signotec

Documentation
Digital Signature Fields

(Add-on for the SignoAPI)

Generating and signing Adobe Acrobat-compliant digital signature fields in PDF documents

Version: *1.2.76*
Date: *26 October 2018*

Confidential
Subject to change without notice

Contents

1. SignPDF3 component: Signing and verifying DigSig fields in PDF documents/other functions for DigSig fields

Digital signature fields (DigSig fields) form part of the mechanism defined by Adobe® for digitally signing PDF documents on the basis of standard certificates. These digital signature fields are also referred to as Adobe-compliant signature fields. (DigSig = digital signature)

This component allows PDF documents to be digitally signed and important supplementary biometric information for the respective signer to be stored securely in a document.

Whenever a document is signed a new revision of the PDF document is generated and digitally signed. This ensures that the document can be checked for its intactness and genuineness in any DigSig-compatible PDF display program such as Adobe Acrobat® or Adobe® Reader, for example.

All the data captured for the signature digitalisation (including the signature's biometric data) is encrypted using RSA encryption and stored securely in the invisible area of the respective DigSig field. See also the detailed explanations of the used XML structures in the appendix.

1.1. System requirements

The component SignPDF3 is based on the Microsoft CryptoAPI (CAPI) (http://msdn.microsoft.com/en-us/library/aa380256(VS.85).aspx).
It uses inter alia the hash algorithms SHA256 and SHA512 which are currently claimed to be safe.
For that reason it is needed to use an actual *Cryptographic Service Provider*s (CSP) from Microsoft. A CSP which supports the needed algorithms is only available on Windows XP systems with Service Pack 3 installed or newer operating systems (The SP3 includes Updates for the Microsoft Kernel Mode Cryptographic Module).
To use the component it is mandatory to have Windows XP with Service Pack 3 or a newer operating system installed.
For Windows 2000 is such a CSP not available, so this component is not working under Windows 2000. On such systems, the setup will not install the SignPDF3.dll.

Important information:
The component signPDF3.dll has the following dependencies

1. The dependency is a DLL STPdfLib13.dll. It must be in the same folder with signPDF3.dll.
2. The COM dependency is the **signlib.dll**. Before the component signPDF3.dll is instantiated, the DLL must be registered. In the command line, run the command *regsvr32 signlib.dll*. The signlib.dll is only required in 32 bit version of signPDF3.dll and for compatibility reasons.
3. The methods that return a BSTR, require the following procedure.

```
//Get the actual PadID from the pad
virtual /* [helpstring][id][propget] */ HRESULT
STDMETHODCALLTYPE get_PadID(/* [retval][out] */ BSTR
__RPC_FAR *pVal) = 0;

HRESULT     hr          = E_FAIL;
BSTR        bstrPadID   = NULL;
```

```
hr = m_pSignPad->get_PadID(&bstrPadID);

if(bstrPadID != NULL)
{
      SysFreeString(bstrPadID);
      bstrPadID = NULL;
}
```

The returned BSTR must necessarily be released using SysFreeString.

1.2. Definition of the ISignDigSigPDF interface

Interface ISignDigSigPDF: IDispatch

1.3. Interface ID allocation

The IID for the interface:

Interface	IID
ISignDigSigPDF	8EBAA8A2-E603-4C66-AAF9-716BB6AD14BA

The CLSID for the interface:

Interface	CLSID
ISignDigSigPDF	6CF797C0-F930-4AC9-8E39-2968282964B4

1.4. Declaration information

The following declaration must be made when instantiating under Visual Basic (VB).

Interface	Declaration
ISignDigSigPDF	Signotec.signpdf3

1.5. Methods and properties

SignPdfDocument method

This method is deprecated. Please use SignPdfDocumentMemory() instead.

SignPdfDocumentMemory method

Electronic signing of a DigSig field in a PDF document using the signature data provided (SignData).
The captured signature and biometrics are inserted into the DigSig field, which is uniquely identified by its name.
If the name of the field does not yet exist, this is generated (positional information must then be set).

Note: For a DigSig field to be signed, two certificates must be passed to the methods. The first must be a PKCS12-certificate file with the private key for digitally signing the document according to the DigSig specifications. And the second must be a public X.509-certificate or PKCS12-certificate with the public

key for encrypting the signotec supplementary data (encryption of the biometrics).

Note: The signature data (SignData) must be passed from a previous signature capture on a signotec LCD signature pad.

Note: The signature image (Image) must be passed from a previous signature capture on a signotec LCD signature pad. For PDF/A-1b compliance, the signature image must not contain an alpha channel.

Note: The XML structure for the parameter "bstrXMLSignatureData" is specified in detail in the appendix.

Note: Based on the PDF file version, either the SHA256 or the SHA1 algorithm is used to create digital signatures. Documents with a PDF version higher than 1.5 will be signed by using at least the SHA256 algorithm, for older versions is this not allowed (see PDF specifications).

BSTR [in]	bstrPadID	Unique ID of the connected pad.
VARIANT* [in/out]	pvarPdfArray	PDF in a variant.
VARIANT (VT_ARRAY) [in]	varSignDataArray	The biometric data block to be inserted. (VT_ARRAY with SignData).
BSTR [in]	bstrSigfieldTitle	Unique name of the DigSig field to be signed.
VARIANT [in]	varSignCertArray	Data array of PKCS12-certificate with the private key for digitally signing the document (for DigSig mechanism).
BSTR [in]	bstrSignCertPassword	Password for the PKCS12-certificate data array with the private key for digitally signing the document (for DigSig mechanism).
VARIANT [in]	varCryptBioCertArray	Data array of the public X.509-certificate or PKCS12-certificate with the public key for encrypting the biometrics.
BSTR [in]	bstrCryptBioCertPassword	Optional password in case a PKCS12-certificate data array with a public key for encrypting the biometrics is to be used.
BSTR [in]	bstrXMLSignatureData	XML-coded character string with information that is to be additionally inserted (specified separately). See the chapter 2.4. XML structure with the input information needed to sign
VARIANT (VT_ARRAY) [in]	varImageArray	signature image, the be inserted in PDF document. For PDF/A-1b compliance, the signature image must not contain an alpha channel.
long [in]	lOptions	The parameter lOptions is a bitmask. The following bits are allowed: - 0x00: Do not sign the document

		in PDF/A compliant form - 0x01: Sign document compliantly PDF/A-1b - 0x02: Sign document conforming to PDF/A-2b - 0x04: Lock all form fields - 0x08: Only lock filled form fields Bits 0x01 and 0x02 must not both be set. Bits 0x04 and 0x08 must not both be set. Valid values are 0, 1, 2, 4, 5, 6, 8, 9 and 10. In all other cases, E_INVALIDARG is returned.
long* [in/out]	plReserved	Reserved, always 0.

```
HRESULT SignPdfDocumentMemory(      [in] BSTR     bstrPadID,
                                    [in] VARIANT*      pvarPdfArray,
                                    [in] VARIANT varSignDataArray,
                                    [in] BSTR     bstrSigfieldTitle,
                                    [in] VARIANT varSignCertArray,
                                    [in] BSTR     bstrSignCertPassword,
                                    [in] VARIANT varCryptBioCertArray,
                                    [in] BSTR     bstrCryptBioCertPassword,
                                    [in] BSTR     bstrXMLSignatureData,
                                    [in] VARIANT varImageArray,
                                    [in] long     lOptions,
                                    [in/out] long*plReserved);
```

PreparePdfDocumentMemory method

Prepare of electronic signing of a DigSig field in a PDF document using the specified signature data (SignData).

The PDF document is being prepared for signing. The captured signature curve and biometrics are encrypted and inserted into the DigSig field that is uniquely identified by the name. Encrypting of biometrics can both in the pad than even before the method is called. Signing can be done as well in the pad or after calling the method. If the name of the field not exists, so this will be created (position information of the field must then be used). The method prepares a placeholder for later insertion of the digital signature in the PDF document and determines its position. The method returns the prepared PDF file into a byte array and the placeholder position.

After calling the method, a SHA256 hash must be made via the byte array that contains the prepared PDF document. The hash must be signed then. This creates a digital signature. The generated signature must be written at the location of the placeholder in the PDF document.

Note: To encrypt the signotec Additional data (biometric encryption) a public X.509-cerificate or PKCS12-public key certificate is used. If a PKCS12-certificate is used, the password must be passed in the parameter "bstrCryptBioCertPassword".

Note: The signature data (SignData) must be taken from the headed signature capture on a signotec LCD signature pad.

Note: The signature image (Image) must be applied from previous signature capture on a signotec LCD signature pad. For PDF/A-1b compliance, the signature image must not contain an alpha channel.

Note: The XML structure for the parameter "bstrXMLSignatureData" is described in detail in the appendix.

BSTR [in]	bstrPadID	Unique ID of the connected pad.
VARIANT* [in/out]	pvarPdfArray	PDF in a variant.
VARIANT (VT_ARRAY) [in]	varSignDataArray	The biometric data block to be inserted. (VT_ARRAY with SignData).
BSTR [in]	bstrSigfieldTitle	Unique name of the DigSig field to be signed.
VARIANT [in]	varSignCertArray	Data array by PKCS12-certificate with the public key to determine its properties.
VARIANT [in]	varCryptBioCertArrayOrRefString	Data array by public X.509-certificate or PKCS12-certificate with the public key used to encrypt the biometrics.
BSTR [in]	bstrCryptBioCertPassword	Password when the array of PKCS12-certificate is used with the public key to encrypt the biometrics. If the biometric data is encrypted in the pad, certificate ID of the pad certificate in a BSTR string must be passed.
BSTR [in]	bstrXMLSignatureData	XML-coded character string with information that is to be additionally inserted (specified separately). See the chapter 2.4. XML structure with the input information needed to sign
VARIANT (VT_ARRAY) [in]	varImageArray	Signature image that is inserted into a PDF document. For PDF/A-1b compliance, the signature image must not contain an alpha channel.

VARIANT*(VT_ARRAY) [out]	pvarDocHashArray	The data array contains the PDF document that is returned. A SHA256 hash must be made about the data. The hash must be then signed and written into the document.
long* [out]	plDocSignaturePosition	The offset of the placeholder for the signature in the PDF document. The signed hash (digital signature) must be written to the location in the document to complete the signing of DigSig field.
long [in]	lOptions	The parameter lOptions is a bitmask. The following bits are allowed: - 0x00: Do not sign the document in PDF/A compliant form - 0x01: Sign document compliantly PDF/A-1b - 0x02: Sign document conforming to PDF/A-2b - 0x04: Lock all form fields - 0x08: Only lock filled form fields Bits 0x01 and 0x02 must not both be set. Bits 0x04 and 0x08 must not both be set. Valid values are 0, 1, 2, 4, 5, 6, 8, 9 and 10. In all other cases, E_INVALIDARG is returned.
long* [in/out]	plReserved	Reserved, always 0.

HRESULT **PreparePdfDocumentMemory**([in] BSTR bstrPadID,
[in/out] VARIANT* pvarPdfArray,
[in] VARIANT varSignDataArray,
[in] BSTR bstrSigfieldTitle,
[in] VARIANT varSignCertArray,
[in] VARIANT
varCryptBioCertArrayOrRefString,
[in] BSTR bstrCryptBioCertPassword,
[in] BSTR bstrXMLSignatureData,
[in] VARIANT varImageArray,

```
[out] VARIANT*      pvarDocHashArray,
[out] long*    plDocSignaturePosition,
[in] long      lOptions,
[in/out] long* plReserved);
```

PreparePDFDocumentSigningMemory method

Prepare of electronic signing of a DigSig field in a PDF document.

The PDF document is being prepared for signing. The method prepares a placeholder for later insertion of the digital signature in the PDF document and determines its position. The method returns the prepared PDF file into a byte array and the placeholder position.

After calling the method, a SHA256 hash must be made via the byte array that contains the prepared PDF document. The hash must be signed then. This creates a digital signature. The generated signature must be written at the location of the placeholder in the PDF document.

Note: The XML structure for the parameter "bstrXMLSignatureData" is described in detail in the appendix.

VARIANT* [in/out]	pvarPdfArray	PDF in a variant.
BSTR [in]	bstrSigfieldTitle	Unique name of the DigSig field to be signed.
BSTR [in]	bstrXMLSignatureData	XML-coded character string with information that is to be additionally inserted (specified separately). See the chapter 2.4. XML structure with the input information needed to sign
VARIANT (VT_ARRAY) [in]	varImageArray	Signature image that is inserted into a PDF document. For PDF/A-1b compliance, the signature image must not contain an alpha channel.
VARIANT*(VT_ARRAY) [out]	pvarDocHashArray	The data array contains the PDF document that is returned. A SHA256 hash must be made about the data. The hash must be then signed and written into the document.
long* [out]	plDocSignaturePosition	The offset of the placeholder for the signature in the PDF document. The signed hash (digital signature) must be written to the location in the document to complete the signing of DigSig field.
long [in]	lOptions	The parameter lOptions is a bitmask. The following bits are allowed: - 0x00: Do not sign the

		document in PDF/A compliant form - 0x01: Sign document compliantly PDF/A-1b - 0x02: Sign document conforming to PDF/A-2b - 0x04: Lock all form fields - 0x08: Only lock filled form fields Bits 0x01 and 0x02 must not both be set. Bits 0x04 and 0x08 must not both be set. Valid values are 0, 1, 2, 4, 5, 6, 8, 9 and 10. In all other cases, E_INVALIDARG is returned.
long* [in/out]	plReserved	Reserved, always 0.

```
HRESULT PreparePDFDocumentSigningMemory (
                    [in/out] VARIANT*   pvarPdfArray,
                    [in] BSTR       bstrSigfieldTitle,
                    [in] BSTR       bstrXMLSignatureData,
                    [in] VARIANT        varImageArray,
                    [out] VARIANT*      pvarDocHashArray,
                    [out] long*     plDocSignaturePosition,
                    [in] long       lOptions,
                    [in/out] long*plReserved);
```

GetReferenceCount method

This method is deprecated. Please use GetReferenceCountMemory() instead.

GetReferenceCountMemory method

Determines the number of DigSig fields that are present (all or only those that are filled) on the specified page of the specified PDF document.

VARIANT [in]	varPdfArray	PDF in a variant
int [in]	nPage	Specifies which page should be searched. -1 returns all fields in the whole document.
long [out]	*plCount	[in] Pointer to a long value. [out] Number of DigSig fields found.
int [in]	nOption	0 = only filled fields, 1 = all fields count.

```
HRESULT GetReferenceCountMemory(        [in]    VARIANT varPdfArray,
                    [in]    int     nPage,
                    [out]   long*   plCount,
                    [in]    int     nOption);
```

GetReference method

This method is deprecated. Please use GetReferenceMemory() instead.

GetReferenceMemory method

Extracts and decrypts the supplementary data from a signed DigSig field in the specified PDF document. The DigSig field is identified via its specified unique name.

Note: In order to decrypt the supplementary data (SignData and supplementary information) a PKCS12-certificate file with the private key for decrypting the biometrics must be specified. This must be suitable for the public key that was used for the encryption.
Note: The XML structure for the parameter "pbstrXMLResultData" is specified in detail in the appendix.

VARIANT [in]	varPdfArray	PDF in a variant.
int [in]	nReserved	Reserved. Always 0.
BSTR [in]	bstrSigfieldTitle	Unique name of the DigSig field that is to be read out.
VARIANT [in]	varPFXCertArray	File name and path to the PKCS12-certificate file with the private key for decrypting the biometric data.
BSTR [in]	bstrPFXPassword	Password for the PKCS12-certificate file with the private key for decrypting the biometrics.
VARIANT* (VT_ARRAY) [out]	pvarSigndata	[in] Pointer to an empty VARIANT. [out] Pointer to a VARIANT of type VT_ARRAY that holds the data of the SignData structure.
BSTR* [out]	pbstrXMLResultData	Output of an XML-coded character string with information on the signature process.
int [in]	nOption	0 = Default, 1 = Advanced signature/integrity hash checking (see XML structure).

```
HRESULT GetReferenceMemory(     [in]    VARIANT varPdfArray,
                                [in]    int     nReserved,
                                [in]    BSTR    bstrSigfieldTitle,
                                [in]    VARIANT varPFXCertArray,
                                [in]    BSTR    bstrPFXPassword,
                                [out]   VARIANT* pvarSigndata,
                                [out]   BSTR*   pbstrXMLResultData,
                                [in]    int     nOption);
```

VerifyPdfDocument method

This method is deprecated. Please use VerifyPdfDocumentMemory() instead.

VerifyPdfDocumentMemory method

Extracts and decrypts the digital signature data from the signed DigSig fields in the specified PDF document and checks that the fields are valid and intact.

Note: The XML structure for the parameter "pbstrXMLResultData" is specified in detail in the appendix.
Note: This method does not return the encrypted biometric data of previously signed DigSig fields. This encrypted information can only be obtained using the GetReference() method and the respective certificate.

VARIANT [in]	varPdfArray	PDF in a variant.
BSTR* [out]	pbstrXMLResultData	Output of an XML-coded character string with information on the signed DigSig fields.
long* [out]	plStatus	[out] Pointer to a long value. Status (currently always 0).

HRESULT **VerifyPdfDocumentMemory**([in] VARIANT varPdfArray,
[out] BSTR* pbstrXMLResultData,
[out] long* plStatus);

GetDSFieldInfo method

This method is deprecated. Please use GetDSFieldInfoMemory() instead.

GetDSFieldInfoMemory method

Extracts the data from all the DigSig fields in the specified PDF document and displays the details for these fields.

Note: The XML structure for the parameter "pbstrXMLResultData" is specified in detail in the appendix.
Note: This method does not return the encrypted biometric data of previously signed DigSig fields. This information can only be obtained using the GetReference() method.

VARIANT [in]	varPdfArray	PDF in a variant.
BSTR* [out]	pbstrXMLResultData	Output of an XML-coded character string with information on the signature process.
int [in]	nReserved	Reserved, always 0.

HRESULT **GetDSFieldInfoMemory**([in] VARIANT varPdfArray,
[out] BSTR* pbstrXMLResultData,
[in] int nReserved);

SetLicenseKey method

Sets a hardware-independent licence key permitting the use of this interface. A temporary key can be requested for testing purposes or a full licence can be purchased.

Note: This interface cannot be used without a valid key.

BSTR [in]	bstrLicenseKey	Licence key as a character string (BSTR).

HRESULT **SetLicenseKey**([in] BSTR bstrLicenseKey);

VerifyCertificate method

This method is deprecated. Please use VerifyCertificateMemory() instead.

VerifyCertificateMemory method

Validates the certificate and returns its status.

VARIANT [in]	varCertArray	Certificate in a variant
BSTR [in]	bstrOptionalPFXPassword	Password for the certificate, the parameter is optional
int [in]	nReserved	Reserved, always 0
long* [out]	plCertStatus	Certificate status. See chapter: signotec tags of status of certificate

HRESULT **VerifyCertificateMemory**([in] VARIANT varCertArray,
[in] BSTR bstrOptionalPFXPassword,
[in] int nReserved,
[out] long* plCertStatus);

CreatePKCS12CertificateMemory method

Generates a pfx certificate and stores it in a file.

BSTR [in]	bstrSubject	X.500 - string. The string contains a definition of various atributen the certificate (RFC 2253). An example of X.500 - String: "CN=Siggi Pinsel, L=Ratingen, O=signotec GmbH, OU=signosign, Email=sigcert@signotec.de, C=DE, ST=NRW, STREET=Am Gierath, Title=Doctor of Science, GivenName=Siggi Genius, Initials=S.P.G, SN=Pinsel, DC=signotec"
BSTR [in]	bstrPassword	Password for the new certificate
short [in]	nRSABits	Key length, possible values 1024, 2048
short [in]	nValidYears	The period of validity in years
VARIANT* [out]	pvarCertArray	Variant with a newly generated certificate as a byte array

HRESULT **CreatePKCS12CertificateMemory** ([in] BSTR bstrSubject,
[in] BSTR bstrPassword,
[in] short nRSABits,
[in] short nValidYears,

[out] VARIANT* pvarCertArray);

CreateDSFieldMemory method

Creates an empty not signed signature field.

VARIANT* [in/out]	pvarPdfArray	PDF in a variant.
BSTR [in]	bstrTitle	Unique name of the signature field in the PDF document.
int [in]	nPage	The page of the PDF, where the new signature field is generated.
double [in]	dX	X - coordinate of the upper left corner of the new signature field. The origin of the coordinate system is in the upper left corner of the PDF.
double [in]	dY	Y - coordinate of the upper left corner of the new signature field. The origin of the coordinate system is in the upper left corner of the PDF.
double [in]	dWidth	The width of the new signature field.
double [in]	dHeight	The height of the new signature field.
int nFlags [in]	nFlags	Reserved for future use. Now it is not used.

HRESULT **CreateDSFieldMemory** ([in] VARIANT* pvarPdfArray,
[in] BSTR bstrTitle,
[in] int nPage,
[in] double dX,
[in] double dY,
[in] double dWidth,
[in] double dHeight,
[in] int nFlags);

DrawTextBoxPdfDocumentMemory method

Adds an arbitrary text into an existing PDF document.
The desired page and the position of the text must be defined, by giving the corners of a rectangle (the actual text box).
Font name, font size, font color and emphasis can be chosen.
The text will not be framed and will be brought in transparent so that other texts are not masked. The text is introduced as a string, not as an image.

Note: The method should not be used on already signed documents, otherwise the previously given signatures will be invalidated, and to bring in another signature will be impossible.
Note: There is an automatic text wrap when the width of the specified text box is too small for the length of the text.
Note: There is no automatic scaling of the font size to the size of the text box. The developer must itself ensure that the space requirement of the text does not

exceed the size of the text box. Otherwise, the non-displayable portion of the text is truncated.

Note: If bstrFontName is empty, the parameter nStandardFont determines what standard font is used.

nStandardFont	Standard Font
0	Courier
1	CourierBold
2	CourierBoldOblique
3	CourierOblique
4	Helvetica
5	HelveticaBold
6	HelveticaBoldOblique
7	HelveticaOblique
8	TimesRoman
9	TimesBold
10	TimesItalic
11	TimesBoldItalic
12	Symbol
13	ZapfDingbats

VARIANT* [in/out]	pvarPdfArray	PDF byte array in a variant.
BSTR [in]	bstrText	Text which should be inserted.
short [in]	xPos1	Upper left corner of the text box in points (X coordinate).
short [in]	yPos1	Upper left corner of the text box in points (Y coordinate).
short [in]	xPos2	Lower right corner of the text box in points (X coordinate).
short [in]	yPos2	Lower right corner of the text box in points (Y coordinate).
int [in]	nPage	Page of the document on which the text is to be inserted.
BSTR [in]	bstrFontName	name of the font as wide-char string. If the string is empty, one of the standard fonts is used. The last parameter determines which then.
double [in]	dbFontSize	Font size as double-value
VARIANT_BOOL [in]	bFontBold	Use text emphasis bold. Possible values: VARIANT_TRUE or VARIANT_FALSE
VARIANT_BOOL [in]	bFontItalic	Use text emphasis italic. Possible values: VARIANT_TRUE or VARIANT_FALSE
VARIANT_BOOL [in]	bUnderline	Use text emphasis Underline. Possible values: VARIANT_TRUE or VARIANT_FALSE
OLE_COLOR [in]	oleColor	Color of the text be inserted as OLE_COLOR structure (Note the order of the text color BGR) 0x00FF0000 for pure blue; 0x0000FF00 for pure green; 0x000000FF for pure red;
int [in]	nStandardFont	If bstrFontName is empty, determines which standard font must be used.

HRESULT DrawTextBoxPdfDocumentMemory

```
(
        [in/out] VARIANT* pvarPdfArray,
        [in] BSTR bstrText,
        [in] short xPos1,
        [in] short yPos1,
        [in] short xPos2,
        [in] short yPos2,
        [in] int nPage,
        [in] BSTR bstrFontName,
        [in] double dbFontSize,
        [in] VARIANT_BOOL bFontBold,
        [in] VARIANT_BOOL bFontItalic,
        [in] VARIANT_BOOL bUnderline,
        [in] OLE_COLOR oleColor,
        [in] int nStandardFont
);
```

get_SignaturePlaceholderLength property

gets the length in bytes of the placeholder in the PDF document which will later contain the signature as hex string. That means the byte array of the signature must not be longer than the half of this value. The default value is 8192, the property must be set before the methods PreparePdfDocumentMemory or PreparePDFDocumentSigningMemory are called, if another size is required.

long* [out]	pVal	length in bytes of the placeholder.

HRESULT **get_SignaturePlaceholderLength** ([out] long *pVal);

set_SignaturePlaceholderLength property

sets the length in bytes of the placeholder in the PDF document which will later contain the signature as hex string. That means the byte array of the signature must not be longer than the half of this value. The default value is 8192, the property must be set before the methods PreparePdfDocumentMemory or PreparePDFDocumentSigningMemory are called, if another size is required.

Long [in]	newVal	length in bytes of the placeholder.

HRESULT **set_SignaturePlaceholderLength** ([in] long newVal);

2. Used XML structures in the component SignPDF3

2.1. XML structure to check if a DigSig field is valid

XML structure and enclosed elements which are returned from a call of the method **VerifyPdfDocument()** or **VerifyPdfDocumentMemory()** of the component SignPDF3.

Example:

```
<?xml version="1.0" encoding="utf-8" ?>
<digsignatures>
<digsignature Name="sgnsignature_1">
<Name>Max Mustermann</Name>
<Reason>Ich bin mit dem Inhalt einverstanden.</Reason>
<Location>Wien< /Location>
<ContactInfo>signotec GmbH</ContactInfo>
<Time>D:20090211141611</Time>
<Page>1</Page>
<Mandatory>false</Mandatory>
<Subfilter>adbe.x509.rsa_sha1</Subfilter>
<Filter>Adobe.PPKLite</Filter>
<HashAlgorithm>SHA256</HashAlgorithm>
<Status>1</Status>
<CertExpired>0</CertExpired>
<Rect>
        <Left>323</Left>
        <Right>497</Right>
        <Top>271</Top>
        <Bottom>325</Bottom>
</Rect>
<Certificate>
        <Issuer>C=DE, CN=DEMO , O=DEMO AG</Issuer>
        <Serial>48a9fe5043ea1e854ecf1caa89cf5b60</Serial>
        <PublicKeysize>2048</PublicKeysize>
        <ValidTo>2012/01/07 20:23:44</ValidTo>
        <ValidFrom>2008/01/07 20:23:44</ValidFrom>
        <CertErrorStatus>The certificate or certificate chain is based on an
   untrusted root;</CertErrorStatus>
        <CertErrorStatusCode>32</CertErrorStatusCode>
</Certificate>
<SIGNATURE_INFO>
        <COMPANY>signotec GmbH</COMPANY>
        <VERSION>8.0.0.88</VERSION>
        <SIGN_TIME>D:20130924093040+02'00'</SIGN_TIME>
        <USERID>ws</USERID>
        <MACHINE>OFFICE_1169</MACHINE>
        <MACADDRESS>0017F991241A</MACADDRESS>
        <PADID>1000013325</PADID>
        <PADMODEL>Sigma HID</PADMODEL>
        <PADTYPE>101</PADTYPE>
        <ADDREFERENCE>TRUE</ADDREFERENCE>
        <CERT>
                <CIPHERENC_FILENAME>Demo1_Cert.p12</CIPHERENC_FILENAME>
                <BIOENC_FILENAME>Demo2_BIO.pem</BIOENC_FILENAME>
```

```
        </CERT>
</SIGNATURE_INFO>
</digsignature>
<digsignature Name="sgnsignature_2">
        <Next signature field like sgnsignature_1>
</digsignatures>
```

Elements <digsignatures> and <digsignature>

The root element of this XML document is <digsignatures>. A <digsignature> element is present for each signature field beneath this root element. These elements contain the following elements:

Element <Name> (optional)

The Name element holds the name of the signer or the automatically determined name for which the certificate has been issued. Standard display – also in Adobe Acrobat® Reader.

Element <Reason> (optional)

The Reason element holds a brief description explaining why a document has been signed. Standard display – also in Adobe Acrobat® Reader.

Element <Location> (optional)

The Location element holds a brief description of the location. Standard display – also in Adobe Acrobat® Reader.

Element <ContactInfo> (optional)

The ContactInfo element optionally holds a name or contact information that is automatically determined from the certificate. Standard display – also in Adobe Acrobat® Reader under Certificate display.

Element <Time>

This element holds the timestamp at which the signature field was signed (in Adobe® format).

Element <Page>

This element holds the page in the PDF on which the signature field is located.

Element <Mandatory>

This element specifies whether the field is a mandatory signature field.

Element <SubFilter>

This element specifies the method used to insert the signature. The standard format here is '**adbe.x509.rsa_sha1**'.

Element <Filter>

This element specifies the standard checking method that may be used for checking. The standard method here is '**Adobe.PPKLite**'.

Element <HashAlgorithm>

This element describes the hash algorithm which was used for the digital signature (the value depends on the file version of the PDF).

Element <Status>

This element specifies the status of the signature field based on the document revision. The following values may be returned:

STATUS_VALID	0	Signature is valid.
STATUS_CHANGEDAFTERSIGNING	1	Signature is valid (changed afterwards).
STATUS_INVALID	2	Signature is invalid.
STATUS_EMPTY	3	Signature field is empty (not signed).
STATUS_UNKNOWN	4	Cannot be checked (unknown format).

Element <CertExpired>

This element specifies the status of the certificate. The expiry date of the embedded public section of the certificate is checked.

0 = OK.
1 = Expired certificate.

Element <Rect/Left> (optional, only if the signature field is not yet present)

This element holds the coordinate of the left side of the signature field in pixels relative to the upper left corner of the PDF document page.

Element <Rect/Right> (optional, only if the signature field is not yet present)

This element holds the coordinate of the right side of the signature field in pixels relative to the upper left corner of the PDF document page.

Element <Rect/Top> (optional, only if the signature field is not yet present)

This element holds the coordinate of the top side of the signature field in pixels relative to the upper left corner of the PDF document page.

Element <Rect/Bottom> (optional, only if the signature field is not yet present)

This element holds the coordinate of the bottom side of the signature field in pixels relative to the upper left corner of the PDF document page.

Element <Certificate/Issuer>

This element holds identifiers for the embedded certificate.
The following standardised identifiers may be included:

CN	The certificate owner's common name
E	The certificate owner's e-mail address
T	The certificate owner's locality
ST	The certificate owner's state of residence
O	The organisation to which the certificate owner belongs
OU	The name of the organisational unit to which the certificate owner belongs

C The certificate owner's country of residence
STREET The certificate owner's street address
ALL The certificate owner's complete distinguished name

Element < Certificate/Serial>

This element contains the serial number of the certificate used.

Element <Certificate/PublicKeySize>

This element holds the RSA encryption key size such as '2048', for example.

Element <Certificate/ValidTo>

This element holds the end of the validity period for the used certificate.

Element <Certificate/ValidFrom>

This element holds the beginning of the validity period for the used certificate.

Element <Certificate/CertErrorStatus>

This element contains the error status of the used certificate. The error status is a character string. The string contains semicolon-delimited error descriptions of field *DWORD dwErrorStatus*; in the *CERT_TRUST_STATUS* structure.

Element <Certificate/CertErrorStatusCode>

This element contains the error status code of the used certificate. The error code is the value of the DWORD dwErrorStatus; in the *CERT_TRUST_STATUS* structure.

Element <SIGNATURE_INFO/COMPANY>

This element specifies the manufacturer of the API and is always set to 'signotec GmbH'.

Element <SIGNATURE_INFO/VERSION>

This element specifies the version of the API that was used for signing.

Element <SIGNATURE_INFO/SIGN_TIME>

This element specifies the time at which signing took place.

Element <SIGNATURE_INFO/USERID>

This element holds the name of the logged-on user.

Element <SIGNATURE_INFO/ADDREFERENCE>

This optional element is intended for internal use only.

Element <SIGNATURE_INFO/MACHINE>

This element holds the name of the PC on which signing took place.

Element <SIGNATURE_INFO/MACADDRESS>

This element holds the unique MAC address of the PC on which signing took place.

Element <SIGNATURE_INFO/PADID>

This element holds the unique device ID of the connected pad.

Element <SIGNATURE_INFO/PADMODEL>

This element holds the device type that was used for signing.

Element <CERT/CIPHERENC_FILENAME>

This element holds the name of the P12 certificate that was used for signing the document.

Element <CERT/BIOENC_FILENAME>

This element holds the name of the certificate that was used for encrypting the biometrics.

2.2. XML structure with the supplementary data of a DigSig field after RSA decryption

XML structure and enclosed elements which are returned from a call of the method **GetReference()** or **GetReferenceMemory()** of the component SignPDF3.

Example of signing and encrypted in the signPDF3 component:

```
<?xml version="1.0" encoding="utf-8" ?>
<SIGNATURE_INFO>
    <COMPANY>signotec GmbH</COMPANY>
    <VERSION>8.00.00.031</VERSION>
    <SIGN_TIME>11.02.2009 14:16:11</SIGN_TIME>
    <USERID>Max Mustermann</USERID>
    <ADDREFERENCE>TRUE</ADDREFERENCE>
    <MACHINE>OFFICE_1169</MACHINE>
    <MACADDRESS>0017F991241A</MACADDRESS>
    <PADID>1000013325</PADID>
    <PADMODEL>Sigma HID</PADMODEL>
    <PADTYPE>101</PADTYPE>
    <CERT>
    <CIPHERENC_FILENAME>DEMO1_CERT.p12</CIPHERENC_FILENAME>
    <BIOENC_FILENAME>DEMO2_BIO.cer</BIOENC_FILENAME>
    </CERT>
    <BIOMETRIC_INTEGRITY>
        <DOC-HASH_VALUE>
        45BF6B0822BA17A2E16E1AD222B1073E721731F3E9097F4E0F285
        DAF2D206D63</DOC-HASH_VALUE>
        <DOC-HASH_RECALCEDVALUE>
        45BF6B0822BA17A2E16E1AD222B1073E721731F3E9097F4E0F285
        DAF2D206D63</DOC-HASH_RECALCEDVALUE>
        <DOC-HASH_ALGO>SHA256</DOC-HASH_ALGO>
        <TIMESTAMP>D:20090211141611</TIMESTAMP>
        <MACHINE>OFFICE_1169</MACHINE>
        <MACADDRESS>0017F991241A</MACADDRESS>
        <PADID>1000013325</PADID>
        <PADMODEL>101</PADMODEL>
```

```
    </BIOMETRIC_INTEGRITY>
</SIGNATURE_INFO>
```

Example of signing and encrypting in Pad:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SIGNATURE_INFO>
    <COMPANY>signotec GmbH</COMPANY>
    <VERSION>8.1.2.6</VERSION>
    <SIGN_TIME>D:20180927140151+02'00'</SIGN_TIME>
    <USERID>Max Mustermann</USERID>
    <MACHINE>MAX_MUSTERMANN-PC </MACHINE>
    <MACADDRESS>FCAA1F07B0B1</MACADDRESS>
    <PADID>1990092846</PADID>
    <PADTYPE>101</PADTYPE>
    <PADMODEL>Sigma HID</PADMODEL>
    <BIOMETRIC_INTEGRITY>
        <DOC-
        HASH_VALUE>1230DAE8D50FDA3452E5536D4262A43595290CBBD
        E823C9A6E57CE5F0A5A7ECC</DOC-HASH_VALUE>
        <DOC-
        HASH_RECALCEDVALUE>1230DAE8D50FDA3452E5536D4262A43595
        290CBBDE823C9A6E57CE5F0A5A7ECC</DOC-
        HASH_RECALCEDVALUE>
        <DOC-HASH_ALGO>SHA256</DOC-HASH_ALGO>
        <BIO-
        HASH_VALUE>C146F79C211E5676AE13BBF41E950E1A8D95C41828
        2C188AA3E5CA6F96355C18</BIO-HASH_VALUE>
        <BIO-HASH_ALGO>SHA256</BIO-HASH_ALGO>
        <TIMESTAMP>20180927140151+02'00'</TIMESTAMP>
        <MACHINE>MAX_MUSTERMANN-PC </MACHINE>
        <USERNAME>Max Mustermann</USERNAME>
        <PADID>1990092846</PADID>
        <PADMODEL>Sigma HID</PADMODEL>
        <CONTENTLENGTH>8781</CONTENTLENGTH>
        <HASHTYPE>COMBINATION</HASHTYPE>
        <RSA-SCHEME>PSS</RSA-SCHEME>
        <RSA-
        SIGNATURE>EaLT2MVA+ez9apYZuARZNglq9pL+OBLGRQa4AZK3CjC
        Rosoxq9pzRwARVPuA+XOQgE64vk2ll3ccyNSIYpM4gzHgT4pbN2zp
        C70+Q8X4CBP98AMh2EAh3aoseYJ5jLX+6jSrhUjPBUNYoZ03s5UlBY
        4LJ/WautWpXbFQPO7O8Xg=</RSA-SIGNATURE>
          <RSA-SIGNATURE_STATUS>0</RSA-SIGNATURE_STATUS>
    </BIOMETRIC_INTEGRITY>
</SIGNATURE_INFO>
```

Element <SIGNATURE_INFO>

The root element of this XML document is <SIGNATURE_INFO>. The following attributes appear beneath this root element:

Element <COMPANY> (from the unencrypted section)

This element specifies the manufacturer of the API and is always set to 'signotec GmbH'.

Element <VERSION> (from the unencrypted section)

This element specifies the version of the API that was used for signing.

Element <SIGN_TIME> (from the unencrypted section)

This element specifies the time at which signing took place.

Element <USERID> (from the unencrypted section)

This element holds the name of the logged-on user.

Element <ADDREFERENCE> (optional, from the unencrypted section)

This element is intended for internal use only.

Element <MACHINE> (from the unencrypted section)

This element holds the name of the PC on which signing took place.

Element <MACADDRESS> (from the unencrypted section)

This element holds the unique MAC address of the PC on which signing took place.

Element <PADID> (from the unencrypted section)

This element holds the unique device ID of the connected pad.

Element <PADMODEL> (from the unencrypted section)

This element holds the device model that was used for signing.

Element <PADTYPE> (from the unencrypted section)

This element contains the device type with which it was signed.

Element <CERT/CIPHERENC_FILENAME> (from the unencrypted section)

This element holds the name of the P12 certificate that was used for signing the document.

Element <CERT/BIOENC_FILENAME> (from the unencrypted section)

This element holds the name of the certificate that was used for encrypting the biometrics.

The following XML data is only output when the options parameter is set to '1' (when encrypting in the pad, the following XML data is always output):

Element <BIOMETRIC_INTEGRITY/DOC-HASH_VALUE> (from the encrypted section)

This element holds the hash of the document before the digital signature and biometrics have been inserted. This hash is located in the encrypted section.

Element <BIOMETRIC_INTEGRITY/DOC-HASH_RECALCEDVALUE>

This element holds the checked hash of the document before the digital signature and biometrics have been inserted. The value of this element must be the same as the value of 'DOC-HASH-VALUE' in the encrypted section.

Element <BIOMETRIC_INTEGRITY/DOC-HASH_ALGO> (from the encrypted section)

This element holds the hash algorithm that is used. For internal use.

Element <BIOMETRIC_INTEGRITY/BIO-HASH_VALUE> (from the encrypted section)

This element contains the hash of the biometric data.

Element <BIOMETRIC_INTEGRITY/BIO-HASH_ALGO> (from the encrypted section)

This element contains the used hash algorithm. Internal Use. Always SHA256 at the moment.

Element <BIOMETRIC_INTEGRITY/TIMESTAMP> (from the encrypted section)

This element describes when it was signed.

Element <BIOMETRIC_INTEGRITY/MACHINE> (from the encrypted section)

This element holds the name of the PC on which signing took place. The value of this element must be the same as the value of the element in the unencrypted section.

Element <BIOMETRIC_INTEGRITY/USERNAME> (from the encrypted section)

This element contains the name of the logged-in user.

Element <BIOMETRIC_INTEGRITY/MACADDRESS> (from the encrypted section)

This element holds the unique MAC address of the PC on which signing took place. The value of this element must be the same as the value of the element in the unencrypted section.

Element <BIOMETRIC_INTEGRITY/PADID> (from the encrypted section)

This element holds the unique device ID of the connected pad.
The value of this element must be the same as the value of the element in the unencrypted section.

Element <BIOMETRIC_INTEGRITY/PADMODEL> (from the encrypted section)

This element holds the device type that was used for signing.
The value of this element must be the same as the value of the element in the unencrypted section.

Element <BIOMETRIC_INTEGRITY/CONTENTLENGTH> (from the encrypted section)

This element contains the length of the area over which the document hash was calculated.

Element <BIOMETRIC_INTEGRITY/HASHTYPE> (from the encrypted section)

This element contains the hash type with which the RSA signature was calculated. There are three types "COMBINATION", "HASH1" and "HASH2", for more details please see the demo application "SignoAPIDigSigMemDemoCSharp".

Element <BIOMETRIC_INTEGRITY/RSA-SCHEME> (from the encrypted section)

This element contains the RSA schema.

Element <BIOMETRIC_INTEGRITY/RSA-SIGNATURE> (from the encrypted section)

This element contains the RSA signature, for more details see the demo application "SignoAPIDigSigMemDemoCSharp".

Element < BIOMETRIC_INTEGRITY/RSA-SIGNATURE_STATUS> (from the encrypted section)

This element contains the status of the RSA signature. It can contain the following values:
"0" - the RSA signature is intact,
"2" - the RSA signature is not intact, the data was unauthorized manipulated.

2.3. XML structure with general information for an empty, unsigned DigSig field

XML structure and enclosed elements which are returned from a call of the method **GetDSFieldInfo()** or **GetDSFieldInfoMemory()** of the component SignPDF3.

Example:

```
<?xml version="1.0" encoding="utf-8" ?>
<digsignatures>
    <digsignature Name="sgnsignature_1">
    <Name />
    <Reason />
    <Location />
    <ContactInfo />
    <Time />
    <Page>1</Page>
    <Mandatory>false</Mandatory>
    <Subfilter />
    <Filter />
    <HashAlgorithm />
    <Status>3</Status>
    <CertExpired>0</CertExpired>
    <Rect>
        <Left>323</Left>
        <Right>497</Right>
        <Top>271</Top>
        <Bottom>325</Bottom>
    </Rect>
    <Certificate>
        <Issuer />
        <Serial />
        <PublicKeysize />
        <ValidTo />
        <ValidFrom />
```

```
            <CertErrorStatus />
            <CertErrorStatusCode>0</CertErrorStatusCode>
      </Certificate>
      </digsignature>
 <digsignature Name="sgnsignature_x">
   <next signature field like above>
  </digsignature>
 </digsignatures>
```

Elements <digsignatures> and <digsignature>

The root element of this XML document is <digsignatures>. A <digsignature> element is present for each signature field beneath this root element. These elements contain the following elements:

Element <Name> (empty)

The Name element holds the name of the signer or the automatically determined name for which the certificate has been issued. Standard display – also in Adobe Acrobat® Reader.

Element <Reason> (empty)

The Reason element holds a brief description explaining why a document has been signed. Standard display – also in Adobe Acrobat® Reader.

Element <Location> (empty)

The Location element holds a brief description of the location. Standard display – also in Adobe Acrobat® Reader.

Element <ContactInfo> (empty)

The ContactInfo element optionally holds a name or contact information that is automatically determined from the certificate. Standard display – also in Adobe Acrobat® Reader under Certificate display.

Element <Time> (empty)

This element holds the timestamp at which the signature field was signed (in Adobe® format).

Element <Page>

This element holds the page in the PDF on which the signature field is located.

Element <Mandatory>

This element specifies whether the field is a mandatory signature field.

Element <SubFilter> (empty)

This element specifies the method used to insert the signature. The standard format here is '**adbe.x509.rsa_sha1**'.

Element <Filter> (empty)

This element specifies the standard checking method that may be used for checking. The standard method here is '**Adobe.PPKLite**'.

Element < HashAlgorithm> (empty)

This element specifies the hash algorithm used to insert the signature.

Element <Status> (STATUS_EMPTY = 3)

This element specifies the status of the signature field based on the document revision. The following values may be returned:

STATUS_VALID	0	Signature is valid.
STATUS_CHANGEDAFTERSIGNING	1	Signature is valid (changed afterwards).
STATUS_INVALID	2	Signature is invalid.
STATUS_EMPTY	3	Signature field is empty (not signed).
STATUS_UNKNOWN	4	Cannot be checked (unknown format).

Note: Must be 3 for a new DigSig field.

Element <CertExpired> (empty)

This element specifies the status of the certificate. The expiry date of the embedded public section of the certificate is checked.

0 = OK.
1 = Expired certificate.

Element <Rect/Left>

This element holds the coordinate of the left side of the signature field in pixels relative to the upper left corner of the PDF document page.

Element <Rect/Right>

This element holds the coordinate of the right side of the signature field in pixels relative to the upper left corner of the PDF document page.

Element <Rect/Top>

This element holds the coordinate of the top side of the signature field in pixels relative to the upper left corner of the PDF document page.

Element <Rect/Bottom>

This element holds the coordinate of the bottom side of the signature field in pixels relative to the upper left corner of the PDF document page.

Element <Certificate/Issuer> (empty)

This element holds identifiers for the embedded certificate.
The following standardised identifiers may be included:

CN	The certificate owner's common name
E	The certificate owner's e-mail address
T	The certificate owner's locality
ST	The certificate owner's state of residence
O	The organization to which the certificate owner belongs

OU	The name of the organisational unit to which the certificate owner belongs
C	The certificate owner's country of residence
STREET	The certificate owner's street address
ALL	The certificate owner's complete distinguished name

Element <Certificate/Serial> (empty)

This element holds the serial number for the embedded certificate.

Element <Certificate/PublicKeySize> (empty)

This element holds the RSA encryption key size such as '**2048'**, for example.

Element <Certificate/ValidTo> (empty)

This element holds the end of the validity period for the used certificate.

Element <Certificate/ValidFrom> (empty)

This element holds the beginning of the validity period for the used certificate.

Element <Certificate/CertErrorStatus>

This element contains the error status of the used certificate. The error status is a character string. The string contains semicolon-delimited error descriptions of field *DWORD dwErrorStatus*; in the *CERT_TRUST_STATUS* structure.

Element <Certificate/CertErrorStatusCode>

This element contains the error status code of the used certificate. The error code is the value of the DWORD dwErrorStatus; in the *CERT_TRUST_STATUS* structure.

2.4. XML structure with the input information needed to sign

XML structure and enclosed elements which have to be present and filled as parameter „bstrXMLSignatureData" when calling the methods **SignPdfDocument()**, **SignPdfDocumentMemory()**, **PreparePdfDocumentMemory()** or **PreparePDFDocumentSigningMemory ()** of the component SignPDF3 to create and sign or just sign a DigSig field

Example:

```
<?xml version="1.0" encoding="utf-8" ?>
<SIGNATURE_INFO>
    <Name>Max Mustermann</Name>
    <Reason>Ich habe das Dokument gelesen</Reason>
    <Location>Wien</Location>
    <ContactInfo>signotec GmbH</ContactInfo>
    <TimeStamp>
        <Color>808080</Color>
    </TimeStamp>
    <CustomText>
        <Text>Max Mustermann</Text>
    </CustomText>
    <Rect>
        <Left>115</Left>
        <Right>468</Right>
```

```
            <Top>520</Top>
            <Bottom>646</Bottom>
        </Rect>
        <Signature>
         <Color>808080</Color>
         <Alignment>0</Alignment>
        </Signature>
        <Page>1</Page>
</SIGNATURE_INFO>
```

Element <SIGNATURE_INFO>

The root element of this XML document is <SIGNATURE_INFO>. The following attributes appear beneath this root element:

Element <Name> (optional)

The Name element optionally holds the name of the signer or the automatically determined name for which the certificate has been issued. Standard display – also in Adobe Acrobat® Reader.

Element <Reason> (optional)

The Reason element holds a brief description explaining why a document has been signed. Standard display – also in Adobe Acrobat® Reader.

Element <Location> (optional)

The Location element holds a brief description of the location. Standard display – also in Adobe Acrobat® Reader.

Element <ContactInfo> (optional)

The ContactInfo element holds an optional name or contact information that is automatically determined from the certificate. Standard display – also in Adobe Acrobat® Reader under Certificate display.

Element <Timestamp/Color>

This element holds the colour used for the visible timestamp located at the bottom of the signature. The example holds the RGB value '808080'. This corresponds to a shade of grey.
The colour is defined as a hexadecimal RGB value.

FF0000	= red
00FF00	= green
0000FF	= blue
808080	= grey

Any other colour combination is possible here.

Element <CustomText/Text> (optional)

This element holds optional text in the colour of the timestamp that is embedded and displayed at the bottom of the signature. This could be the name of the signer, for example.

Element <Rect/Left> (optional, ! necessary if the signature field is not yet present or the position of the existing signature field should be changed)

This element holds the coordinate of the left side of the signature field in points (pt) relative to the upper left corner of the PDF document page (see diagram).

Element <Rect/Right> (optional, ! necessary if the signature field is not yet present or the position of the existing signature field should be changed)

This element holds the coordinate of the right side of the signature field in points (pt) relative to the upper left corner of the PDF document page (see diagram).

Element <Rect/Top> (optional, ! necessary if the signature field is not yet present or the position of the existing signature field should be changed)

This element holds the coordinate of the top side of the signature field in points (pt) relative to the upper left corner of the PDF document page (see diagram).

Element <Rect/Bottom> (optional, ! necessary if the signature field is not yet present or the position of the existing signature field should be changed)

This element holds the coordinate of the bottom side of the signature field in points (pt) relative to the upper left corner of the PDF document page (see diagram).

Element <Signature/Color> (optional)

This element holds the colour used for the signature line.
The example holds the RGB value '808080'. This corresponds to a shade of grey.
The colour is defined as a hexadecimal RGB value.

FF0000	= red
00FF00	= green
0000FF	= blue
808080	= grey

Any other colour combination is possible here.

Element <Signature/Alignment> (optional)

This element holds the value of the alignment for the signature inside the signature field, if the signature is smaller then the field.

Possible values:

0 = Right aligned
1 = Left aligned
0 = Centered

Element <Page> (optional, ! necessary if the signature field is not yet present)

This element holds the page number in the PDF on which the signature field is to be generated. The name of the DigSig field must be unique within a document and may not be defined repeatedly on different pages of the document. This is a legal requirement as form fields can, in principle, be defined with the same name on different pages.

The additional elements for **PreparePdfDocumentMemory()**, which when signing and encrypting in the pad (filled) must be passed.

Example:

```
<RSAParams>
<HashType>COMBINATION</HashType>
<ContentLength>8781</ContentLength>
<RSAScheme>PSS</RSAScheme>
<DocAlgorithm>SHA256</DocAlgorithm>
<BioAlgorithm>SHA256</BioAlgorithm>

<RSASignature>IaaotszlPmu6OYSb5g0F6MyapmPeAHEeLIIoNDIy9q+FOhra
Lmzf+/z9D5oU1YXvjoOlLAMgLoxAuk957qVoRer/QMspbG2eM3OGmAo+2p
mPIebpzZg9oBTIsAtk4SYSzVdtcVq0bMgkcad7/MmPt/ETKVd/h5em4cKqG
Xk+NYYaAEIMeMqITFK3/VVVpAXvLoO7VZtdqf1aw/J5V+ONr+xgGmAEuD
mlp4r/dqMyZjCVlbdqYBXhThS9kFQHrNMQB8VOp8VgQqp1Ta7zukQ44nC/
Wwskd3i7+ZfMh7fSsjt2b397WhY/wQ0oPy0YvCBaI6V1B6lc4lnTZld1gtEEw
w==</RSASignature>
</RSAParams
```

Element < RSAParams>

This element contains subelements for signing and encrypting in the pad.

Element < HashType>

This element contains the hash type.

Element < ContentLength>

This element contains the length of the document before the signing process.

Element < RSAScheme>

This element contains the RSA scheme.

Element < DocAlgorithm>

This element contains the document hash algorithm.

Element < BioAlgorithm>

This element contains the biometric hash algorithm

Element < RSASignature>

This element contains the RSA signature.

2.5. Explaination of the coordinate system used for DigSig fields

Diagram illustrating the coordinate system for the signature field using a page in a PDF document whose dimensions are 843*595 points (pt). The origin is always the upper left corner (0,0). All values are specified in points:

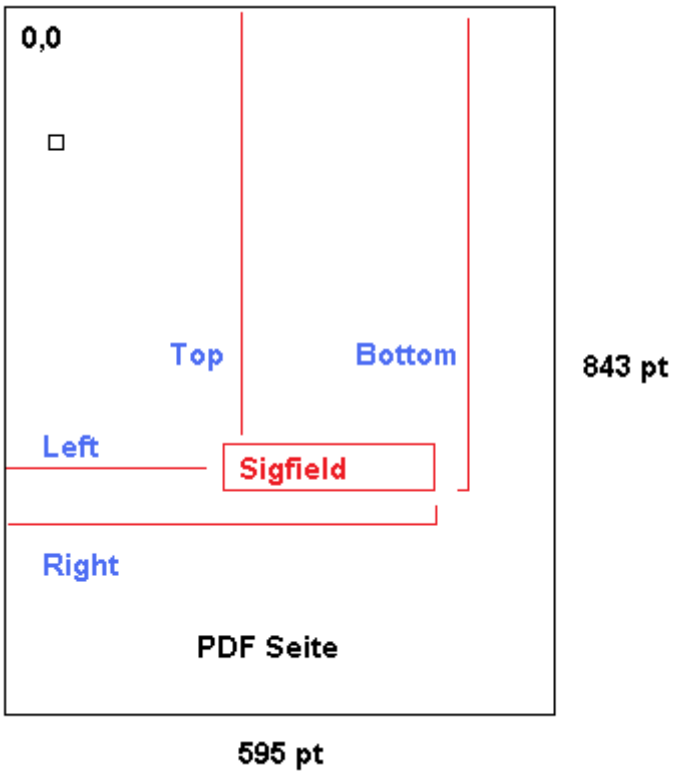

Diagramm of the coordinate system used for DigSig fields