
SecureString passwordNew = new SecureString();
passwordNew.AppendChar('0');
passwordNew.AppendChar('e');
SecureString passwordOld = new SecureString();
try
{
    stPad.RSASetSignPassword(passwordNew, passwordOld, 0);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.78.3.2 Implementation in Visual Basic

```
Dim passwordNew As SecureString = New SecureString()
passwordNew.AppendChar('0')
passwordNew.AppendChar('e')
Dim passwordOld As SecureString = New SecureString()
Try
    STPad.RSASetSignPassword(passwordNew, passwordOld, 0)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.79 RSASetEncryptionCert/RSASetEncryptionCertPw method

This method imports a public X.509 certificate and permanently stores the public RSA key and the certificate ID ('Issuer' and 'Serial Number') in the signature device. This key is used for the encryption of biometric data (see also `RSAGetSignData()`).

It is possible to disable this functionality permanently inside the device. It can also be protected with a password. Please refer to your contact at signotec as required.

With the Sigma model this method only works from firmware 1.16. With the Omega model, it only works from firmware 1.25.

Parameter	Values	I/O	Description

`VARIANT& vaCert`
`BYTE* pbtCert`
`X509Certificate2 cert`
`string cert`
`ByVal cert As X509Certificate2`
`ByVal cert As String`
	`!= NULL`	I	Public X.509 certificate from the memory or a file; if the device property "Supports4096BitKeys" has been set, the pad will support all multiples of 8 within the range of `1024` to `4096` as certificate length. Otherwise only the values `1024` and `2048` bit are supported. See also `DeviceGetCapabilities()`.

	`NULL`	I	The key is deleted from the signature device without storing a new key; subsequently, it is no longer possible to call up the `RSAGetSignData()` method.
`LONG nSize`	`0`	I	The `pbtCert` pointer is a `WCHAR*` type and points to the certificate file path or URL.
	`> 0`	I	Size of transferred byte array

`BSTR bstrDevicePassword`
`LPCWSTR szDevicePassword`
`SecureString devicePassword`
`ByVal devicePassword As SecureString`
	max. 32 characters	I	Password of the device (if it is password protected); please refer to your contact at signotec for details

Return value	**Values**	**Description**	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	

8.79.1 STPadCapt.ocx

Available from Version 8.0.26. The status described is available from Version 8.2.0.

```
LONG RSASetEncryptionCert(VARIANT& vaCert)
```

```
LONG RSASetEncryptionCertPw(VARIANT& vaCert, BSTR bstrDevicePassword)
```

Note: The `vaCert` parameter must contain a byte array or a string.

8.79.1.1 Implementation in C#

Work in the memory:

```
X509Certificate2 cert = new X509Certificate2(@"C:\Cert.cer");
int nResult = axSTPadCapt1.RSASetEncryptionCert
                                        (cert.Export(X509ContentType.Cert));
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

Work with files:

```
int nResult = axSTPadCapt1.RSASetEncryptionCert(@"C:\Cert.cer");
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.79.1.2 Implementation in Visual Basic

Work in the memory:

```
Dim cert As X509Certificate2 = New X509Certificate2("C:\Cert.cer")
Dim nResult As Integer = AxSTPadCapt1.RSASetEncryptionCert _
                                          (cert.Export(X509ContentType.Cer))
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

Work with files:

```
Dim nResult As Integer = AxSTPadCapt1.RSASetEncryptionCert("C:\Cert.cer")
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.79.2 STPadLib.dll

Available from Version 8.0.26. The status described is available from Version 8.2.0.

```
LONG STRSASetEncryptionCert(BYTE* pbtCert, LONG nSize)
```

```
LONG STRSASetEncryptionCertPw(BYTE* pbtCert, LONG nSize, LPCWSTR
szDevicePassword)
```

8.79.2.1 Implementation in C++

Work in the memory:

```
long nResult = STRSASetEncryptionCert(&btCert, sizeof(btCert));
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

Work with files:

```
long nResult = STRSASetEncryptionCert((BYTE*)L"C:\\Cert.cer", 0);
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

8.79.3 STPadLibNet.dll

Available from Version 8.0.26. The status described is available from Version 8.4.2.

```
void RSASetEncryptionCert
(System.Security.Cryptography.X509Certificates.X509Certificate2 cert)

void RSASetEncryptionCert(string cert)

void RSASetEncryptionCert()

void RSASetEncryptionCertPw
(System.Security.Cryptography.X509Certificates.X509Certificate2 cert,
System.Security.SecureString devicePassword)

void RSASetEncryptionCertPw(string cert, System.Security.SecureString
devicePassword)

void RSASetEncryptionCertPw(System.Security.SecureString devicePassword)

Sub RSASetEncryptionCert(ByVal cert As
System.Security.Cryptography.X509Certificates.X509Certificate2)

Sub RSASetEncryptionCert(ByVal cert As System.Security.SecureString)

Sub RSASetEncryptionCert()

Sub RSASetEncryptionCertPw(ByVal cert As
System.Security.Cryptography.X509Certificates.X509Certificate2, ByVal
devicePassword As System.Security.SecureString)

Sub RSASetEncryptionCertPw(ByVal cert As String, ByVal devicePassword As
System.Security.SecureString)

Sub RSASetEncryptionCertPw(ByVal devicePassword As
System.Security.SecureString)
```

8.79.3.1 Implementation in C#

Work in the memory:

```
try
{
    stPad.RSASetEncryptionCert(new X509Certificate2(@"C:\Cert.cer"));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

Work with files:

```
try
{
    stPad.RSASetEncryptionCert(@"C:\Cert.cer");
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.79.3.2 Implementation in Visual Basic

Work in the memory:

```
Try
    STPad.RSASetEncryptionCert(New X509Certificate2("C:\Cert.cer"))
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

Work with files:

```
Try
    STPad.RSASetEncryptionCert("C:\Cert.cer")
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.80 RSAGetEncryptionCertId method

This method exports the ID of the public key that is stored in the signature device and used for the encryption of biometric data. The ID is a character string consisting of the 'Issuer' and 'Serial Number' of the certificate in hexadecimal representation that has been transferred when saving the key, e.g., 'example certificate (ff 0a)'

With the Sigma model this method only works from firmware 1.16. With the Omega model, it only works from firmware 1.25.

Parameter	Values	I/O	Description
LPCWSTR szId	NULL	I	The method returns the length of the ID in the pnStringLength parameter
	!= NULL	I/O	Array in which the ID is to be written; if the array is too small, the end characters are cut off
LONG* pnStringLength	>= 0	I/O	Length of the ID string incl. terminated 0 or size of the szId array in bytes
Return value	**Values**	**Description**	
BSTR	empty	Error	
	other	ID of the public key	
LONG	0	Method was executed successfully	
	< 0	Error	
String	!= NULL	ID of the public key	

8.80.1 STPadCapt.ocx

Available from Version 8.0.26.

```
BSTR RSAGetEncryptionCertId()
```

8.80.1.1 Implementation in C#

```
string strId = = axSTPadCapt1.RSAGetEncryptionCertId();
if (strId  == null)
    MessageBox.Show(String.Format("Error");
else
    MessageBox.Show(String.Format("The cert ID is: {0}", strId));
```

8.80.1.2 Implementation in Visual Basic

```
Dim strId As String = AxSTPadCapt1.RSAGetEncryptionCertId()
If btSignature Is Nothing Then
    MsgBox("Error")
Else
    MsgBox("The cert ID is: " & strId)
End If
```

8.80.2 STPadLib.dll

Available from Version 8.0.26.

```
LONG STRSAGetEncryptionCertId(LPCWSTR szId, LONG* pnStringLength)
```

8.80.2.1 Implementation in C++

```
long nLen = 0;
long nResult = STRSAGetEncryptionCertId(NULL, &nLen);
if (nResult > 0)
{
    WCHAR* szId = new WCHAR[nLen / sizeof(WCHAR)];
    nResult = STRSAGetEncryptionCertId(szId, &nLen);
    if (nResult > 0)
    {
        WCHAR* szText = new WCHAR[nLen / sizeof(WCHAR) + 64];
        swprintf_s(szText, nLen / sizeof(WCHAR) + 64,
                   L" The cert ID is: %s", szId);
        AfxMessageBox(szText);
        delete [] szText;
    }
    delete [] szId;
 }
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

8.80.3 STPadLibNet.dll

Available from Version 8.0.26.

```
string RSAGetEncryptionCertId()
```

```
Function RSAGetEncryptionCertId() As String
```

8.80.3.1 Implementation in C#

```
try
{
    string strId = stPad.RSAGetEncryptionCertId();
    MessageBox.Show(String.Format("The cert ID is: {0}", strId));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.80.3.2 Implementation in Visual Basic

```
Try
    Dim strId As String = STPad.RSAGetEncryptionCertId()
    MsgBox("The cert ID is: " & strId)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.81 RSAGetSignData method

This method returns the digitalised signature in RSA encrypted SignData format. Encryption is carried out in the signature device using the key that has been saved in the device using the `RSASetEncryptionCert()` method.

This SignData format cannot be visualised by other signotec components, such as signview.dll; rather, it must first be decrypted again using the `RSADecryptSignData()` method. In order to access data directly in conventional SignData format, please use the `SignatureGetSignData()` method.

The data is returned in compressed format by default and consequently hash 2 can only be calculated on the basis of the biometric data using the `RSAExtractExtraData()` method. In order to be able to calculate hash 2 independently, data can also be returned in uncompressed format; please note the sample code in this respect.

This method discards a hash 1 that was calculated with the `RSACreateDisplayHash()` method and ends content signing. The signing of data using the `RSASign()` method must therefore take place beforehand.

With the Sigma model this method only works from firmware 1.16. With the Omega model, it only works from firmware 1.25.

Parameter	Values	I/O	Description
`BYTE* pbtSignData`	`NULL`	I	The method returns the required size of the array in the `pnSize` parameter.
	other	I/O	Array (in the required size) in which the SignData is written; `pnSize` must correspond to the value returned for the previous call.
`LONG* pnSize`	`> 0`	I/O	Size of the array (in bytes) in which the SignData is to be written

LONG nOptions SignDataGetFlag options ByVal options As SignDataGetFlag	Bitmask containing one or more hexadecimal values from the following list:		
	0x01	I	The data is returned in uncompressed format

Return value	Values	Description
VARIANT	empty	Error
	other	Signature in SignData format as a byte array
LONG	0	Method was executed successfully
	< 0	Error
byte[] Byte()	!= NULL	Signature in SignData format

8.81.1 STPadCapt.ocx

Available from Version 8.0.26.

```
VARIANT RSAGetSignData(LONG nOptions)
```

8.81.1.1 Implementation in C#

Reading compressed SignData:

```
byte[] btSignData = (byte[])axSTPadCapt1.RSAGetSignData(0);
if (btSignData == null)
{
    MessageBox.Show(String.Format("Error"));
    return;
}
```

Reading uncompressed SignData and extraction of data from which hash 2 has been calculated:

```
byte[] btSignData = (byte[])axSTPadCapt1.RSAGetSignData(1);
if (btSignData == null)
{
    MessageBox.Show(String.Format("Error"));
    return;
}
int nOffset = BitConverter.ToInt16(btSignData, 28) + 32;
nOffset += BitConverter.ToInt32(btSignData, nOffset);
byte[] btHash2Data = new byte[btSignData.Length - nOffset];
Array.Copy(btSignData, nOffset, btHash2Data, 0, btHash2Data.Length);
```

8.81.1.2 Implementation in Visual Basic

Reading compressed SignData:

```
Dim btSignData As Byte() = AxSTPadCapt1.RSAGetSignData(0)
If btSignData Is Nothing Then
    MsgBox("Error")
    Exit Sub
End If
```

Reading uncompressed SignData and extraction of data from which hash 2 has been calculated:

```
Dim btSignData As Byte() = AxSTPadCapt1.RSAGetSignData(1)
If btSignData Is Nothing Then
    MsgBox("Error")
    Exit Sub
End If
Dim nOffset As Integer = BitConverter.ToInt16(btSignData, 28) + 32
nOffset += BitConverter.ToInt32(btSignData, nOffset)
Dim btHash2Data(btSignData.Length - nOffset) As Byte
Array.Copy(btSignData, nOffset, btHash2Data, 0, btHash2Data.Length)
```

8.81.2 STPadLib.dll

Available from Version 8.0.26.

```
LONG STRSAGetSignData(BYTE* pbtSignData, LONG* pnSize, LONG nOptions)
```

The following values defined in the header file can be used for the `nOptions` parameter:

```
#define STPAD_GETSIGNDATA_UNCOMPRESSED    0x01
```

8.81.2.1 Implementation in C++

Reading compressed SignData:

```
long nSize = 0;
long nResult;
nResult = STRSAGetSignData(NULL, &nSize, 0);
BYTE* pbtSignData = NULL;
if (nResult == 0)
{
    pbtSignData = new BYTE[nSize];
    nResult = STRSAGetSignData(pbtSignData, &nSize,
                               STPAD_GETSIGNDATA_UNCOMPRESSED);
}
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

Reading uncompressed SignData and extraction of data from which hash 2 has been calculated:

```
long nSize = 0;
long nResult;
nResult = STRSAGetSignData(NULL, &nSize, STPAD_GETSIGNDATA_UNCOMPRESSED);
BYTE* pbtSignData = NULL;
if (nResult == 0)
{
    pbtSignData = new BYTE[nSize];
    nResult = STRSAGetSignData(pbtSignData, &nSize,
                               STPAD_GETSIGNDATA_UNCOMPRESSED);
}
if (nResult < 0)
    AfxMessageBox(L"Error!");
int nOffset = *((short*)(&btSignData[28])) + 32;
nOffset += *((int*)(&btSignData[nOffset]));
BYTE* pbtHash2Data = new BYTE[nSize - nOffset];
memcpy(pbtHashData, &btSignData[nOffset], nSize - nOffset);
```

8.81.3 STPadLibNet.dll

Available from Version 8.0.26.

```
byte[] RSAGetSignData(signotec.STPadLibNet.SignDataGetFlag options)
```

```
Function RSAGetSignData() As Byte(ByVal options As
signotec.STPadLibNet.SignDataGetFlag)
```

The `SignDataGetFlag` enumeration is defined as follows:

```
None = 0x00,
Uncompressed = 0x01
```

8.81.3.1 Implementation in C#

Reading compressed SignData:

```
byte[] btSignData;
try
{
    btSignData = stPad.RSAGetSignData(SignDataGetFlag.None);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

Reading uncompressed SignData and extraction of data from which hash 2 has been calculated:

```
byte[] btSignData;
try
{
    btSignData = stPad.RSAGetSignData(SignDataGetFlag.Uncompressed);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
int nOffset = BitConverter.ToInt16(btSignData, 28) + 32;
nOffset += BitConverter.ToInt32(btSignData, nOffset);
byte[] btHash2Data = new byte[btSignData.Length - nOffset];
Array.Copy(btSignData, nOffset, btHash2Data, 0, btHash2Data.Length);
```

8.81.3.2 Implementation in Visual Basic

Reading uncompressed SignData:

```
Dim btSignData() As Byte
Try
    btSignData = STPad.RSAGetSignData(SignDataGetFlag.Uncompressed)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

Reading uncompressed SignData and extraction of data from which hash 2 has been calculated:

```
Dim btSignData() As Byte
Try
    btSignData = STPad.RSAGetSignData(SignDataGetFlag.Uncompressed)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
Dim nOffset As Integer = BitConverter.ToInt16(btSignData, 28) + 32
nOffset += BitConverter.ToInt32(btSignData, nOffset)
Dim btHash2Data(btSignData.Length - nOffset) As Byte
Array.Copy(btSignData, nOffset, btHash2Data, 0, btHash2Data.Length)
```

8.82 RSADecryptSignData method

This method decrypts RSA encrypted SignData as delivered by `RSAGetSignData()` and converts it into the conventional SignData format, as expected by other signotec components such as signview.dll for example. Additional data, which is included where appropriate, but cannot be converted, can also be decrypted.

Note: This method should not be used to read out the remaining data. Please use the `RSAExtractExtraData()` method instead.

Parameter	Values	I/O	Description
`BYTE* pbtDecrypted`	`NULL`	I	The method returns the required size of the array in the pnDecryptedSize parameter
	`!= NULL`	I/O	Array in the required size in which the decrypted SignData is written; pnDecryptedSize must correspond to the value returned for the previous call
`LONG pnDecryptedSize`	`> 0`	I/O	Size of the array in which the decrypted SignData is to be written
`VARIANT& vaEncrypted`	`!= NULL`	I	Byte array with RSA encrypted SignData

`BYTE* pbtEncrypted`
`byte[] encrypted`
`ByVal encrypted As Byte()`
	`NULL`	I	Obsolete, please use the `RSAExtractExtraData()` method instead

`LONG nEncryptedSize`	`>= 0`	I	Size of the array that is referenced by `pbtEncrypted` in bytes
`VARIANT& vaCert`	`!= NULL`	I	Certificate in PKCS#12 format from the memory or a file

`BYTE* pbtCert`
`X509Certificate2 cert`
`string cert`
`ByVal cert As X509Certificate2`
`ByVal cert As String`
	`NULL`	I	Obsolete, please use the `RSAExtractExtraData()` method instead

LONG nCertSize	0	I	The pbtCert pointer is a WCHAR* type and points to the certificate file path or URL.
	> 0	I	Size of transferred byte array
BSTR bstrPassword LPCWSTR szPassword SecureString password ByVal password As SecureString	all	I	Password used to fetch the private key
LONG* pnExtraData SignDataDecryptFlag extraData ByVal extraData As SignDataDecryptFlag	A bitmask, which may contain one or more hexadecimal values from the following list and which defines which of the RSA encrypted SignData shall also be decrypted in order to retrieve them when the method RSAExtractExtraData() is called subsequently. On method return, it indicates which additional data have been decrypted and temporarily stored. For details, please see the description of the RSAExtractExtraData() method.		
	0x01	I/O	Timestamp
	0x02	I/O	Serial number
	0x04	I/O	Key source
	0x08	I/O	Firmware version
	0x10	I/O	Hash 1
	0x20	I/O	Hash 2
Return value	**Values**	**Description**	
VARIANT	empty	Error	
	other	Decrypted SignData	
LONG	0	Method was executed successfully	
	< 0	Error	
byte[] Byte()	!= NULL	Decrypted SignData	

8.82.1 STPadCapt.ocx

Available from Version 8.0.26. The status described is available from Version 8.5.1.

```
VARIANT RSADecryptSignData(VARIANT& vaEncrypted, VARIANT& vaCert, BSTR
bstrPassword, LONG* pnExtraData)
```

Note: The vaEncrypted parameter must contain a byte array and the vaCert parameter must contain a byte array or a string.

8.82.1.1 Implementation in C#

Work in the memory:

```
X509Certificate2 cert = new X509Certificate2(@"C:\Cert.pfx", "password",
                                              X509KeyStorageFlags.Exportable);
int nExtraData = 0x3f;
byte[] btSignData = axSTPadCapt1.RSADecryptSignData(btEncrypted,
                                    cert.Export(X509ContentType.Pkcs12),
                                    "password", ref nExtraData);
if (btSignData == null)
{
    MessageBox.Show(String.Format("Error"));
    return;
}
```

Work with files:

```
int nExtraData = 0x3f;
byte[] btSignData = axSTPadCapt1.RSADecryptSignData(btEncrypted,
                                @"C:\Cert.pfx", "password", ref nExtraData);
if (btSignData == null)
{
    MessageBox.Show(String.Format("Error"));
    return;
}
```

The remaining data can then be queried using the `RSAExtractExtraData()` method.

8.82.1.2 Implementation in Visual Basic

Work in the memory:

```
Dim cert As X509Certificate2 = New X509Certificate2("C:\Cert.pfx", _
                               "password", X509KeyStorageFlags.Exportable)
Dim nExtraData As Integer = &H3F
Dim btSignData() As Byte
btSignData = AxSTPadCapt1.RSADecryptSignData(btEncrypted, _
             cert.Export(X509ContentType.Pkcs12), "password", nExtraData)
If btSignData Is Nothing Then
    MsgBox("Error")
    Exit Sub
End If
```

Work with files:

```
Dim nExtraData As Integer = &H3F
Dim btSignData() As Byte
btSignData = AxSTPadCapt1.RSADecryptSignData(btEncrypted, _
                                             "C:\Cert.pfx", "password",
                                             nExtraData)
If btSignData Is Nothing Then
    MsgBox("Error")
    Exit Sub
End If
```

The remaining data can then be queried using the `RSAExtractExtraData()` method.

8.82.2 STPadLib.dll

Available from Version 8.0.26. The status described is available from Version 8.5.1.

```
LONG STRSADecryptSignData(BYTE* pbtDecrypted, LONG* pnDecryptedSize, BYTE*
pbtEncrypted, LONG nEncryptedSize, BYTE* pbtCert, LONG nCertSize, LPCWSTR
szPassword, LONG* pnExtraData)
```

The following values defined in the header file can be used for the `pnExtraData` parameter:

```
#define STPAD_DECRYPTSIGNDATA_TIMESTAMP   0x01
#define STPAD_DECRYPTSIGNDATA_SERIAL      0x02
#define STPAD_DECRYPTSIGNDATA_KEYSOURCE   0x04
#define STPAD_DECRYPTSIGNDATA_FIRMWARE    0x08
#define STPAD_DECRYPTSIGNDATA_HASH1       0x10
#define STPAD_DECRYPTSIGNDATA_HASH2       0x20
#define STPAD_DECRYPTSIGNDATA_ALL         0x3F
```

8.82.2.1 Implementation in C++

Work in the memory:

```
long nSize = 0;
long nExtraData = STPAD_DECRYPTSIGNDATA_ALL;
long nResult = STRSADecryptSignData(NULL, &nSize, &btEncrypted,
                                    sizeof(btEncrypted), &btCert,
                                    sizeof(btCert), L"password", &nExtraData);
BYTE* pbtSignData = NULL;
if (nResult == 0)
{
    pbtSignData = new BYTE[nSize];
    nResult = STRSADecryptSignData(pbtSignData, &nSize, &btEncrypted,
                                    sizeof(btEncrypted), &btCert,
                                    sizeof(btCert), L"password", &nExtraData);
}
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

Work with files:

```
long nSize = 0;
long nExtraData = STPAD_DECRYPTSIGNDATA_ALL;
long nResult = STRSADecryptSignData(NULL, &nSize, &btEncrypted,
                                    sizeof(btEncrypted), (BYTE*)L"C:\\Cert.pfx",
                                    0, L"password", &nExtraData);
BYTE* pbtSignData = NULL;
if (nResult == 0)
{
    pbtSignData = new BYTE[nSize];
    nResult = STRSADecryptSignData(pbtSignData, &nSize, &btEncrypted,
                                    sizeof(btEncrypted), (BYTE*)L"C:\\Cert.pfx",
                                    0, L"password", &nExtraData);
}
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

The remaining data can then be queried using the `RSAExtractExtraData()` method.

8.82.3 STPadLibNet.dll

Available from Version 8.0.26. The status described is available from Version 8.5.1.

```
byte[] RSADecryptSignData(byte[] encrypted,
System.Security.Cryptography.X509Certificates.X509Certificate2 cert,
System.Security.SecureString password, ref
signotec.STPadLibNet.SignDataDecryptFlag extraData)

byte[] RSADecryptSignData(byte[] encrypted, string cert,
System.Security.SecureString password, ref
signotec.STPadLibNet.SignDataDecryptFlag extraData)

byte[] RSADecryptSignData(ref signotec.STPadLibNet.SignDataDecryptFlag
extraData)

Function RSADecryptSignData(ByVal encrypted As Byte(), ByVal cert As
System.Security.Cryptography.X509Certificates.X509Certificate2, ByVal password
As System.Security.SecureString, ByRef extraData As
signotec.STPadLibNet.SignDataDecryptFlag) As Byte()

Function RSADecryptSignData(ByVal encrypted As Byte(), ByVal cert As String,
ByVal password As System.Security.SecureString, ByRef extraData As
signotec.STPadLibNet.SignDataDecryptFlag) As Byte()

Function RSADecryptSignData(ByRef encrypted As
signotec.STPadLibNet.SignDataDecryptFlag) As Byte()
```

The `SignDataDecryptFlag` enumeration is defined as follows:

```
None = 0x00,
Timestamp = 0x01,
Serial = 0x02,
KeySource = 0x04,
Firmware = 0x08,
Hash1 = 0x10,
Hash2 = 0x20,
All = 0x3f
```

8.82.3.1 Implementation in C#

Work in the memory:

```
SecureString password = new SecureString();
password.AppendChar('p');
password.AppendChar('w');
X509Certificate2 cert = new X509Certificate2(@"C:\Cert.pfx", password,
                                             X509KeyStorageFlags.Exportable);
SignDataDecryptFlag extraData = SignDataDecryptFlag.All;
byte[] signData = null;
try
{
    signData = stPad.RSADecryptSignData(encrypted, cert, password, ref
                                        extraData);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
password.Dispose();
```

Work with files:

```
SecureString password = new SecureString();
password.AppendChar('p');
password.AppendChar('w');
SignDataDecryptFlag extraData = SignDataDecryptFlag.All;
byte[] signData = null;
try
{
    signData = stPad.RSADecryptSignData(encrypted, @"C:\Cert.pfx",
                                        password, ref extraData);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

The remaining data can then be queried using the `RSAExtractExtraData()` method.

8.82.3.2 Implementation in Visual Basic

Work in the memory:

```
Dim password As SecureString = New SecureString()
password.AppendChar("p")
password.AppendChar("w")
Dim cert As X509Certificate2 = New X509Certificate2("C:\Cert.pfx", _
                                password, X509KeyStorageFlags.Exportable)
Dim extraData As SignDataDecryptFlag = SignDataDecryptFlag.All;
Dim signData As Byte() = Nothing
Try
    signData = STPad.RSADecryptSignData(encrypted, cert, password, _
                                        extraData)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
password.Dispose()
```

Work with files:

```
Dim password As SecureString = New SecureString()
password.AppendChar("p")
password.AppendChar("w")
Dim extraData As SignDataDecryptFlag = SignDataDecryptFlag.All;
Dim signData As Byte() = Nothing
Try
    signData = STPad.RSADecryptSignData(encrypted, "C:\Cert.pfx", _
                                        password, extraData)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
password.Dispose()
```

The remaining data can then be queried using the `RSAExtractExtraData()` method.

8.83 RSAExtractExtraData method

This method extracts the extra data contained in the RSA encrypted SignData, alongside the actual biometric data, as delivered by `RSAGetSignData()`. This data is partially encrypted.

If only the unencrypted contained data are to be extracted (serial number, key source, device options, firmware version, hash 2), this method can be called with RSA encrypted SignData. If encrypted contained data are to be extracted as well, it is necessary to call the `RSADecryptSignData()` method in advance. `RSAExtractExtraData()` must then be called without SignData and it will return the data stored temporarily by the `RSADecryptSignData()` method. Calling without SignData without previously calling the `RSADecryptSignData()` method results in an error.

Note: In all cases, calling this method deletes any data stored temporarily by the `RSADecryptSignData()` method.

Parameter	Values	I/O	Description

`LONG nHash2Algorithm`
`HashAlgo hash2Algorithm`
`ByVal hash2Algorithm As HashAlgo`
	Algorithm that is to be used to calculate the biometric hash (hash 2); as it is contained in the SignData in encrypted form and decrypted by `RSADecryptSignData()`, is only has to be transmitted when calling with SignData.
	It must always correspond to the algorithm that was used for the original signing procedure, as otherwise it is not possible to check the signature.		

	`0`	I	SHA-1
	`1`	I	SHA-256
	`2`	I	SHA-512
`EXTRADATA* pExtraData`	`!= NULL`	I/O	Pointer to structure to which the extracted data are written (for details, see below); for member `hash2`, the member `hashalgo` must be filled when calling with SignData (for meaning, see above)

`VARIANT& vaSignData`
`BYTE* pbtSignData`
`byte[] signData`
`ByVal signData As Byte()`
	`!= NULL`	I	Byte array with RSA encrypted SignData

	`NULL`	I	The method returns all extra data stored temporarily by a previous call of the `RSADecryptSignData()` method
`LONG nSignDataSize`	`0`	I	Size of the array that is referenced by `pbtSignData` in bytes

Return value	Values	Description
`VARIANT`	empty	Error
	other	Exported extra data as byte array (for details, see below)
`LONG`	`0`	Method was executed successfully
	`< 0`	Error
`ExtraData`	`!= NULL`	Class instance from which the extra data can be read (for details see below)

The following information may be contained in the extra data:

Information	Description
Timestamp	Time stamp of the signature in seconds since 1 January 1970 in UTC
Serial number	Serial number of the signature device, with which the data was captured

Key source	Origin of the key used for signing 0=key generated in the device 1=externally generated key loaded in the device 2=key saved as default setting
Device options	Device options of the signature device; please refer to your contact at signotec for details.
Firmware version	Firmware version of the signature device, with which the data was captured.
Hash 1	Hash 1, which has been specified by `RSASetHash()` at the start of the signature process.
Hash 2	Hash 2 of the biometric data; this hash is not contained directly in the SignData but is calculated from the data.

How this information can be read from the extra data is described in the following sections.

8.83.1 STPadCapt.ocx

Available from Version 8.5.1.

```
VARIANT RSAExtractExtraData(LONG nHash2Algorithm, VARIANT& vaSignData)
```

Note: The `vaHash` parameter must contain a byte array.

The returned byte array always has the same structure. The individual pieces of information must therefore be interpreted on the basis of the following offsets as shown. If the particular piece of information has not been extracted, the specified invalid value will be contained.

Information	Offset	Data type	Invalid value
Timestamp	0	8 bytes, little endian, signed	0xFFFFFFFFFFFFFFFF (-1)
Serial number	8	4 bytes, little endian, unsigned	0xFFFFFFFF (4294967295)
Key source	12	4 bytes, little endian, unsigned	0xFFFFFFFF (4294967295)
Firmware version			
Major	20	4 bytes, little endian, unsigned	0xFFFFFFFF (4294967295)
Minor	24	4 bytes, little endian, unsigned	0xFFFFFFFF (4294967295)
Hash 1			
Algorithm	28	4 bytes, little endian, unsigned	0xFFFFFFFF (4294967295)
Hash	32	64 bytes, byte array; the actual length of the hash is dependent on the algorithm: 0 (SHA-1) = 20 bytes 1 (SHA-256) = 32 bytes 2 (SHA-512) = 64 bytes	-
Hash 2			
Algorithm	96	4 bytes, little endian, unsigned	0xFFFFFFFF (4294967295)

	Hash	100	64 bytes, byte array; the actual length of the hash is dependent on the algorithm: 0 (SHA-1) = 20 bytes 1 (SHA-256) = 32 bytes 2 (SHA-512) = 64 bytes	–

8.83.1.1 Implementation in C#

Use without decryption:

```
byte[] extraData = (byte[])axSTPadCapt1.RSAExtractExtraData(1, signData);
if (extraData == null)
    MessageBox.Show(String.Format("Error");
```

Use after previous decryption by `RSADecryptSignData()`:

```
byte[] extraData = (byte[])axSTPadCapt1.RSAExtractExtraData(0, null);
if (extraData == null)
    MessageBox.Show(String.Format("Error");
```

The extra data can then be read as follows:

```
DateTime dateTime;
Int64 timeStamp = BitConverter.ToInt64(extraData, 0);
if (timestamp != -1)
    dateTime = new DateTime(1970, 1, 1).AddSeconds((double)timestamp);

UInt32 serial = BitConverter.ToUInt32(extraData, 8);

UInt32 keySource = BitConverter.ToUInt32(extraData, 12);

Version fwVersion = null;
UInt32 fwMajor = BitConverter.ToUInt32(extraData, 20);
UInt32 fwMinor = BitConverter.ToUInt32(extraData, 24);
if ((fwMajor != UInt32.MaxValue) && (fwMinor != UInt32.MaxValue))
    fwVersion = new Version(fwMajor, fwMinor);

byte[] hash1 = null;
switch (BitConverter.ToUInt32(extraData, 28))
{
    case 0:
        hash1 = new byte[20];
        break;
    case 1:
        hash1 = new byte[32];
        break;
    case 2:
        hash1 = new byte[64];
        break;
}
if (hash1 != null)
     Array.Copy(extraData, 32, hash1, 0, hash1.Length);

byte[] hash2 = null;
switch (BitConverter.ToUInt32(extraData, 96))
{
    case 0:
        hash2 = new byte[20];
        break;
    case 1:
        hash2 = new byte[32];
        break;
    case 2:
        hash2 = new byte[64];
        break;
}
if (hash2 != null)
     Array.Copy(extraData, 100, hash2, 0, hash2.Length);
```

8.83.1.2 Implementation in Visual Basic

Use without decryption:

```
Dim extraData As Byte = axSTPadCapt1.RSAExtractExtraData(1, signData)
If extraData Is Nothing Then
    MsgBox("Error ")
End If
```

Use after previous decryption by `RSADecryptSignData()`:

```
Dim extraData As Byte = axSTPadCapt1.RSAExtractExtraData(0, Nothing)
If extraData Is Nothing Then
    MsgBox("Error ")
End If
```

The extra data can then be read as follows:

```
Dim dateTime As DateTime
Dim timeStamp As Int64 = BitConverter.ToInt64(extraData, 0)
If timeStamp <> -1 Then
    dateTime = New DateTime(1970, 1, 1).AddSeconds(timeStamp)
End If

Dim serial As UInt32 = BitConverter.ToUInt32(extraData, 8)

Dim keySource As UInt32 = BitConverter.ToUInt32(extraData, 12)

Dim fwVersion As Version
Dim fwMajor As UInt32 = BitConverter.ToUInt32(extraData, 20)
Dim fwMinor As UInt32 = BitConverter.ToUInt32(extraData, 24)
If ((fwMajor <> UInt32.MaxValue) And (fwMinor <> UInt32.MaxValue)) Then
    fwVersion = New Version(fwMajor, fwMinor)
End If

Dim hash1() As Byte = Nothing
Select Case BitConverter.ToUInt32(extraData, 28)
    Case 0
        hash1 = New Byte(19) {}
    Case 1
        hash1 = New Byte(31) {}
    Case 2
        hash1 = New Byte(63) {}
End Select
If Not hash1 Is Nothing Then
    Array.Copy(extraData, 32, hash1, 0, hash1.Length)
End If

Dim hash2() As Byte = Nothing
Select Case BitConverter.ToUInt32(extraData, 96)
    Case 0
        hash2 = New Byte(19) {}
    Case 1
        hash2 = New Byte(31) {}
    Case 2
        hash2 = New Byte(63) {}
End Select
If Not hash2 Is Nothing Then
    Array.Copy(extraData, 100, hash2, 0, hash2.Length)
End If
```

8.83.2 STPadLib.dll

Available from Version 8.5.1.

```
LONG STRSAExtractExtraData(EXTRADATA* pExtraData, BYTE* pbtSignData=NULL, LONG
nSignDataSize=0)
```

The individual pieces of information can be accessed in the returned `EXTRADATA` structure as follows. If the particular piece of information has not been extracted, the specified invalid value will be contained.

Information	Member	Data type	Invalid value
Timestamp	timestamp	__time64_t	STPAD_EXTRADATA_NOTIME
Serial number	serialnumber	DWORD	STPAD_EXTRADATA_NOVALUE
Key source	keysource	DWORD	STPAD_EXTRADATA_NOVALUE
Firmware version (major)	firmwaremejor	DWORD	STPAD_EXTRADATA_NOVALUE
Firmware version (minor)	firmwareminor	DWORD	STPAD_EXTRADATA_NOVALUE
Hash 1	hash1	HASH	-
Algorithm	hashalgo	HASHALGO	STPAD_EXTRADATA_NOVALUE
Hash	hash	BYTE[64]	-
Hash 2	hash2	HASH	-
Algorithm	hashalgo	HASHALGO	STPAD_EXTRADATA_NOVALUE
Hash	hash	BYTE[64]	-

The `EXTRADATA` and `HASH` structures are defined as follows:

```
typedef struct EXTRADATA
{
    __time64_t timestamp;
    DWORD serialnumber;
    DWORD keysource;
    DWORD deviceoptions;
    DWORD firmwaremajor;
    DWORD firmwareminor;
    HASH hash1;
    HASH hash2;
} EXTRADATA;

typedef struct HASH
{
    HASHALGO hashalgo;
    BYTE hash[64];
} HASH;
```

The `HASHALGO` enumeration is defined as follows:

```
enum HASHALGO
{
    kSha1 = 0,
    kSha256 = 1,
    kSha512 = 2
};
```

The invalid values are defined as follows:

```
#define STPAD_EXTRADATA_NOTIME    -1
#define STPAD_EXTRADATA_NOVALUE   0xFFFFFFFF
```

8.83.2.1 Implementation in C++

Use without decryption:

```
EXTRADATA extraData;
memset(&extraData, 0xFF, (sizeof(EXTRADATA));
extraData.hash2.hashalgo = kSha256;
long nResult = STRSAExtractExtraData(&extraData, &btSignData,
                                     sizeof(btSignData));
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

Use after previous decryption by `RSADecryptSignData()`:

```
EXTRADATA extraData;
memset(&extraData, 0xFF, (sizeof(EXTRADATA));
long nResult = STRSAExtractExtraData(&extraData);
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

8.83.3 STPadLibNet.dll

Available from Version 8.5.1.

`ExtraData RSAExtractExtraData(HashAlgo hash2Algorithm, byte[] signData)`

`ExtraData RSAExtractExtraData()`

`Function RSAExtractExtraData(ByVal hash2Algorithm As signotec.STPadLibNet.HashAlgo, ByVal signData As Byte()) As ExtraData`

`Function RSAExtractExtraData() As ExtraData`

The returned `ExtraData` class contains the individual pieces of information as properties. If the particular piece of information has not been extracted, the specified invalid value will be contained.

Information	Property	Data type	Invalid value
Timestamp	`TimeStamp`	`DateTime`	`DateTime()` (01.01.0001, 0:00)
Serial number	`SerialNumber`	`UInt32`	`0`
Key source	`KeySource`	`KeySource`	`Unknown`
Firmware version	`Firmware`	`Version`	`null` `Nothing`
Hash 1	`Hash1`	`Hash`	`null` `Nothing`
Algorithm	`Algorithm`	`HashAlgo`	-
Hash	`Data`	`byte[]` `Byte()`	-
Hash 2	`Hash1`	`Hash`	`null` `Nothing`
Algorithm	`Algorithm`	`HashAlgo`	-
Hash	`Data`	`byte[]` `Byte()`	-

The `HashAlgo` enumeration is defined as follows:

```
enum class HashAlgo
{
    SHA1 = 0,
    SHA256 = 1,
    SHA512 = 2
};
```

The `KeySource` enumeration is defined as follows:

```
enum class KeySource
{
    Unknown = -1,
    Internal = 0,
    External = 1,
    Factory = 2
};
```

8.83.3.1 Implementation in C#

Use without decryption:

```
try
{
    ExtraData extraData = stPad.RSAExtractExtraData(HashAlgo.SHA256,
                                                    signData);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

Use after previous decryption by `RSADecryptSignData()`:

```
try
{
    ExtraData extraData = stPad.RSAExtractExtraData();
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.83.3.2 Implementation in Visual Basic

Use without decryption:

```
Try
    Dim extraData As ExtraData =
                      STPad.RSAExtractExtraData(HashAlgo.SHA256, signData)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

Use after previous decryption by `RSADecryptSignData()`:

```
Try
    Dim extraData As ExtraData = STPad.RSAExtractExtraData()
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.84 PDFLoad method

This method can be used to load a PDF document in order to subsequently display it on the signature device using the `DisplaySetPDF()` method.

To be able to use this method, STPdfLib18.dll or STPdfLib16.dll must be in the application's search path or next to STPadCapt.ocx (if used).

Parameter	Values	I/O	Description
`VARIANT& vaDocument`	`!= NULL`	I	PDF document from the memory or a file

`BYTE* pbtDocument`
`byte[] document`
`string document`
`ByVal document As Byte()`
`ByVal document As String`
	`NULL`	I	The currently loaded PDF document is unloaded from the memory

`LONG nSize`	`0`	I	The `pbtDocument` pointer is a `WCHAR*` type and points to the PDF file path or URL.
	`> 0`	I	Size of transferred array in bytes

`BSTR bstrPassword`
`LPCWSTR szPassword`
`SecureString password`
`ByVal password As SecureString`
	all	I	Password of the PDF document (optional)

Return value	**Values**	**Description**	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	

8.84.1 STPadCapt.ocx

Available from Version 8.1.4.

```
LONG PDFLoad(VARIANT& vaDocument, LPCWSTR bstrPassword)
```

8.84.1.1 Implementation in C#

Work in the memory:

```
int nResult = axSTPadCapt1.PDFLoad(pdf, null);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

Work with files:

```
int nResult = axSTPadCapt1.PDFLoad(@"C:\Doc.pdf", null);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.84.1.2 Implementation in Visual Basic

Work in the memory:

```
Dim nResult As Integer = AxSTPadCapt1.PDFLoad(pdf, Nothing)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

Work with files:

```
Dim nResult As Integer = AxSTPadCapt1.PDFLoad("C:\Doc.pdf", Nothing)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.84.2 STPadLib.dll

Available from Version 8.1.4.

```
LONG STPDFLoad(BYTE* pbtDocument, LONG nSize, LPCWSTR szPassword)
```

8.84.2.1 Implementation in C++

Work in the memory:

```
int nResult = STPDFLoad(btDocument, sizeof(btDocument), NULL);
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

Work with files:

```
int nResult = STPDFLoad((BYTE*)L"C:\\Doc.pdf", 0, NULL);
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

8.84.3 STPadLibNet.dll

Available from Version 8.1.4. The status described is available from Version 8.4.2.

```
void PDFLoad()
void PDFLoad(byte[] document)
void PDFLoad(string document)
void PDFLoad(byte[] document, System.Security.SecureString password)
void PDFLoad(string document, System.Security.SecureString password)
Sub PDFLoad()
Sub PDFLoad(ByVal document As Byte())
Sub PDFLoad(ByVal document As String)
Sub PDFLoad(ByVal document As Byte(), ByVal password As
System.Security.SecureString)
Sub PDFLoad(ByVal document As String, ByVal password As
System.Security.SecureString)
```

8.84.3.1 Implementation in C#

Work in the memory:

```
try
{
    stPad.PDFLoad(pdf, (SecureString)null);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

Work with files:

```
try
{
    stPad.PDFLoad(@"C:\Doc.pdf", (SecureString)null);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.84.3.2 Implementation in Visual Basic

Work in the memory:

```
Try
    STPad.PDFLoad(pdf, DirectCast(Nothing, SecureString))
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

Work with files:

```
Try
    STPad.PDFLoad("C:\Doc.pdf", DirectCast(Nothing, SecureString))
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.85 PDFGetPageCount method

This method delivers the number of pages of the currently loaded PDF document.

Parameter	Values	I/O	Description
-	-	-	-
Return value	**Values**	**Description**	
LONG int Integer	> 0	Number of pages	
	< 0	Error (not STPadLibNet.dll)	

8.85.1 STPadCapt.ocx

Available from Version 8.1.4.

```
LONG PDFGetPageCount()
```

8.85.1.1 Implementation in C#

```
int nCount = axSTPadCapt1.PDFGetPageCount();
if (nCount < 0)
    MessageBox.Show(String.Format("Error {0}", nCount);
else
    MessageBox.Show(String.Format("The document has {0} pages.", nCount);
```

8.85.1.2 Implementation in Visual Basic

```
Dim nCount As Integer = AxSTPadCapt1.PDFGetPageCount
If nCount < 0 Then
    MsgBox("Error " & CStr(nCount))
Else
    MsgBox("The document has " & CStr(nCount) & " pages.")
End If
```

8.85.2 STPadLib.dll

Available from Version 8.1.4.

```
LONG STPDFGetPageCount()
```

8.85.2.1 Implementation in C++

```
long nCount = STPDFGetPageCount();
WCHAR szText[64];
if (nCount < 0)
    swprintf_s(szText, 64, L"Error %d", nCount);
else
    swprintf_s(szText, 64, L"The document has %d pages.", nCount);
AfxMessageBox(szText);
```

8.85.3 STPadLibNet.dll

Available from Version 8.1.4.

```
int PDFGetPageCount()

Function PDFGetPageCount() As Integer
```

8.85.3.1 Implementation in C#

```
try
{
    int nCount = stPad.PDFGetPageCount();
    MessageBox.Show(String.Format("The document has {0} pages.",
                    nCount));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.85.3.2 Implementation in Visual Basic

```
Try
    Dim nCount As Integer = STPad.PDFGetPageCount()
    MsgBox(The document has " & CStr(nCount) & " pages.")
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.86 PDFGetWidth method

This method returns the width of a page of the currently loaded PDF document.

Parameter	Values	I/O	Description
LONG nPage int page ByVal page As Integer	> 0	I	Number of the page whose width is to be returned (starting at 1)
LONG nUnit MEASUREMENTUNIT nUnit MeasurementUnit unit ByVal unit As MeasurementUnit	0	I	Pixels are used as the unit of measurement (based on the display of the signature device used)
	1	I	Millimetres are used as the unit of measurement
	2	I	Inches are used as the unit of measurement
Return value	**Values**	**Description**	
DOUBLE double Double	>= 0	Page width	
	< 0	Error (not STPadLibNet.dll)	

8.86.1 STPadCapt.ocx

Available from Version 8.1.4.

```
DOUBLE PDFGetWidth(LONG nPage, LONG nUnit)
```

8.86.1.1 Implementation in C#

```
double dWidth = axSTPadCapt1.PDFGetWidth(1, 1);
if (dWidth < 0.0)
    MessageBox.Show(String.Format("Error {0}", (int)dWidth));
else
    MessageBox.Show(String.Format("The page has a width of {0}
                                   millimetres.", dWidth));
```

8.86.1.2 Implementation in Visual Basic

```
Dim dWidth As Double = AxSTPadCapt1.PDFGetWidth(1, 1)
If dWidth < 0R Then
    MsgBox("Error " & CStr(dWidth))
Else
    MsgBox("The page has a width of " & CStr(dWidth) & " millimetres.")
End If
```

8.86.2 STPadLib.dll

Available from Version 8.1.4.

```
LONG STPDFGetWidth(LONG nPage, MEASUREMENTUNIT nUnit)
```

The `MEASUREMENTUNIT` enumeration is defined as follows:

```
kPixels = 0,
kMillimetres = 1,
kInches = 2
```

8.86.2.1 Implementation in C++

```
double dWidth = STPDFGetWidth(1, kMillimetres);
WCHAR szText[64];
if (dWidth < 0.)
    swprintf_s(szText, 64, L"Error %d", (int)dWidth);
else
    swprintf_s(szText, 64, L"The page has a width of %d millimetres.",
               dWidth);
```

8.86.3 STPadLibNet.dll

Available from Version 8.1.4.

```
double PDFGetWidth(int page, MeasurementUnit unit)
```

```
Function PDFGetWidth(ByVal page as Integer, ByVal unit As MeasurementUnit) As
Double
```

The `MeasurementUnit` enumeration is defined as follows:

```
Pixels = 0,
Millimetres = 1,
Inches = 2
```

8.86.3.1 Implementation in C#

```
try
{
    double dWidth = stPad.PDFGetWidth(1, MeasurementUnit.Millimetres);
    MessageBox.Show(String.Format("The page has a width of {0}
                                   millimetres.", dWidth));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.86.3.2 Implementation in Visual Basic

```
Try
    Dim dWidth As Double
    dHeight = STPad.PDFGetWidth(1, MeasurementUnit.Millimetres)
    MsgBox("The page has a width of " & CStr(dWidth) & " millimetres.")
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.87 PDFGetHeight method

This method returns the height of a page of the currently loaded PDF document.

Parameter	Values	I/O	Description
LONG nPage int page ByVal page As Integer	> 0	I	Number of the page whose height is to be returned (starting at 1)
LONG nUnit MEASUREMENTUNIT nUnit MeasurementUnit unit ByVal unit As MeasurementUnit	0	I	Pixels are used as the unit of measurement (based on the display of the signature device used)
	1	I	Millimetres are used as the unit of measurement
	2	I	Inches are used as the unit of measurement
Return value	**Values**	**Description**	
DOUBLE double Double	>= 0	Page height	
	< 0	Error (not STPadLibNet.dll)	

8.87.1 STPadCapt.ocx

Available from Version 8.1.4.

```
DOUBLE PDFGetHeight(LONG nPage, LONG nUnit)
```

8.87.1.1 Implementation in C#

```
double dHeight = axSTPadCapt1. PDFGetHeight(1, 1);
if (dHeight < 0.0)
    MessageBox.Show(String.Format("Error {0}", (int)dHeight));
else
    MessageBox.Show(String.Format("The page has a height of {0}
                                   millimetres.", dHeight));
```

8.87.1.2 Implementation in Visual Basic

```
Dim dHeight As Double = AxSTPadCapt1.PDFGetHeight(1, 1)
If dHeight < 0R Then
    MsgBox("Error " & CStr(dHeight))
Else
    MsgBox("The page has a height of " & CStr(dHeight) & " millimetres.")
End If
```

8.87.2 STPadLib.dll

Available from Version 8.1.4.

```
LONG STPDFGetHeight(LONG nPage, MEASUREMENTUNIT nUnit)
```

The `MEASUREMENTUNIT` enumeration is defined as follows:

```
kPixels = 0,
kMillimetres = 1,
kInches = 2
```

8.87.2.1 Implementation in C++

```
double dHeight = STPDFGetHeight(1, kMillimetres);
WCHAR szText[64];
if (dHeight < 0.)
    swprintf_s(szText, 64, L"Error %d", (int)dHeight);
else
    swprintf_s(szText, 64, L" The page has a height of %d millimetres.",
               dHeight);
```

8.87.3 STPadLibNet.dll

Available from Version 8.1.4.

```
double PDFGetHeight(int page, MeasurementUnit unit)
```

```
Function PDFGetHeight(ByVal page as Integer, ByVal unit As MeasurementUnit) As
Double
```

The `MeasurementUnit` enumeration is defined as follows:

```
Pixels = 0,
Millimetres = 1,
Inches = 2
```

8.87.3.1 Implementation in C#

```
try
{
    double dHeight = stPad.PDFGetHeight(1, MeasurementUnit.Millimetres);
    MessageBox.Show(String.Format("The page has a height of {0}
                                   millimetres.", dHeight));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.87.3.2 Implementation in Visual Basic

```
Try
    Dim dHeight As Double
    dHeight = STPad.PDFGetHeight(1, MeasurementUnit.Millimetres)
    MsgBox("The page has a height of " & CStr(dHeight) & " millimetres.")
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.88 PDFSelectRect method

This method can be used to specify an excerpt of a page of the currently loaded PDF document, which can then be displayed on the signature device with the `DisplaySetPDF()` method. The full page is always displayed by default.

Parameter	Values	I/O	Description

`LONG nPage`
`int page`
`ByVal page As Integer`
	all	I	Number of the page to be output (starting at 1)

`DOUBLE dLeft`
`double left`
`ByVal left As Double`
	>= 0	I	Left border of the excerpt on the PDF page

`DOUBLE dTop`
`double top`
`ByVal top As Double`
	>= 0	I	Upper border of the excerpt on the PDF page

`DOUBLE dWidth`
`double width`
`ByVal width As Double`
	>= 0	I	Width of the excerpt (the excerpt may not be outside of the page, the page width can be determined using `PDFGetWidth()`)

`DOUBLE dHeight`
`double height`
`ByVal height As Double`
	>= 0	I	Height of the excerpt (the excerpt may not be outside of the page, the page height can be determined using `PDFGetHeight()`)

LONG nUnit MEASUREMENTUNIT nUnit MeasurementUnit unit ByVal unit As MeasurementUnit	0	I	Pixels are used as the unit of measurement (based on the display of the signature device used)
	1	I	Millimetres are used as the unit of measurement
	2	I	Inches are used as the unit of measurement
Return value	**Values**	**Description**	
LONG	0	Method was executed successfully	
	< 0	Error	

8.88.1 STPadCapt.ocx

Available from Version 8.1.4.

```
LONG PDFSelectRect(LONG nPage, DOUBLE dLeft, DOUBLE dTop, DOUBLE dWidth, DOUBLE dHeight, LONG nUnit)
```

8.88.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.PDFSelectRect(1, 0.0, 0.0, 640.0, 480.0, 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.88.1.2 Implementation in Visual Basic

```
Dim nResult As Integer
nResult = AxSTPadCapt1.PDFSelectRect(1, 0R, 0R,640R, 480R, 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nResult))
End If
```

8.88.2 STPadLib.dll

Available from Version 8.1.4.

```
LONG STPDFSelectRect(LONG nPage, DOUBLE dLeft, DOUBLE dTop, DOUBLE dWidth, DOUBLE dHeight, MEASUREMENTUNIT nUnit)
```

The MEASUREMENTUNIT enumeration is defined as follows:

```
kPixels = 0,
kMillimetres = 1,
kInches = 2
```

8.88.2.1 Implementation in C++

```
int nResult = STPDFSelectRect(1, 0., 0., 640., 480., 0);
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

8.88.3 STPadLibNet.dll

Available from Version 8.1.4.

```
void PDFSelectRect(int page, double left, double top, double width, double
height, MeasurementUnit unit)
```

```
Sub PDFSelectRect(ByVal page As Integer, ByVal left As Double, ByVal top As
Double, ByVal width As Double, ByVal height As Double, ByVal unit As
MeasurementUnit)
```

The `MeasurementUnit` enumeration is defined as follows:

```
Pixels = 0,
Millimetres = 1,
Inches = 2
```

8.88.3.1 Implementation in C#

```
try
{
    stPad.PDFSelectRect(1, 0.0, 0.0, 640.0, 480.0,
                        MeasurementUnit.Pixels);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.88.3.2 Implementation in Visual Basic

```
Try
    STPad.PDFSelectRect(1, 0R, 0R,640R, 480R, MeasurementUnit.Pixels)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.89 PDFAddImage / PDFAddImageFromFile method

This method can be used to add an image to a page of the currently loaded PDF document. The added image(s) is/are then rendered on the page image the next time the `DisplaySetPDF()` method is called. This can be used to highlight areas of the document on the signature device, for example.

Parameter	Values	I/O	Description

`LONG nPage`
`int page`
`ByVal page As Integer`
	`> 0`	I	Number of the page to be output (starting at 1)

`LONG nLeft`
`int left`
`ByVal left As Integer`
	all	I	Left position of the image on the PDF page in pixels (based on the display of the signature device used); is adapted to the scaling factor of the `DisplaySetPDF()` method

`LONG nTop`
`int top`
`ByVal top As Integer`
	all	I	Top position of the image on the PDF page in pixels (based on the display of the signature device used); is adapted to the scaling factor of the `DisplaySetPDF()` method

LONG nImageHandle HBITMAP hBitmap Bitmap bitmap ByVal bitmap As Bitmap	!= NULL	I	HBITMAP or System.Drawing.Bitmap, to be output; if there is an existing Alpha channel, it is taken into account
BSTR bstrPath LPCWSTR szPath string path ByVal path As String	!= NULL	I	Full file path or URL of the image; supported image formats are BMP, GIF, JPEG, PNG and TIFF; any existing Alpha channel is taken into account
Return value	**Values**	**Description**	
LONG int Integer	>= 0	ID of the added image (unique to the specified page)	
	< 0	Error (not STPadLibNet.dll)	

8.89.1 STPadCapt.ocx

Available from Version 8.4.2.0.

```
LONG PDFAddImage(LONG nPage, LONG nLeft, LONG nTop, LONG nImageHandle)
LONG PDFAddImageFromFile(LONG nPage, LONG nLeft, LONG nTop, BSTR bstrPath)
```

8.89.1.1 Implementation in C#

Work in the memory:

```
Bitmap bitmap = (Bitmap)Bitmap.FromFile(@"C:\Image.png");
IntPtr hBitmap = bitmap.GetHbitmap(Color.Black);
int nId = axSTPadCapt1.PDFAddImage(1, 0, 0, hBitmap);
DeleteObject(hBitmap);
if (nId < 0)
    MessageBox.Show(String.Format("Error {0}", nId);
```

Work with files:

```
int nId = axSTPadCapt1.PDFAddImageFromFile(1, 0, 0, @"C:\Image.png");
if (nId < 0)
    MessageBox.Show(String.Format("Error {0}", nId);
```

8.89.1.2 Implementation in Visual Basic

Work in the memory:

```
Dim bitmap As Bitmap = Bitmap.FromFile("C:\Image.png")
Dim hBitmap As IntPtr = bitmap.GetHbitmap(Color.Black)
Dim nId As Integer = AxSTPadCapt1.PDFAddImage(1, 0, 0, hBitmap)
DeleteObject(hBitmap)
If nId < 0 Then
    MsgBox("Error " & CStr(nId))
End If
```

Work with files:

```
Dim nId As Integer
nId = AxSTPadCapt1.PDFAddImageFromFile(1, 0, 0, "C:\Image.png")
If nId < 0 Then
    MsgBox("Error " & CStr(nId))
End If
```

8.89.2 STPadLib.dll

Available from Version 8.4.2.0.

```
LONG STPDFAddImage(LONG nPage, LONG nLeft, LONG nTop, HBITMAP hBitmap)
```

```
LONG STPDFAddImageFromFile(LONG nPage, LONG nLeft, LONG nTop, LPCWSTR szPath)
```

8.89.2.1 Implementation in C++

Work in the memory:

```
HBITMAP hBm = (HBITMAP)LoadImage(0, L"C:\\Image.bmp", IMAGE_BITMAP, 0, 0,
                                 LR_LOADFROMFILE | LR_CREATEDIBSECTION);
LONG nId = STPDFAddImage(1, 0, 0, hBm);
DeleteObject(hBm);
if (nId < 0)
    AfxMessageBox(L"Error!");
```

Work with files:

```
LONG nId = STPDFAddImageFromFile(1, 0, 0, L"C:\\Image.png");
if (nId < 0)
    AfxMessageBox(L"Error!");
```

8.89.3 STPadLibNet.dll

Available from Version 8.4.2.0.

```
int PDFAddImage(int page, int left, int top, System.Drawing.Bitmap bitmap)
```

```
int PDFAddImageFromFile(int page, int left, int top, string path)
```

```
Function PDFAddImage(ByVal page As Integer, ByVal left As Integer, ByVal top As
Integer, ByVal bitmap As System.Drawing.Bitmap) As Integer
```

```
Function PDFAddImageFromFile(ByVal page As Integer, ByVal left As Integer,
ByVal top As Integer, ByVal path As string) As Integer
```

8.89.3.1 Implementation in C#

Work in the memory:

```
int nId;
try
{
    Bitmap bitmap = (Bitmap)Bitmap.FromFile(@"C:\Image.png");
    nId = stPad.PDFAddImage(1, 0, 0, bitmap);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

Work with files:

```
int nId;
try
{
    nId = stPad.PDFAddImageFromFile(1, 0, 0, @"C:\Image.png");
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.89.3.2 Implementation in Visual Basic

Work in the memory:

```
Dim nId As Integer
Try
    Dim bitmap As Bitmap = Bitmap.FromFile("C:\Image.png")
    nId = STPad.PDFAddImage(1, 0, 0, bitmap)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

Work with files:

```
Dim nId As Integer
Try
    nId = STPad.PDFAddImageFromFile(1, 0, 0, "C:\Image.png")
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

8.90 PDFRemoveImage method

This method can be used to remove an image that has been added to a page using `PDFAddImage()` or `PDFAddImageFromFile()`. The removed image is no longer rendered on the page image the next time the `DisplaySetPDF()` method is called.

Parameter	Values	I/O	Description

`LONG nPage`
`int page`
`ByVal page As Integer`
	`> 0`	I	Number of the page to be output (starting at 1)

`LONG nId`
`int id`
`ByVal id As Integer`
	`>= 0`	I	ID of the image added that must be returned when calling `PDFAddImage()` or `PDFAddImageFromFile()`.

Return value	**Values**	**Description**	
`LONG`	`0`	Method was executed successfully	
	`< 0`	Error	

8.90.1 STPadCapt.ocx

Available from Version 8.4.2.0.

```
LONG PDFRemoveImage(LONG nPage, LONG nId)
```

8.90.1.1 Implementation in C#

```
int nResult = axSTPadCapt1.PDFRemoveImage(1, 0);
if (nResult < 0)
    MessageBox.Show(String.Format("Error {0}", nResult);
```

8.90.1.2 Implementation in Visual Basic

```
Dim nResult As Integer = AxSTPadCapt1.PDFRemoveImage(1, 0)
If nResult < 0 Then
    MsgBox("Error " & CStr(nId))
End If
```

8.90.2 STPadLib.dll

Available from Version 8.4.2.0.

```
LONG STPDFRemoveImage(LONG nPage, LONG nId)
```

8.90.2.1 Implementation in C++

```
LONG nResult = STPDFRemoveImage(1, 0);
if (nResult < 0)
    AfxMessageBox(L"Error!");
```

8.90.3 STPadLibNet.dll

Available from Version 8.4.2.0.

```
void PDFRemoveImage(int page, int id)
```

```
Sub PDFRemoveImage(ByVal page As Integer, ByVal id As Integer)
```

8.90.3.1 Implementation in C#

```
try
{
    stPad.PDFRemoveImage(1, 0);
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

8.90.3.2 Implementation in Visual Basic

```
Try
    STPad.PDFAddImage(1, 0)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

9 Properties

Properties are named according to the following naming convention:

- General hardware properties begin with '**Device**'
- Properties that apply to signatures begin with '**Signature**'
- LCD properties begin with '**Display**'
- Component properties begin with '**Control**'

The STPadLib.dll component has no properties. Get() and Set() methods are used instead. These methods all begin with 'ST,' for example„STDeviceGetLedDefaultFlag()".

Some programming languages, such as C++, for example, do not support COM properties and use wrapper methods instead. In Visual C++, properties are wrapped as 'GetProperty()' and 'SetProperty(),' for example.

9.1 DeviceLedDefaultFlag property

This property specifies whether the LED on the front of the pad automatically changes to green when the device is in signature capture mode. The default setting is TRUE. The LED always lights up yellow as soon as the device has been detected by the PC operating system and is ready for use.

Value	I/O	Description
TRUE	I/O	LED lights up green for a capture process
FALSE	I/O	LED does not change

9.1.1 STPadCapt.ocx

Available from Version 8.0.1.

```
VARIANT_BOOL DeviceLedDefaultFlag
```

9.1.1.1 Implementation in C#

```
axSTPadCapt1.DeviceLedDefaultFlag = false;
```

9.1.1.2 Implementation in Visual Basic

```
AxSTPadCapt1.DeviceLedDefaultFlag = False
```

9.1.2 STPadLib.dll

Available from Version 8.0.19.

```
BOOL STDeviceGetLedDefaultFlag()
VOID STDeviceSetLedDefaultFlag(BOOL bFlag)
```

9.1.2.1 Implementation in C++

```
STDeviceSetLedDefaultFlag(FALSE);
```

9.1.3 STPadLibNet.dll

Available from Version 8.0.19.

```
bool DeviceLedDefaultFlag { get; set; }
```

```
Property DeviceLedDefaultFlag As Boolean
```

9.1.3.1 Implementation in C#

```
stPad.DeviceLedDefaultFlag = false;
```

9.1.3.2 Implementation in Visual Basic

```
STPad.DeviceLedDefaultFlag = False
```

9.2 ControlVersion property

This property holds the version number of the component.

Value	I/O	Description
max. 16 characters	O	Version number of the component

9.2.1 STPadCapt.ocx

Available from Version 8.0.19.

```
BSTR ControlVersion
```

9.2.1.1 Implementation in C#

```
MessageBox.Show(String.Format("Version: {0}",
                              axSTPadCapt1.ControlVersion));
```

9.2.1.2 Implementation in Visual Basic

```
MsgBox("Version: " & AxSTPadCapt1.ControlVersion)
```

9.2.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STControlGetVersion(WCHAR szVersion[16])
```

9.2.2.1 Implementation in C++

```
WCHAR szVersion[16];
LONG nRc = STControlGetVersion(szVersion);
if (nRc < 0)
    wprintf(L"Error %d", nRc);
else
    wprintf(L"Version: %s", szVersion);
```

9.2.3 STPadLibNet.dll

Available from Version 8.0.19.

```
string ControlVersion { get; }
```

```
ReadOnly Property ControlVersion() As String
```

9.2.3.1 Implementation in C#

```
try
{
    MessageBox.Show(String.Format("Version: {0}", stPad.ControlVersion));
}
catch (STPadException ex)
{
    MessageBox.Show(ex.Message);
}
```

9.2.3.2 Implementation in Visual Basic

```
Try
    MsgBox("Version: " & STPad.ControlVersion)
Catch ex As STPadException
    MsgBox(ex.Message)
End Try
```

9.3 ControlAppName property

This property can contain the name of the application that uses the component. Users can use this name to exclusively assign one or more image memories. Please refer to section 'Exclusive use of non-volatile memory' for details.

Value	I/O	Description
NULL	I/O	Application does not use any memories exclusively
!= NULL	I/O	Name of the application (may contain spaces)

9.3.1 STPadCapt.ocx

Available from Version 8.0.17.

```
BSTR ControlAppName
```

9.3.1.1 Implementation in C#

```
axSTPadCapt1.ControlAppName = "My Great App";
```

9.3.1.2 Implementation in Visual Basic

```
AxSTPadCapt1.ControlAppName = "My Great App"
```

9.3.2 STPadLib.dll

Available from Version 8.0.19.

```
VOID STControlSetAppName(LPCWSTR szName)
```

9.3.2.1 Implementation in C++

```
STControlSetAppName(L"My Great App");
```

9.3.3 STPadLibNet.dll

Available from Version 8.0.19.

```
string ControlAppName { get; set; }
```

```
Property ControlAppName As String
```

9.3.3.1 Implementation in C#

```
stPad.ControlAppName = "My Great App";
```

9.3.3.2 Implementation in Visual Basic

```
STPad.ControlAppName = "My Great App"
```

9.4 ControlBackColor property

This property specifies the colour in which the control element's window is displayed. The default setting is white.

Value	I/O	Description
>= 0	I/O	Window colour

9.4.1 STPadCapt.ocx

Available from Version 8.0.1.

```
OLE_COLOR ControlBackColor
```

9.4.1.1 Implementation in C#

```
axSTPadCapt1.ControlBackColor = Color.FromArgb(238, 121, 0);
```

9.4.1.2 Implementation in Visual Basic

```
AxSTPadCapt1.ControlBackColor = Color.FromArgb(238, 121, 0)
```

9.4.2 STPadLib.dll

Not available.

9.4.3 STPadLibNet.dll

Available from Version 8.0.21 (only in the STPadLibControl class).

```
Color ControlBackColor { get; set; }
```

```
Property ControlBackColor As Color
```

9.4.3.1 Implementation in C#

```
stPad.ControlBackColor = Color.FromArgb(238, 121, 0);
```

9.4.3.2 Implementation in Visual Basic

```
STPad.ControlBackColor = Color.FromArgb(238, 121, 0)
```

9.5 ControlRectColor property

This property specifies the colour of the one-pixel-wide border within the control element's window. No border is drawn if this value is identical to the value of the `ControlBackColor` property. The default setting is orange.

Value	I/O	Description
>= 0	I/O	Border colour

9.5.1 STPadCapt.ocx

Available from Version 8.0.3.

```
OLE_COLOR ControlRectColor
```

9.5.1.1 Implementation in C#

```
axSTPadCapt1.ControlRectColor = Color.FromArgb(238, 121, 0);
```

9.5.1.2 Implementation in Visual Basic

```
AxSTPadCapt1.ControlRectColor = Color.FromArgb(238, 121, 0)
```

9.5.2 STPadLib.dll

Not available.

9.5.3 STPadLibNet.dll

Available from Version 8.0.21 (only in the STPadLibControl class).

```
Color ControlRectColor { get; set; }
```

```
Property ControlRectColor As Color
```

9.5.3.1 Implementation in C#

```
stPad.ControlRectColor = Color.FromArgb(238, 121, 0);
```

9.5.3.2 Implementation in Visual Basic

```
STPad.ControlRectColor = Color.FromArgb(238, 121, 0)
```

9.6 ControlPenColor property

This property specifies the colour in which the captured signature is rendered in the control element's window. The default setting is blue.

Value	I/O	Description
>= 0	I/O	Pen colour

9.6.1 STPadCapt.ocx

Available from Version 8.0.1.

```
OLE_COLOR ControlPenColor
```

9.6.1.1 Implementation in C#

```
axSTPadCapt1.ControlPenColor = Color.FromArgb(238, 121, 0);
```

9.6.1.2 Implementation in Visual Basic

```
AxSTPadCapt1.ControlPenColor = Color.FromArgb(238, 121, 0)
```

9.6.2 STPadLib.dll

Not available.

9.6.3 STPadLibNet.dll

Available from Version 8.0.21 (only in the STPadLibControl class).

```
Color ControlPenColor { get; set; }
```

```
Property ControlPenColor As Color
```

9.6.3.1 Implementation in C#

```
stPad.ControlPenColor = Color.FromArgb(238, 121, 0);
```

9.6.3.2 Implementation in Visual Basic

```
STPad.ControlPenColor = Color.FromArgb(238, 121, 0)
```

9.7 ControlPenWidth property

This property specifies the pen width with which the captured signature is rendered in the control element's window. The default setting is 0 (variable).

Value	I/O	Description
0	I/O	A variable pen width is used that is dependent on the window size and the pressure values
> 0	I/O	Fixed pen width in pixels
< 0	I/O	Fixed pen width in pixels (absolute value); the pressure values are visualized by drawing in variable brightness

9.7.1 STPadCapt.ocx

Available from Version 8.0.3.

```
SHORT ControlPenWidth
```

9.7.1.1 Implementation in C#

```
axSTPadCapt1.ControlPenWidth = 0;
```

9.7.1.2 Implementation in Visual Basic

```
AxSTPadCapt1.ControlPenWidth = 0
```

9.7.2 STPadLib.dll

Not available.

9.7.3 STPadLibNet.dll

Available from Version 8.0.21 (only in the STPadLibControl class).

```
int ControlPenWidth { get; set; }
```

```
Property ControlPenWidth As Integer
```

9.7.3.1 Implementation in C#

```
stPad.ControlPenWidth = 0;
```

9.7.3.2 Implementation in Visual Basic

```
STPad.ControlPenWidth = 0
```

9.8 ControlMirrorDisplay property

This property specifies whether the content of the LCD should also be displayed in the control element's window. The drawing is always centred in the control. The default setting is 1.

Value	I/O	Description
0	I/O	Nothing is displayed in the control element
1	I/O	The signature is displayed in real time in the control element
2	I/O	The signature as well as all bitmaps and text are displayed in real time in the control element; for good results the control must have the same size as the LDC (in pixels)
3	I/O	The signature is displayed in real time in the control element; it's scaled that the per `SensorSetSignRect()` defined rectangle fills out the window of the control completely
4	I/O	Corresponds to the value 2, in this mode, the hotspots configured on the pad can also be operated with the mouse in the control element; this does not apply to hotspots that have been added with the `SensorAddKeypadHotspot()` method

9.8.1 STPadCapt.ocx

Available from Version 8.0.3. The status described is available from Version 8.0.29.

```
SHORT ControlMirrorDisplay
```

9.8.1.1 Implementation in C#

```
axSTPadCapt1.ControlMirrorDisplay = 2;
```

9.8.1.2 Implementation in Visual Basic

```
AxSTPadCapt1.ControlMirrorDisplay = 2
```

9.8.2 STPadLib.dll

Not available.

9.8.3 STPadLibNet.dll

Available from Version 8.0.21 (only in the STPadLibControl class). The status described is available from Version 8.0.29.

```
signotec.STPadLibNet.MirrorMode ControlPenWidth { get; set; }
```

```
Property ControlPenWidth As signotec.STPadLibNet.MirrorMode
```

The `MirrorMode` enumeration is defined as follows:

```
Nothing = 0,
Signature = 1,
Everything = 2,
SignRect = 3,
EverythingActiveHotSpots = 4
```

9.8.3.1 Implementation in C#

```
stPad.ControlMirrorDisplay = MirrorMode.Everything;
```

9.8.3.2 Implementation in Visual Basic

```
STPad.ControlMirrorDisplay = MirrorMode.Everything;
```

9.9 DisplayWidth property

This property holds the width of the LCD It can only be queried after a device has been opened.

Value	I/O	Description
>= 0	O	Width of the display in pixels
< 0	O	Error

9.9.1 STPadCapt.ocx

Available from Version 8.0.1.

```
LONG DisplayWidth
```

9.9.1.1 Implementation in C#

```
MessageBox.Show(String.Format("Display width is {0}",
                              axSTPadCapt1.DisplayWidth));
```

9.9.1.2 Implementation in Visual Basic

```
MsgBox("Display width is " & CStr(AxSTPadCapt1.DisplayWidth)
```

9.9.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDisplayGetWidth()
```

9.9.2.1 Implementation in C++

```
wprintf(L"Display width is %d", STDisplayGetWidth());
```

9.9.3 STPadLibNet.dll

Available from Version 8.0.19.

```
int DisplayWidth { get; }
```

```
ReadOnly Property DisplayWidth As Integer
```

9.9.3.1 Implementation in C#

```
MessageBox.Show(String.Format("Display width is {0}",
                              stPad.DisplayWidth));
```

9.9.3.2 Implementation in Visual Basic

```
MsgBox("Display width is " & CStr(STPad.DisplayWidth)
```

9.10 DisplayHeight property

This property holds the height of the LCD It can only be queried after a device has been opened.

Value	I/O	Description
>= 0	O	Height of the display in pixels
< 0	O	Error

9.10.1 STPadCapt.ocx

Available from Version 8.0.1.

```
LONG DisplayHeight
```

9.10.1.1 Implementation in C#

```
MessageBox.Show(String.Format("Display height is {0}",
                              axSTPadCapt1.DisplayHeight));
```

9.10.1.2 Implementation in Visual Basic

```
MsgBox("Display height is " & CStr(AxSTPadCapt1.DisplayHeight)
```

9.10.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDisplayGetHeight()
```

9.10.2.1 Implementation in C++

```
wprintf(L"Display height is %d", STDisplayGetHeight());
```

9.10.3 STPadLibNet.dll

Available from Version 8.0.19.

```
int DisplayHeight { get; }
```

```
ReadOnly Property DisplayHeight As Integer
```

9.10.3.1 Implementation in C#

```
MessageBox.Show(String.Format("Display height is {0}",
                              stPad.DisplayHeight));
```

9.10.3.2 Implementation in Visual Basic

```
MsgBox("Display height is " & CStr(STPad.DisplayHeight)
```

9.11 DisplayResolution property

This property contains the resolution of the LCD. It can only be queried after a device has been opened.

Value	I/O	Description
>= 0	O	Resolution of the display in pixels per inch (ppi).
< 0	O	Error

9.11.1 STPadCapt.ocx

Available from Version 8.4.2.0.

```
DOUBLE DisplayResolution
```

9.11.1.1 Implementation in C#

```
MessageBox.Show(String.Format("Display resolution is {0} ppi",
                              axSTPadCapt1.DisplayResolution));
```

9.11.1.2 Implementation in Visual Basic

```
MsgBox("Display resolution is " & CStr(AxSTPadCapt1.DisplayResolution)
```

9.11.2 STPadLib.dll

Available from Version 8.4.2.0.

```
DOUBLE STDisplayGetResolution()
```

9.11.2.1 Implementation in C++

```
wprintf(L"Display resolution is %d", STDisplayGetResolution());
```

9.11.3 STPadLibNet.dll

Available from Version 8.4.2.0.

```
double DisplayResolution { get; }
```

```
ReadOnly Property DisplayResolution As Double
```

9.11.3.1 Implementation in C#

```
MessageBox.Show(String.Format("Display resolution is {0}",
                              stPad.DisplayResolution));
```

9.11.3.2 Implementation in Visual Basic

```
MsgBox("Display resolution is " & CStr(STPad.DisplayResolution)
```

9.12 DisplayTargetWidth property

This property holds the width of the memory defined with the `DisplaySetTarget()` method. It can only be queried after a device has been opened.

Value	I/O	Description
>= 0	O	Width of the memory in pixels
< 0	O	Error

9.12.1 STPadCapt.ocx

Available from Version 8.0.17.

```
LONG DisplayTargetWidth
```

9.12.1.1 Implementation in C#

```
MessageBox.Show(String.Format("Target width is {0}",
                              axSTPadCapt1.DisplayTargetWidth));
```

9.12.1.2 Implementation in Visual Basic

```
MsgBox("Target width is " & CStr(AxSTPadCapt1.DisplayTargetWidth)
```

9.12.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDisplayGetTargetWidth()
```

9.12.2.1 Implementation in C++

```
wprintf(L"Target width is %d", STDisplayGetTargetWidth());
```

9.12.3 STPadLibNet.dll

Available from Version 8.0.19.

```
int DisplayTargetWidth { get; }
```

```
ReadOnly Property DisplayTargetWidth As Integer
```

9.12.3.1 Implementation in C#

```
MessageBox.Show(String.Format("Target width is {0}",
                              stPad.DisplayTargetWidth));
```

9.12.3.2 Implementation in Visual Basic

```
MsgBox("Target width is " & CStr(STPad.DisplayTargetWidth)
```

9.13 DisplayTargetHeight property

This property holds the height of the memory defined with the `DisplaySetTarget()` method. It can only be queried after a device has been opened.

Value	I/O	Description
>= 0	O	Height of the memory in pixels
< 0	O	Error

9.13.1 STPadCapt.ocx

Available from Version 8.0.17.

```
LONG DisplayScrollSpeed
```

9.13.1.1 Implementation in C#

```
MessageBox.Show(String.Format("Target height is {0}",
                              axSTPadCapt1.DisplayTargetHeight));
```

9.13.1.2 Implementation in Visual Basic

```
MsgBox("Target height is " & CStr(AxSTPadCapt1.DisplayTargetHeight)
```

9.13.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDisplayGetTargetHeight()
```

9.13.2.1 Implementation in C++

```
wprintf(L"Target height is %d", STDisplayGetTargetHeight());
```

9.13.3 STPadLibNet.dll

Available from Version 8.0.19.

```
int DisplayTargetHeight { get; }
```

```
ReadOnly Property DisplayTargetHeight As Integer
```

9.13.3.1 Implementation in C#

```
MessageBox.Show(String.Format("Target height is {0}",
                              stPad.DisplayTargetHeight));
```

9.13.3.2 Implementation in Visual Basic

```
MsgBox("Target height is " & CStr(STPad.DisplayTargetHeight)
```

9.14 DisplayScrollSpeed property

This property determines the speed at which the screen content is scrolled when a hotspot generated by `STSensorAddScrollHotSpot()` is triggered. The default setting is 100.

Value	I/O	Description
`1 - 1000`	I/O	Scroll speed in lines / second; not all values are possible; invalid values are rounded to the next valid value

9.14.1 STPadCapt.ocx

Available from Version 8.0.17.

```
LONG DisplayScrollSpeed
```

9.14.1.1 Implementation in C#

```
axSTPadCapt1.DisplayScrollSpeed = 100;
```

9.14.1.2 Implementation in Visual Basic

```
AxSTPadCapt1.DisplayScrollSpeed = 100
```

9.14.2 STPadLib.dll

Available from Version 8.0.19.

```
LONG STDisplayGetScrollSpeed()
LONG STDisplaySetScrollSpeed(LONG nSpeed)
```

The `STDisplaySetScrollSpeed()` method returns the value that has actually been set.

9.14.2.1 Implementation in C++

```
STDisplaySetScrollSpeed(100);
```

9.14.3 STPadLibNet.dll

Available from Version 8.0.19.

```
int DisplayScrollSpeed { get; set; }
Property DisplayScrollSpeed As Integer
```

9.14.3.1 Implementation in C#

```
stPad.DisplayScrollSpeed = 100;
```

9.14.3.2 Implementation in Visual Basic

```
STPad.DisplayScrollSpeed = 100
```

9.15 DisplayRotation property

This property defines the rotation with which the image data is transferred to the signature device and the sensor areas are defined. A change of the value does not take effect immediately, but as follows:

As soon as the content of an image memory is changed for the first time after a change of the property, the complete image memory is deleted before the new content is transferred to the memory. The set rotation is observed for the new content.

The overlay rectangle always respects the rotation of the foreground memory, so it is automatically rotated as soon as the content of the foreground memory is rotated. However, this does not apply to the content of the overlay memory, which must be refilled after changing the property.

Hotspots, the scroll area as well as the signature window should be deleted after changing the property and reset as soon as the content of the foreground memory has been changed.

The default setting is 0.

Value	I/O	Description
`0, 180`	I/O	Rotation in degrees (clockwise)

9.15.1 STPadCapt.ocx

Available from Version 8.0.29.

```
LONG DisplayRotation
```

9.15.1.1 Implementation in C#

```
axSTPadCapt1.DisplayRotation = 180;
```

9.15.1.2 Implementation in Visual Basic

```
AxSTPadCapt1.DisplayRotation = 180
```

9.15.2 STPadLib.dll

Available from Version 8.0.29.

```
LONG STDisplayGetRotation()
LONG STDisplaySetRotation(LONG nRotation)
```

9.15.2.1 Implementation in C++

```
STDisplaySetRotation(180);
```

9.15.3 STPadLibNet.dll

Available from Version 8.0.29.

```
int DisplayRotation { get; set; }
Property DisplayRotation As Integer
```

9.15.3.1 Implementation in C#

```
stPad.DisplayRotation = 180;
```

9.15.3.2 Implementation in Visual Basic

```
STPad.DisplayRotation = 180
```

9.16 SignatureState property

This property holds the current state of the signature capture process.

Value	I/O	Description
TRUE	O	Signature capture process is running
FALSE	O	Signature capture process is not running

9.16.1 STPadCapt.ocx

Available from Version 8.0.1.

```
VARIANT_BOOL SignatureState
```

9.16.1.1 Implementation in C#

```
if (!axSTPadCapt1.SignatureState)
    axSTPadCapt1.SignatureStart();
else
    axSTPadCapt1.SignatureConfirm();
```

9.16.1.2 Implementation in Visual Basic

```
If AxSTPadCapt1.SignatureState = False Then
    AxSTPadCapt1.SignatureStart()
Else
    AxSTPadCapt1.SignatureConfirm()
End If
```

9.16.2 STPadLib.dll

Available from Version 8.0.19.

```
BOOL STSignatureGetState()
```

9.16.2.1 Implementation in C++

```
if (!STSignatureGetState())
    STSignatureStart();
else
    STSignatureConfirm();
```

9.16.3 STPadLibNet.dll

Available from Version 8.0.19.

```
bool SignatureState { get; }
```

```
ReadOnly Property SignatureState As Boolean
```

9.16.3.1 Implementation in C#

```
if (!stPad.SignatureState)
    stPad.SignatureStart();
else
    stPad.SignatureConfirm();
```

9.16.3.2 Implementation in Visual Basic

```
If STPad.SignatureState = False Then
    STPad.SignatureStart()
Else
    STPad.SignatureConfirm()
End If
```

9.17 RSASignPasswordLength property

This property determines the minimum password length that the password to be set with `RSASetSignPassword()` must have before `RSASign()` can be used for signing. If the pad supports the function, every time `RSASetSigningCert()` or `RSAGenerateSigningCert()` is called, the current value is set as the minimum password length. If the function is not supported, a value set here that is unequal to 0 will result in an error.

0 (no minimum length) is set by default.

This property is not automatically reset when the pad is closed.

The minimum password length specification works with the Sigma model from firmware 2.13, the Zeta model from firmware 1.0, the Omega model from firmware 2.17, the Gamma model from firmware 1.32 and the Delta model from firmware 1.31.

Value	I/O	Description
0	I/O	No minimum password length is set
> 0	I/O	A minimum password length is set. Odd numbers from 1 to 31 are allowed.

9.17.1 STPadCapt.ocx

Available from Version 8.4.3.

```
LONG RSAPasswordLength
```

9.17.1.1 Implementation in C#

```
axSTPadCapt1.RSAPasswordLength = 31;
```

9.17.1.2 Implementation in Visual Basic

```
AxSTPadCapt1.RSAPasswordLength = 31
```

9.17.2 STPadLib.dll

Available from Version 8.4.3.

```
LONG STRSAGetSignPasswordLength()
```

```
LONG STRSASetSignPasswordLength(LONG nPasswordLength)
```

9.17.2.1 Implementation in C++

```
STRSASetSignPasswordLength(31);
```

9.17.3 STPadLibNet.dll

Available from Version 8.4.3.

```
int RSAPasswordLength { get; set; }
```

```
Property RSAPasswordLength As Integer
```

9.17.3.1 Implementation in C#

```
stPad.RSAPasswordLength = 31;
```

9.17.3.2 Implementation in Visual Basic

```
STPad.RSAPasswordLength = 31
```

9.18 SignatureSignData property

This method is obsolete and is only included for compatibility reasons. Please use the `SignatureGetSignData()` method instead.

10 Events

Events are named according to the following naming convention:

- General hardware events begin with '**Device**'
- Events that apply to the signature begin with '**Signature**'
- Sensor events begin with '**Sensor**'
- Display events begin with '**Display**'

The STPadLib.dll component uses a callback mechanism to pass events through to the application. For more information, see the `STControlSetCallback()` method.

The events are implemented as delegates in the STPadLibNet.dll component. They run in the thread of the DLL, which is why UI elements of the application cannot be addressed directly. A further delegate must be addressed using `Invoke()` instead. Please also see the demo application included.

10.1 DeviceDisconnected event

This event is called as soon as a device is disconnected through an external event (e. g. unplugging the device).

Parameter	Values	Description

`LONG nIndex`
`int index`
`index As Integer`
	`>= 0`	Index of the disconnected device

Return value	**Values**	**Description**
-	-	-

10.1.1 STPadCapt.ocx

Available from Version 8.0.3.

```
void DeviceDisconnected(LONG nIndex)
```

10.1.1.1 Implementation in C#

```
private void axSTPadCapt1_DeviceDisconnected(object sender,
                AxSTPadCaptLib._DSTPadCaptEvents_DeviceDisconnectEvent e)
{
    MessageBox.Show(String.Format("Device {0} disconnected!", e.nIndex);
}
```

10.1.1.2 Implementation in Visual Basic

```
Private Sub AxSTPadCapt1_DeviceDisconnected _
             (ByVal sender As System.Object, ByVal e As _
             AxSTPadCaptLib._DSTPadCaptEvents_DeviceDisconnectedEvent) _
             Handles AxSTPadCapt1.DeviceDisconnected
    MsgBox("Device " & CStr(e.nIndex) & " disconnected!")
End Sub
```

10.1.2 STPadLib.dll

Available from Version 8.0.19.

```
VOID DeviceDisconnected(LONG nIndex)
```

10.1.2.1 Implementation in C++

```
VOID CMyClass::DeviceDisconnected(LONG nIndex)
{
    wprintf(L"Device %d disconnected!", nIndex);
}
```

10.1.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void raise_DeviceDisconnected(object sender,
signotec.STPadLibNet.DeviceDisconnectedEventArgs e)
```

```
Event DeviceDisconnected(ByVal sender As Object, ByVal e As
signotec.STPadLibNet.DeviceDisconnectedEventArgs)
```

The `DeviceDisconnectedEventArgs` class contains the following member described above:

```
public int index
```

```
Public index As Integer
```

10.1.3.1 Implementation in C#

```
private void STPad_DeviceDisconnected(object sender,
                                      DeviceDisconnectEventArgs e)
{
    MessageBox.Show(String.Format("Device {0} disconnected!", e.index);
}
```

10.1.3.2 Implementation in Visual Basic

```
Private Sub STPad_DeviceDisconnected(ByVal sender As Object, _
                        ByVal e As DeviceDisconnectedEventArgs) Handles _
                        STPad.DeviceDisconnected
    MsgBox("Device " & CStr(e.index) & " disconnected!")
End Sub
```

10.2 SignatureDataReceived event

This event is called when signature data is received from the pad.

Parameter	Values	Description

`LONG nXPos`
`int xPos`
`xPos As Integer`
	`>= 0`	x value of the received data record

`LONG nYPos`
`int yPos`
`yPos As Integer`
	`>= 0`	y value of the received data record

`LONG nPressure`
`int pressure`
`pressure As Integer`
	`0 - 1024`	Pressure value of the received data record

LONG nTimestamp int timestamp timestamp As Integer	>= 0	Timestamp of the received data record
Return value	**Values**	**Description**
-	-	-

10.2.1 STPadCapt.ocx

Available from Version 8.0.19.

```
void SignatureDataReceived(LONG nXPos, LONG nYPos, LONG nPressure, LONG
nTimestamp)
```

10.2.1.1 Implementation in C#

```
private void axSTPadCapt1_SignatureDataReceived(object sender,
                AxSTPadCaptLib._DSTPadCaptEvents_SignatureDataReceived e)
{
    MessageBox.Show(String.Format("X: {0}; Y: {1}; P: {2}; T: {3}",
                                  e.nXPos, e.nYPos, e.nPressure,
                                  e.nTimestamp);
}
```

10.2.1.2 Implementation in Visual Basic

```
Private Sub AxSTPadCapt1_SignatureDataReceived _
                (ByVal eventSender As System.Object, ByVal e As _
                AxSTPadCaptLib._DSTPadCaptEvents_SignatureDataReceived) _
                Handles AxSTPadCapt1.SignatureDataReceived
    MsgBox("X: " & CStr(e.nXPos) & "; Y: " & CStr(e.nYPos) & "; P: " & _
           CStr(e.nPressure) & "; T: " & CStr(e.nTimestamp))
End Sub
```

10.2.2 STPadLib.dll

Available from Version 8.0.19.

```
VOID SignatureDataReceived(LONG nXPos, LONG nYPos, LONG nPressure, LONG
nTimestamp)
```

10.2.2.1 Implementation in C++

```
void CMyClass::SignatureDataReceived(long nXPos, long nYPos, long
nPressure, long nTimestamp)
{
    WCHAR szText[64];
    swprintf_s(szText, 64, L"X: %d; Y: %d; P: %d; T: %d", nXPos, nYPos,
               nPressure, nTimestamp);
    AfxMessageBox(szText);
}
```

10.2.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void raise_SignatureDataReceived(object sender, signotec.STPadLibNet.
SignatureDataReceivedEventArgs e)

Event SignatureDataReceived(ByVal sender As Object, ByVal e As
signotec.STPadLibNet.SignatureDataReceivedEventArgs)
```

The `SignatureDataReceivedEventArgs` class contains the following members described above:

```
public int xPos
public int yPos
public int pressure
public int timestamp

Public xPos As Integer
Public yPos As Integer
Public pressure As Integer
Public timestamp As Integer
```

10.2.3.1 Implementation in C#

```
private void STPad_SignatureDataReceived(object sender,
                                         SignatureDataReceivedEventArgs e)
{
    MessageBox.Show(String.Format("X: {0}; Y: {1}; P: {2}; T: {3}",
                                  e.xPos, e.yPos, e.pressure,
                                  e.timestamp);
}
```

10.2.3.2 Implementation in Visual Basic

```
Private Sub STPad_SignatureDataReceived(ByVal sender As Object, _
               ByVal e As SignatureDataReceivedEventArgs) Handles  _
               STPad.SignatureDataReceived
    MsgBox("X: " & CStr(e.xPos) & "; Y: " & CStr(e.yPos) & "; P: " & _
           CStr(e.pressure) & "; T: " & CStr(e.timestamp))
End Sub
```

10.3 SensorHotSpotPressed event

This event is called as soon as the user lifts the pen off a rectangle defined with `SensorAddHotSpot()`, `SensorAddScrollHotSpot()` or `SensorAddKeyPadHotSpot()` after placing it in this rectangle.

Parameter	Values	Description

`LONG nHotSpotId`
`int hotSpotId`
`hotSpotId As Integer`
	`>= 0`	ID of the operated hotspot

	`-1`	An area set with `SensorAddKeypadHotSpot()` was clicked once and added to the list stored in the signature device
	`-2`	An area set with `SensorAddKeypadHotSpot()` has been clicked at least once but could not be added to the list stored in the signature device because it is full
Return value	**Values**	**Description**
-	-	-

10.3.1 STPadCapt.ocx

Available from Version 8.0.1. The status described is available from Version 8.4.3.

```
void SensorHotSpotPressed(LONG nHotSpotId)
```

10.3.1.1 Implementation in C#

```
private void axSTPadCapt1_SensorHotSpotPressed(object sender,
            AxSTPadCaptLib._DSTPadCaptEvents_SensorHotSpotPressedEvent e)
{
    if (e.nHotSpotId >= 0)
        MessageBox.Show(String.Format("Hotspot {0}", e.nHotSpotId);
    else
        // process keypad hotspot...
}
```

10.3.1.2 Implementation in Visual Basic

```
Private Sub AxSTPadCapt1_SensorHotSpotPressed _
            (ByVal eventSender As System.Object, ByVal e As _
            AxSTPadCaptLib._DSTPadCaptEvents_SensorHotSpotPressedEvent) _
            Handles AxSTPadCapt1.SensorHotSpotPressed
    If (e.nHotSpotId >= 0) Then
        MsgBox("Hotspot " & CStr(e.nHotSpotId)
    Else
        ' process keypad hotspot...
    End If
End Sub
```

10.3.2 STPadLib.dll

Available from Version 8.0.19. The status described is available from Version 8.4.3.

```
VOID SensorHotSpotPressed(LONG nHotSpotId)
```

10.3.2.1 Implementation in C++

```
VOID CMyClass::SensorHotSpotPressed(LONG nHotSpotId)
{
    if (e.nHotSpotId >= 0)
        wprintf(L"Hotspot %d!", nHotSpotId);
    else
        // process keypad hotspot...
}
```

10.3.3 STPadLibNet.dll

Available from Version 8.0.19. The status described is available from Version 8.4.3.

```
void raise_SensorHotSpotPressed(object sender,
signotec.STPadLibNet.SensorHotSpotPressedEventArgs e)
```

```
Event SensorHotSpotPressed(ByVal sender As Object, ByVal e As
signotec.STPadLibNet.SensorHotSpotPressedEventArgs)
```

The `SensorTimeoutOccuredEventArgs` class contains the following member described above:

```
public int hotSpotId
```

```
Public hotSpotId As Integer
```

10.3.3.1 Implementation in C#

```
private void STPad_SensorHotSpotPressed(object sender,
                                        SensorHotSpotPressedEventArgs e)
{
    if (e.nHotSpotId >= 0)
        MessageBox.Show(String.Format("Hotspot {0}", e.hotSpotId);
    else
        // process keypad hotspot...
}
```

10.3.3.2 Implementation in Visual Basic

```
Private Sub STPad_SensorHotSpotPressed(ByVal sender As Object, _
                ByVal e As SensorHotSpotPressedEventArgs) Handles _
                STPad.SensorHotSpotPressed
    If (e.nHotSpotId >= 0) Then
        MsgBox("Hotspot " & CStr(e.hotSpotId)
    Else
        ' process keypad hotspot...
    End If
End Sub
```

10.4 SensorTimeoutOccured Event

This event is called as soon as the timer started with `SensorStartTimer()` has expired.

Parameter	Values	Description

`LONG nPointsCount`
`int pointsCount`
`pointsCount As Integer`
	>= 0	Number of points captured if any

Return value	**Values**	**Description**
-	-	-

10.4.1 STPadCapt.ocx

Available from Version 8.0.11.

```
void SensorTimeoutOccured(LONG nPointsCount)
```

10.4.1.1 Implementation in C#

```
private void axSTPadCapt1_SensorTimeoutOccured(object sender,
             AxSTPadCaptLib._DSTPadCaptEvents_SensorTimeoutOccuredEvent
             e)
{
    MessageBox.Show(String.Format("Timeout, captured points: {0}",
                                  e.nPointsCount);
}
```

10.4.1.2 Implementation in Visual Basic

```
Private Sub AxSTPadCapt1_SensorTimeoutOccured _
           (ByVal eventSender As System.Object, ByVal e As _
           AxSTPadCaptLib._DSTPadCaptEvents_SensorTimeoutOccuredEvent) _
           Handles AxSTPadCapt1.SensorTimeoutOccured
    MsgBox("Timeout, captured points: " & CStr(e.nPointsCount)
End Sub
```

10.4.2 STPadLib.dll

Available from Version 8.0.19.

```
VOID SensorTimeoutOccured(LONG nPointsCount)
```

10.4.2.1 Implementation in C++

```
VOID CMyClass::SensorTimeoutOccured(LONG nPointsCount)
{
    wprintf(L"Timeout, captured points: %d!", nPointsCount);
}
```

10.4.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void raise_SensorTimeoutOccured(object sender,
signotec.STPadLibNet.SensorTimeoutOccuredEventArgs e)
```

```
Event SensorTimeoutOccured(ByVal sender As Object, ByVal e As
signotec.STPadLibNet.SensorTimeoutOccuredEventArgs)
```

The `SensorTimeoutOccuredEventArgs` class contains the following member described above:

```
public int pointsCount
```

```
Public pointsCount As Integer
```

10.4.3.1 Implementation in C#

```
private void STPad_SensorTimeoutOccured(object sender,
                                        SensorTimeoutOccuredEventArgs e)
{
    MessageBox.Show(String.Format("Timeout, captured points: {0}",
                                  e.pointsCount);
}
```

10.4.3.2 Implementation in Visual Basic

```
Private Sub STPad_SensorTimeoutOccured(ByVal sender As Object, _
                ByVal e As SensorTimeoutOccuredEventArgs) Handles _
                STPad.SensorTimeoutOccured
    MsgBox("Timeout, captured points: " & CStr(e.pointsCount)
End Sub
```

10.5 DisplayScrollPosChanged event

This event is called as soon as the scroll position of the display contents has changed.

Parameter	Values	Description
LONG nXPos int xPos xPos As Integer	>= 0	Horizontal offset of the display contents to the left, in pixels
LONG nYPos int yPos yPos As Integer	>= 0	Vertical offset of the display contents to the top, in pixels
Return value	**Values**	**Description**
-	-	-

10.5.1 STPadCapt.ocx

Available from Version 8.0.17.

```
void DisplayScrollPosChanged(LONG nXPos, LONG nYPos)
```

10.5.1.1 Implementation in C#

```
private void axSTPadCapt1_DisplayScrollPosChanged(object sender,
              AxSTPadCaptLib._DSTPadCaptEvents_DisplayScrollPosChanged e)
{
    MessageBox.Show(String.Format("Scroll pos: {0} / {1}", e.nXPos,
                                  e.nYPos);
}
```

10.5.1.2 Implementation in Visual Basic

```
Private Sub AxSTPadCapt1_DisplayScrollPosChanged _
             (ByVal eventSender As System.Object, ByVal e As _
             AxSTPadCaptLib._DSTPadCaptEvents_DisplayScrollPosChanged) _
             Handles AxSTPadCapt1.DisplayScrollPosChanged
    MsgBox("Scroll pos: " & CStr(e.nXPos) & " / " & CStr(e.nYPos))
End Sub
```

10.5.2 STPadLib.dll

Available from Version 8.0.19.

```
VOID DisplayScrollPosChanged(LONG nXPos, LONG nYPos)
```

10.5.2.1 Implementation in C++

```
VOID CMyClass::DisplayScrollPosChanged(LONG nXPos, LONG nYPos)
{
    wprintf(L"Scroll pos: %d / %d", nXPos, nYPos);
}
```

10.5.3 STPadLibNet.dll

Available from Version 8.0.19.

```
void raise_DisplayScrollPosChanged(object sender,
signotec.STPadLibNet.DisplayScrollPosChangedEventArgs e)
```

```
Event DisplayScrollPosChanged(ByVal sender As Object, ByVal e As
signotec.STPadLibNet.DisplayScrollPosChangedEventArgs)
```

The `DisplayScrollPosChangedEventArgs` class contains the following members described above:

```
public int xPos
public int yPos
```

```
Public xPos As Integer
Public yPos As Integer
```

10.5.3.1 Implementation in C#

```
private void STPad_DisplayScrollPosChanged(object sender,
                                    DisplayScrollPosChangedEventArgs e)
{
    MessageBox.Show(String.Format("Scroll pos: {0} / {1}", e.xPos,
                                  e.yPos);
}
```

10.5.3.2 Implementation in Visual Basic

```
Private Sub STPad_DisplayScrollPosChanged(ByVal sender As Object, _
               ByVal e As DisplayScrollPosChangedEventArgs) Handles _
               STPad.DisplayScrollPosChanged
    MsgBox("Scroll pos: " & CStr(e.xPos) & " / " & CStr(e.yPos))
End Sub
```