

# VersyPDF API Reference

## Table of Contents

<b>VersyPDF Library</b>	<b>1</b>
<b>PDF Library Level</b>	<b>2</b>
PDF Library Exception Level	3
<b>PDF Document Level</b>	<b>6</b>
Load And Save PDF Documents	6
Appending Of The Pages To PDF	10
<b>PDF Page Level</b>	<b>18</b>
<b>PDF Page Copier Level</b>	<b>24</b>
<b>PDF Image Level</b>	<b>26</b>
<b>Extended Graphic State Level</b>	<b>35</b>
<b>Font Level</b>	<b>45</b>
<b>Canvas Drawing Level</b>	<b>48</b>
Graphic State Operations	48
Path Construction Operations	53
Path Painting Operations	61
Text Operations	64
Other Drawing Operations	69
<b>PDF Acroform Level</b>	<b>73</b>
<b>PDF Outline Level</b>	<b>84</b>
<b>Thread and Bead Level</b>	<b>92</b>
Thread Operations	92

Bead Operations	96
<b>Annotation Level</b>	<b>99</b>
<b>Action Level</b>	<b>105</b>
Goto Action	107
Goto Remote Action	108
Launch Action	108
Hide Action	109
URI Action	111
Thread Action	111
Named Action	112
Import Action	113
Submit Action	114
Reset Form Action	116
Javascript Action	118
<b>CosObject Level</b>	<b>120</b>
Common Cos Object Functions	120
Cos Null Object	123
Cos Boolean Object	124
Cos Number Objects	125
Cos Real Object	126
Cos Integer Object	127
Cos Name Object	129
Cos String Object	131
Cos Array Object	132
Cos Dictionary Object	136
Cos Stream Object	140
<b>Underline Level</b>	<b>142</b>
Color Level	142
Atom Level	143

File Level	146
Stream Level	150
<b>Structs, Records, Enums</b>	<b>158</b>
<b>Index</b>	<b>a</b>

# 1 VersyPDF Library

**What is VersyPDF Library?** VersyPDF – it's library which allows to create or modify immediately PDF documents. You with the help of your application receives some data and then with the help of our library convert this data to text and images in PDF document. Our library will get you free from knowledge of inner PDF structure of document and give you chance to represent you your data without time loss.

## 2 PDF Library Level

We must initialize the library our data will be thread-safe and to initialize the exceptions system.

After work with library is finished we are to get free all the memory which was used by the library.

### Functions

Function
🔗 InitPDFLibrary ( <a href="#">see page 2</a> )
🔗 InitPDFLibraryWithParams ( <a href="#">see page 2</a> )
🔗 DonePDFLibrary ( <a href="#">see page 3</a> )

### Legend

🔗	Method
---	--------

## 2.1 InitPDFLibrary Function

```
LIB_API PDFLibHandle InitPDFLibrary(char * LicenseName, char * LicenseKey);
```

### File

VSLibA.h

### Parameters

Parameters	Description
char * LicenseName	[in] License Name for check license information.
char * LicenseKey	[in] License key for check license information.

### Returns

PDF Library Handle.

### Description

Initialize PDF library for current username and registration key. Prepare and fill in all library structures.

## 2.2 InitPDFLibraryWithParams Function

```
LIB_API PDFLibHandle InitPDFLibraryWithParams(char * LicenseName, char * LicenseKey,  
PDFAllocProc allocproc, PDFReAllocProc reallocproc, PDFFreeProc freeproc, PDFErrorProc  
errorproc, void * opaque);
```

### File

VSLibA.h

### Parameters

Parameters	Description
char * LicenseName	[in] License Name for check license information.
char * LicenseKey	[in] License key for check license information.
PDFAllocProc allocproc	[in] Pointer to a user-supplied memory allocation function.

PDFReAllocProc reallocproc	[in] Pointer to a user-supplied memory reallocation function.
PDFFreeProc freeproc	[in] Pointer to a user-supplied free function.
PDFErrorProc errorproc	[in] Pointer to a user-supplied error-handling function.
void * opaque	[in] Pointer to user data which may be retrieved in error-handling function.

**Returns**

PDF Library Handle.

**Description**

Initialize PDF library for current username and registration key . Prepare and fill in all library structures.

---

## 2.3 DonePDFLibrary Function

```
LIB_API void DonePDFLibrary(PDFLibHandle * lib);
```

**File**

VSLibA.h

**Parameters**

Parameters	Description
PDFLibHandle * lib	[ in ] Handle to created PDF library.

**Returns**

None.

**Description**

Close PDF library.

---

## 2.4 PDF Library Exception Level

**Functions**

Function
VSGetErrorStr ( see page 3)
PPCallException ( see page 4)
PPGetLastError ( see page 4)
PPPopExceptionBuffer ( see page 5)
PPPushExceptionBuffer ( see page 5)

**Legend**

	Method
---	--------

---

### 2.4.1 VSGetErrorStr Function

```
LIB_API ppInt32 VSGetErrorStr(ppUns32 ErrorCode, char * Buffer);
```

**File**

VSError.h

**Parameters**

Parameters	Description
ppUns32 ErrorCode	[ in ] Error Code.
char * Buffer	[ in ] Buffer where will stored string. Buffer size must be >= 255 chars

**Returns**

Size of the string.

**Description**

Returns string value of error code.

---

## 2.4.2 PPCallException Function

```
LIB_API void PPCallException(PDFLibHandle Lib, ppInt32 err);
```

**File**

VSExcept.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[ in ] Current PDF library.
ppInt32 err	[ in ] Error code.

**Returns**

None.

**Description**

Call the exception from error code.

---

## 2.4.3 PPGetLastError Function

```
LIB_API ppInt32 PPGetLastError(PDFLibHandle Lib);
```

**File**

VSExcept.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[ in ] Current PDF library.

**Returns**

Code of last error.

**Description**

Returns code of last error.



---

## 2.4.4 PPPopExceptionBuffer Function

```
LIB_API void PPPopExceptionBuffer(PDFLibHandle Lib);
```

**File**

VSExcept.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[ in ] Current PDF library.

**Returns**

None.

**Description**

Pop exception code from buffer.

---

## 2.4.5 PPPushExceptionBuffer Function

```
LIB_API PDFjmp_buf * PPPushExceptionBuffer(PDFLibHandle Lib);
```

**File**

VSExcept.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[ in ] Current PDF library.

**Returns**

PDF jump buffer.

**Description**

Push exception the buffer.

## 3 PDF Document Level

This level describes function interfaces for working with PDF document. All the functions are divided into three blocks: functions for loading and saving of the documents, functions for page adding and functions of document property.

### 3.1 Load And Save PDF Documents

Document loading functions are used to get handle document object. It may be early created PDF document located in file or in memory or newly created document.

In any case user receives handle to PDF document object and may use all abilities to modify it.

Functions of the document saving are used to store document modification results.

We may store PDF document in file or in memory stream if we modified or created and filled in new document.

It does not depend if we save or not document it must be closed if we loaded or created document. It gets correctly to finish work with PDF documents and release used memory.

#### Functions

Function
PDFDocCreate (see page 6)
PDFDocLoadFromFile (see page 7)
PDFDocLoadFromBuffer (see page 7)
PDFDocLoadFromStream (see page 8)
PDFDocSaveToFile (see page 8)
PDFDocSaveToStream (see page 8)
PDFDocSaveToBuffer (see page 9)
PDFDocClose (see page 9)

#### Legend

	Method
---	--------

#### 3.1.1 PDFDocCreate Function

```
LIB_API PDFDocHandle PDFDocCreate(PDFLibHandle Lib);
```

##### File

VSDocA.h

##### Parameters

Parameters	Description
PDFLibHandle Lib	[in] Current loaded PDF Library.

##### Returns

PDF Document Handle.

**Description**

Create New PDF Document.

**See Also**

PDFDocClose ([↗](#) see page 9).

---

## 3.1.2 PDFDocLoadFromFile Function

```
LIB_API PDFDocHandle PDFDocLoadFromFile(PDFLibHandle Lib, char * FileName);
```

**File**

VSDocA.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[in] PDF Library Handle.
char * FileName	[in] Name of input file, text string.

**Returns**

PDF Document Handle.

**Description**

Load PDF Document from file.

**See Also**

PDFDocSaveToFile ([↗](#) see page 8).

---

## 3.1.3 PDFDocLoadFromBuffer Function

```
LIB_API PDFDocHandle PDFDocLoadFromBuffer(PDFLibHandle Lib, void * Buffer, ppInt32 Length);
```

**File**

VSDocA.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[in] PDF Library Handle.
void * Buffer	[in] Pointer on PDF document beginning.
ppInt32 Length	[in] Length of buffer in bytes.

**Returns**

PDF Document Handle.

**Description**

Load PDF Document from Memory buffer.

**See Also**

PDFDocSaveToBuffer ([↗](#) see page 9).

---

## 3.1.4 PDFDocLoadFromStream Function

```
LIB_API PDFDocHandle PDFDocLoadFromStream(PDFLibHandle Lib, PDFStreamHandle Stream);
```

### File

VSDocA.h

### Parameters

Parameters	Description
PDFLibHandle Lib	[in] PDF Library Handle.
PDFStreamHandle Stream	[in] PDF Stream Handle.

### Returns

PDF Document Handle.

### Description

Load PDF Document from memory stream or file stream, see PDF Streams.

### See Also

PDFDocSaveToStream ([↗](#) see page 8), PDFStreamHandle, PDFDocLoadFromFile ([↗](#) see page 7), PDFDocLoadFromBuffer ([↗](#) see page 7).

---

## 3.1.5 PDFDocSaveToFile Function

```
LIB_API void PDFDocSaveToFile(PDFDocHandle Doc, char * FileName);
```

### File

VSDocA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
char * FileName	[in] Name of output file, text string.

### Returns

None.

### Description

Save PDF Document in file.

### See Also

PDFDocLoadFromFile ([↗](#) see page 7).

---

## 3.1.6 PDFDocSaveToStream Function

```
LIB_API void PDFDocSaveToStream(PDFDocHandle Doc, PDFStreamHandle Stream);
```

### File

VSDocA.h

---

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
PDFStreamHandle Stream	[out] PDF Stream Handle.

**Returns**

None.

**Description**

Save PDF Document in memory stream or file stream, see PDF Streams.

**See Also**

PDFDocLoadFromStream (🔗 see page 8), PDFStreamHandle, PDFDocSaveToBuffer (🔗 see page 9), PDFDocSaveToFile (🔗 see page 8).

---

## 3.1.7 PDFDocSaveToBuffer Function

```
LIB_API void * PDFDocSaveToBuffer(PDFDocHandle Doc, ppInt32 * Size);
```

**File**

VSDocA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppInt32 * Size	[out] Size of buffer.

**Returns**

Pointer on PDF document beginning.

**Description**

Save PDF Document in memory.

**See Also**

PDFDocLoadFromBuffer (🔗 see page 7).

---

## 3.1.8 PDFDocClose Function

```
LIB_API void PDFDocClose(PDFDocHandle Doc);
```

**File**

VSDocA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.

**Returns**

None.

Description

Close PDF Document and free all structures.

See Also

PDFDocCreate (🔗 see page 6).

# 3.2 Appending Of The Pages To PDF

Functions for working with pages are described in this block. It's possible to find out count of the pages in opened document and append pages to document with the help of these functions.

Functions

Function
🔗 PDFDocAppendPage (🔗 see page 10)
🔗 PDFDocAppendPage2 (🔗 see page 10)
🔗 PDFDocGetPageCount (🔗 see page 11)

Legend

🔗	Method
---	--------

## 3.2.1 PDFDocAppendPage Function

```
LIB_API ppInt32 PDFDocAppendPage(PDFDocHandle Doc, ppReal Width, ppReal Height);
```

File

VSDocA.h

Parameters

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppReal Width	[in] Width of Page.
ppReal Height	[in] Height of Page.

Returns

Index of New Page in Document.

Description

Add Page in PDF Document.

See Also

PDFDocAppendPage2 (🔗 see page 10).

## 3.2.2 PDFDocAppendPage2 Function

```
LIB_API ppInt32 PDFDocAppendPage2(PDFDocHandle Doc, TPDFPageSize Size, TPDFPageOrientation Orientation);
```

**File**

VSDocA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
TPDFPageSize Size	[in] Type of Page Size, see TPDFPageSize.
TPDFPageOrientation Orientation	[in] Orientation of Page, see TPDFPageOrientation.

**Returns**

Index of New Page in Document.

**Description**

Add Page in PDF Document.

**See Also**

PDFDocAppendPage (see page 10), TPDFPageSize, TPDFPageOrientation.

## 3.2.3 PDFDocGetPageCount Function

```
LIB_API ppInt32 PDFDocGetPageCount(PDFDocHandle Doc);
```

**File**

VSDocA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.

**Returns**

Page Count of PDF Document.

**Description**

Get Page Count of PDF Document.

## 3.3 Document Properties

Property functions are used to set or receive document characteristics such as Linearization, Packing, Security, etc. It allows to change way of document storage and to control work with it.

**Functions**

Function
PDFDocIsCrypteed (see page 12)
PDFDocCheckPassword (see page 12)
PDFDocGetPermission (see page 13)
PDFDocSetUsedDC (see page 13)
PDFDocSetJpegImageQuality (see page 13)
PDFDocSetLinearized (see page 14)
PDFDocSetPacked (see page 14)

PDFDocSetRemoveUnUsed (see page 15)
PDFDocSetAutoLaunch (see page 15)
PDFDocSetEMFBWImagesAsJBIG2 (see page 15)
PDFDocSetEMFColorImagesAsJpeg (see page 16)
PDFDocGetInfo (see page 16)
PDFDocGetVersion (see page 17)
PDFDocSetInfo (see page 17)
PDFDocSetVersion (see page 17)

#### Legend

	Method
---	--------

## 3.3.1 PDFDocIsCrypted Function

```
LIB_API ppBool PDFDocIsCrypted(PDFDocHandle Doc);
```

#### File

VSDocA.h

#### Parameters

Parameters	Description
PDFDocHandle Doc	[in] Checking PDF Document Handle.

#### Returns

Boolean value : true - if document is crypted, false is not crypted.

#### Description

Checking the PDF Document on Security.

#### See Also

PDFDocCheckPassword (see page 12), PDFDocSetSecurity.

## 3.3.2 PDFDocCheckPassword Function

```
LIB_API TKeyValidType PDFDocCheckPassword(PDFDocHandle Doc, char * Password);
```

#### File

VSDocA.h

#### Parameters

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
char * Password	[in] Password text string.

#### Returns

Password Validity Type. If kvtnInvalid then Password is invalid.

#### Description

Checking the PDF Document on Validity of Password.



**See Also**

PDFDocsCrypted ([↗](#) see page 12), PDFDocSetSecurity, TKeyValidType.

---

## 3.3.3 PDFDocGetPermission Function

```
LIB_API ppInt32 PDFDocGetPermission(PDFDocHandle Doc);
```

**File**

VSDocA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] Checking PDF Document Handle.

**Returns**

Document Permission's Flags.

**Description**

Get Document Permission's Flags.

**See Also**

PDFDocSetSecurity.

---

## 3.3.4 PDFDocSetUsedDC Function

```
LIB_API void PDFDocSetUsedDC(PDFDocHandle Doc, HDC DC);
```

**File**

VSDocA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
HDC DC	[in] Handle of Device Context.

**Returns**

None.

**Description**

Set hDC concerning which EMF images will be parsed.

---

## 3.3.5 PDFDocSetJpegImageQuality Function

```
LIB_API void PDFDocSetJpegImageQuality(PDFDocHandle Doc, ppInt32 Quality);
```

**File**

VSDocA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppInt32 Quality	[in] Value from 0 to 100: 0 - worst quality, smallest size. 100 - best quality, biggest size.

**Returns**

None.

**Description**

Set "Jpeg Images Quality" for jpeg images stored in PDF Document.

---

## 3.3.6 PDFDocSetLinearized Function

```
LIB_API void PDFDocSetLinearized(PDFDocHandle Doc, ppBool Linearized);
```

**File**

VSDocA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppBool Linearized	[in] if true use linearized save method. If false non linearized method.

**Returns**

None.

**Description**

Set whether PDF document is stored as linearized document or not.

---

## 3.3.7 PDFDocSetPacked Function

```
LIB_API void PDFDocSetPacked(PDFDocHandle Doc, ppBool Packed);
```

**File**

VSDocA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppBool Packed	[in] if true use packed save method. If false don't use it.

**Returns**

None.

**Description**

Set whether PDF document is stored as packed document or not.

---

### 3.3.8 PDFDocSetRemoveUnused Function

```
LIB_API void PDFDocSetRemoveUnused(PDFDocHandle Doc, ppBool Remove);
```

**File**

VSDocA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppBool Remove	[in] If true remove unused ( unlink ) objects. If false don't remove.

**Returns**

None.

**Description**

Remove all unused objects from PDF Document before save it.

---

### 3.3.9 PDFDocSetAutoLaunch Function

```
LIB_API void PDFDocSetAutoLaunch(PDFDocHandle Doc, ppBool AutoLaunch);
```

**File**

VSDocA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppBool AutoLaunch	[in] If true launch PDF document, If false don't.

**Returns**

None.

**Description**

Set AutoLaunch option for PDF Document, it launches after saving (only Windows Platform).

---

### 3.3.10 PDFDocSetEMFBWImagesAsJBIG2 Function

```
LIB_API void PDFDocSetEMFBWImagesAsJBIG2(PDFDocHandle Doc, ppBool AsJBIG2);
```

**File**

VSDocA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.

ppBool AsJBIG2	[in] If value has set to true, all black/white images stored in the EMF file will be saved with JBIG2 compression. If value has set to false, all images stored in the EMF file will be saved with CCITT compression.
----------------	--

**Returns**

None.

**Description**

Set "Emf BW Images as JBIG2" option for PDF Document.

---

## 3.3.11 PDFDocSetEMFColorImagesAsJpeg Function

```
LIB_API void PDFDocSetEMFColorImagesAsJpeg(PDFDocHandle Doc, ppBool EmfAsJpeg);
```

**File**

VSDocA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppBool EmfAsJpeg	[in] If value has set to true, all color images stored in the EMF file will be saved as JPEG images. If value has set to false, all color images stored in the EMF file will be saved with flate compression.

**Returns**

None.

**Description**

Set "Emf color Images as jpeg" option for PDF Document.

---

## 3.3.12 PDFDocGetInfo Function

```
LIB_API char * PDFDocGetInfo(PDFDocHandle Doc, TPDFInformation Info, ppUns32 * StrLen);
```

**File**

VSDocA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
TPDFInformation Info	[in] Type of Description Information.

**Returns**

Handle of Information Object.

**Description**

Get Information from Document Description.

**See Also**

PDFDocSetInfo ([🔗](#) see page 17), TPDFInformation.

---

## 3.3.13 PDFDocGetVersion Function

```
LIB_API TPDFVersion PDFDocGetVersion(PDFDocHandle Doc);
```

**File**

VSDocA.h

**Description**

This is function PDFDocGetVersion.

---

## 3.3.14 PDFDocSetInfo Function

```
LIB_API void PDFDocSetInfo(PDFDocHandle Doc, TPDFInformation Info, char * Str, ppUns32 StrLen);
```

**File**

VSDocA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
TPDFInformation Info	[in] Type of Description Information.
Value	[in] Handle of Information Object.

**Returns**

None.

**Description**

Save information in Document description.

**See Also**

PDFDocGetInfo ([↗](#) see page 16), TPDFInformation.

---

## 3.3.15 PDFDocSetVersion Function

```
LIB_API void PDFDocSetVersion(PDFDocHandle Doc, TPDFVersion Version);
```

**File**

VSDocA.h

**Description**

This is function PDFDocSetVersion.

## 4 PDF Page Level

Functions for working with pages are used to identify page characteristics, to change these characteristics and to create page contents. So it's possible to change page sizes, its orientation, to create and remove contents and to find out contents count. Content filling is realized by PaintBox creating for page or for defined content and painting with the help of the PaintBox working functions.

### Functions

Function
⇒ PDFPageAddContent (🔗 see page 18)
⇒ PDFPageCreatePaintBox (🔗 see page 19)
⇒ PDFPageCreatePaintBoxFromContent (🔗 see page 19)
⇒ PDFPageGetBox (🔗 see page 20)
⇒ PDFPageGetContentCount (🔗 see page 20)
⇒ PDFPageGetCosObject (🔗 see page 20)
⇒ PDFPageGetRotateAngle (🔗 see page 21)
⇒ PDFPageInsertContent (🔗 see page 21)
⇒ PDFPageRemoveContent (🔗 see page 22)
⇒ PDFPageSetBox (🔗 see page 22)
⇒ PDFPageSetRotateAngle (🔗 see page 23)

### Legend

⇒	Method
---	--------

## 4.1 PDFPageAddContent Function

```
LIB_API ppUns32 PDFPageAddContent(PDFDocHandle Doc, ppUns32 Page);
```

### File

VSPageA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppUns32 Page	[in] Page Index in PDF Document.

### Returns

Page Content Streams Count..

### Description

Add void Content to Page Content Streams.

### See Also

PDFPageGetContentCount (🔗 see page 20)

## 4.2 PDFPageCreatePaintBox Function

```
LIB_API PBXHandle PDFPageCreatePaintBox(PDFDocHandle Doc, ppUns32 Page, ppUns32 Resolution);
```

### File

VSPageA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppUns32 Page	[in] Page Index in PDF Document.
ppUns32 Resolution	[in] Points per inch (dpi).

### Returns

Paint Box Handle.

### Description

Create Paint Box for Last Page Content with defined resolution.

### See Also

PDFPageCreatePaintBoxFromContent ([↗](#) see page 19)

---

## 4.3 PDFPageCreatePaintBoxFromContent Function

```
LIB_API PBXHandle PDFPageCreatePaintBoxFromContent(PDFDocHandle Doc, ppUns32 Page, ppUns32 Index, ppUns32 Resolution);
```

### File

VSPageA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppUns32 Page	[in] Page Index in PDF Document.
ppUns32 Index	[in] Content's Index in Page Content Streams.
ppUns32 Resolution	[in] Points per inch (dpi).

### Returns

Paint Box Handle.

### Description

Create Paint Box for Page Content according to index with defined resolution.

### See Also

PDFPageCreatePaintBox ([↗](#) see page 19)

---

## 4.4 PDFPageGetBox Function

```
LIB_API TPDFRect PDFPageGetBox(PDFDocHandle Doc, ppUns32 Page, TPDFPageBoxType Type);
```

### File

VSPageA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppUns32 Page	[in] Page Index in PDF Document.
TPDFPageBoxType Type	[in] PageBox Type for request.

### Returns

PageBox Typed Rectangle.

### Description

Get Typed Rectangle PageBox.

### See Also

PDFPageSetBox (🔗 see page 22)

---

## 4.5 PDFPageGetContentCount Function

```
LIB_API ppUns32 PDFPageGetContentCount(PDFDocHandle Doc, ppUns32 Page);
```

### File

VSPageA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppUns32 Page	[in] Page Index in PDF Document.

### Returns

Page Content Streams Count.

### Description

Get Page Content Streams Count.

---

## 4.6 PDFPageGetCosObject Function

```
LIB_API PDFCosHandle PDFPageGetCosObject(PDFDocHandle Doc, ppUns32 Page);
```

### File

VSPageA.h



**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppUns32 Page	[in] Page Index in PDF Document.

**Returns**

Page Object Handle.

**Description**

Get Page Object Handle.

---

## 4.7 PDFPageGetRotateAngle Function

```
LIB_API TPDFPageRotateAngle PDFPageGetRotateAngle(PDFDocHandle Doc, ppUns32 Page);
```

**File**

VSPageA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppUns32 Page	[in] Page Index in PDF Document.

**Returns**

Page Rotation Angle by which the page displayed or printed should be rotated clockwise.

**Description**

Get Rotation Angle of Page.

**See Also**

PDFPageSetRotateAngle ([↗](#) see page 23)

---

## 4.8 PDFPageInsertContent Function

```
LIB_API ppUns32 PDFPageInsertContent(PDFDocHandle Doc, ppUns32 Page, ppUns32 Index);
```

**File**

VSPageA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppUns32 Page	[in] Page Index in PDF Document.
ppUns32 Index	[in] Content Index for inserting.

**Returns**

Index of void Page Content in Content Streams.

**Description**

Insert void Content to Page Content Streams in Indexed Site.

**See Also**

PDFPageRemoveContent (🔗 see page 22)

---

## 4.9 PDFPageRemoveContent Function

```
LIB_API void PDFPageRemoveContent(PDFDocHandle Doc, ppUns32 Page, ppUns32 Index);
```

**File**

VSPageA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppUns32 Page	[in] Page Index in PDF Document.
ppUns32 Index	[in] Content Index for deleting.

**Returns**

None.

**Description**

Remove Content from Page Content Streams according to index.

**See Also**

PDFPageInsertContent (🔗 see page 21)

---

## 4.10 PDFPageSetBox Function

```
LIB_API void PDFPageSetBox(PDFDocHandle Doc, ppUns32 Page, TPDFPageBoxType Type, TPDFRect Rect);
```

**File**

VSPageA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppUns32 Page	[in] Page Index in PDF Document.
TPDFPageBoxType Type	[in] PageBox Type for setting.
TPDFRect Rect	[in] PageBox Rectangle.

**Returns**

None.

**Description**

Set Typed Rectangle PageBox.

**See Also**

PDFPageGetBox ([📄](#) see page 20)

---

## 4.11 PDFPageSetRotateAngle Function

```
LIB_API void PDFPageSetRotateAngle(PDFDocHandle Doc, ppUns32 Page, TPDFPageRotateAngle Rotate);
```

**File**

VSPageA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppUns32 Page	[in] Page Index in PDF Document.
TPDFPageRotateAngle Rotate	[in] Page Rotation Angle by which the page displayed or printed should be rotated clockwise

**Returns**

None.

**Description**

Set Rotation Angle of Page.

**See Also**

PDFPageGetRotateAngle ([📄](#) see page 21)

## 5 PDF Page Copier Level

Selection and copying page of one document to another is used with help of the `TPDFDocumentConnection` structure and its functions. Copying is executed in four steps:

1. Links creating between documents
2. Pages selection from source document.
3. Pages copying from source to destination document.
4. Links removing between documents.

### Functions

Function
⇒ <code>PDFCopyPagesToDestinationDocument</code> ( <a href="#">see page 24</a> )
⇒ <code>PDFSelectPageFromSourceDocument</code> ( <a href="#">see page 25</a> )
⇒ <code>PDFFreeDocumentConnection</code> ( <a href="#">see page 25</a> )

### Legend

⇒	Method
---	--------

## 5.1 PDFCopyPagesToDestinationDocument Function

```
LIB_API void PDFCopyPagesToDestinationDocument (PPDFDocumentConnection DocumentConnection);
```

### File

VSPagesA.h

### Parameters

Parameters	Description
<code>PPDFDocumentConnection DocumentConnection</code>	[out] pointer to <code>PPDFDocumentConnection</code> structure.

### Returns

None.

### Description

Copy Pages from source to destination document and reset page numbers array, but document connection remains linked with documents.

### See Also

`TPDFDocumentConnection`

---

## 5.2 PDFSelectPageFromSourceDocument Function

```
LIB_API void PDFSelectPageFromSourceDocument(PPDFDocumentConnection DocumentConnection,  
ppUns32 PageIndex);
```

### File

VSPagesA.h

### Parameters

Parameters	Description
PPDFDocumentConnection DocumentConnection	[out] pointer to PPDFDocumentConnection structure
ppUns32 PageIndex	[in] integer index of page from Source Document, as from 0

### Returns

None.

### Description

Add page index ( number as from 0 ) to page numbers array in DocumentConnection structure.

### See Also

TPPDFDocumentConnection

---

## 5.3 PDFFreeDocumentConnection Function

```
LIB_API void PDFFreeDocumentConnection(PPDFDocumentConnection DocumentConnection);
```

### File

VSPagesA.h

### Parameters

Parameters	Description
PPDFDocumentConnection DocumentConnection	[out] pointer to PPDFDocumentConnection structure.

### Returns

None.

### Description

Free page numbers array and remove document connection from DocumentConnection structure.

### See Also

TPPDFDocumentConnection

## 6 PDF Image Level

A sampled image (or just image for short) is a rectangular array of sample values, each representing a color. The image may approximate the appearance of some natural scene obtained through an input scanner or a video camera, or it may be generated synthetically.

An image is defined by a sequence of samples obtained by scanning the image array in row or column order.

It's possible to load images from Jpeg, PNG, Tiff files and from windows bitmap handle in our library.

### Functions

Function
⇒ <a href="#">PDFImageAppendToDoc</a> ( <a href="#">see page 26</a> )
⇒ <a href="#">PDFImageAppendToDocAsMask</a> ( <a href="#">see page 27</a> )
⇒ <a href="#">PDFImageAppendToDocFromFile</a> ( <a href="#">see page 27</a> )
⇒ <a href="#">PDFImageAppendToDocFromStream</a> ( <a href="#">see page 28</a> )
⇒ <a href="#">PDFImageAppendToDocWithMask</a> ( <a href="#">see page 28</a> )
⇒ <a href="#">PDFImageFree</a> ( <a href="#">see page 28</a> )
⇒ <a href="#">PDFImageGetColorDevice</a> ( <a href="#">see page 29</a> )
⇒ <a href="#">PDFImageGetDepth</a> ( <a href="#">see page 29</a> )
⇒ <a href="#">PDFImageGetHeight</a> ( <a href="#">see page 30</a> )
⇒ <a href="#">PDFImageGetScanLine</a> ( <a href="#">see page 30</a> )
⇒ <a href="#">PDFImageGetTIFFCountFromFile</a> ( <a href="#">see page 30</a> )
⇒ <a href="#">PDFImageGetTIFFCountFromStream</a> ( <a href="#">see page 31</a> )
⇒ <a href="#">PDFImageGetWidth</a> ( <a href="#">see page 31</a> )
⇒ <a href="#">PDFImageLoadFromFile</a> ( <a href="#">see page 31</a> )
⇒ <a href="#">PDFImageLoadFromStream</a> ( <a href="#">see page 32</a> )
⇒ <a href="#">PDFImageAppendToDocFromBuffer</a> ( <a href="#">see page 32</a> )
⇒ <a href="#">PDFImageCreate</a> ( <a href="#">see page 33</a> )
⇒ <a href="#">PDFImageGetTIFFCountFromBuffer</a> ( <a href="#">see page 33</a> )
⇒ <a href="#">PDFImageLoadFromBuffer</a> ( <a href="#">see page 34</a> )
⇒ <a href="#">PDFImageLoadFromHandle</a> ( <a href="#">see page 34</a> )

### Legend

⇒	Method
---	--------

## 6.1 PDFImageAppendToDoc Function

```
LIB_API ppUns32 PDFImageAppendToDoc(PDFImageHandle Image, PDFDocHandle Doc,
TIFFCompressionType CompressionType);
```

### File

VSImageA.h

### Parameters

Parameters	Description
PDFImageHandle Image	[ in ] PDF Image handle.
PDFDocHandle Doc	[ in ] PDF document handle.

TImageCompressionType CompressionType	[ in ] Image compression type.
---------------------------------------	--------------------------------

**Returns**

Index of the image in PDF document.

**Description**

Appends image to document with specified compression.

## 6.2 PDFImageAppendToDocAsMask Function

```
LIB_API ppUns32 PDFImageAppendToDocAsMask(PDFImageHandle Image, PDFDocHandle Doc,
TImageCompressionType CompressionType);
```

**File**

VSImageA.h

**Parameters**

Parameters	Description
PDFImageHandle Image	[ in ] PDF Image handle.
PDFDocHandle Doc	[ in ] PDF document handle.
TImageCompressionType CompressionType	[ in ] Image compression type.

**Returns**

Index of the image in PDF document.

**Description**

Appends b/w image to document with specified compression as transparent mask.

## 6.3 PDFImageAppendToDocFromFile Function

```
LIB_API ppUns32 PDFImageAppendToDocFromFile(PDFDocHandle Doc, char * FileName, ppUns32
Index, TImageCompressionType CompressionType);
```

**File**

VSImageA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in ] PDF document handle.
char * FileName	[ in ] Filename of file where stored image
ppUns32 Index	[ in ] Index of the image if file ( Used only for Tiff files )
TImageCompressionType CompressionType	[ in ] Image compression type.

**Returns**

Index of the image in PDF document.

**Description**

Appends image to document with specified compression from file.

---

## 6.4 PDFImageAppendToDocFromStream Function

```
LIB_API ppUns32 PDFImageAppendToDocFromStream(PDFDocHandle Doc, PDFStreamHandle AStream,  
ppUns32 Index, TImageCompressionType CompressionType);
```

### File

VSIImageA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in ] PDF document handle.
PDFStreamHandle AStream	[ in ] Stream where stored image
ppUns32 Index	[ in ] Index of the image if file ( Used only for Tiff files )
TImageCompressionType CompressionType	[ in ] Image compression type.

### Returns

Index of the image in PDF document.

### Description

Appends image to document with specified compression from stream.

---

## 6.5 PDFImageAppendToDocWithMask Function

```
LIB_API ppUns32 PDFImageAppendToDocWithMask(PDFImageHandle Image, PDFDocHandle Doc,  
TImageCompressionType CompressionType, ppUns32 MaskImageIndex);
```

### File

VSIImageA.h

### Parameters

Parameters	Description
PDFImageHandle Image	[ in ] PDF Image handle.
PDFDocHandle Doc	[ in ] PDF document handle.
TImageCompressionType CompressionType	[ in ] Image compression type.

### Returns

Index of the image in PDF document.

### Description

Appends image to document with specified compression with transparent mask.

---

## 6.6 PDFImageFree Function

```
LIB_API void PDFImageFree(PDFImageHandle Image);
```



**File**

VSIImageA.h

**Parameters**

Parameters	Description
PDFImageHandle Image	[ in ] PDF Image handle.

**Returns**

None.

**Description**

Free PDF image.

---

## 6.7 PDFImageGetColorDevice Function

```
LIB_API TPDFColorDevice PDFImageGetColorDevice(PDFImageHandle Image);
```

**File**

VSIImageA.h

**Parameters**

Parameters	Description
PDFImageHandle Image	[ in ] PDF Image handle.

**Returns**

Color space.

**Description**

Gets the color space in which image samples are specified.

---

## 6.8 PDFImageGetDepth Function

```
LIB_API ppComponentDepth PDFImageGetDepth(PDFImageHandle Image);
```

**File**

VSIImageA.h

**Parameters**

Parameters	Description
PDFImageHandle Image	[ in ] PDF Image handle.

**Returns**

The bit count.

**Description**

Gets the bit count per component of the image.

---

## 6.9 PDFImageGetHeight Function

```
LIB_API ppUns32 PDFImageGetHeight(PDFImageHandle Image);
```

### File

VSIImageA.h

### Parameters

Parameters	Description
PDFImageHandle Image	[ in ] PDF Image handle.

### Returns

Height in samples of the bitmap.

### Description

Gets the height of the image.

---

## 6.10 PDFImageGetScanLine Function

```
LIB_API void * PDFImageGetScanLine(PDFImageHandle Image, ppUns32 ScanLineIndex);
```

### File

VSIImageA.h

### Parameters

Parameters	Description
PDFImageHandle Image	[ in ] PDF Image handle.
ppUns32 ScanLineIndex	[ in ] Index of line of the pixels.

### Returns

Pointer to line of the samples.

### Description

Provides indexed access to each line of samples.

---

## 6.11 PDFImageGetTIFFCountFromFile Function

```
LIB_API ppUns32 PDFImageGetTIFFCountFromFile(PDFLibHandle Lib, char * FileName);
```

### File

VSIImageA.h

### Parameters

Parameters	Description
PDFLibHandle Lib	[ in ] PDF Library handle.
char * FileName	[ in ] Filename of the file where stored image.

**Returns**

Count of the images.

**Description**

Gets count of the images in the tiff file.

---

## 6.12 PDFImageGetTIFFCountFromStream Function

```
LIB_API ppUns32 PDFImageGetTIFFCountFromStream(PDFLibHandle Lib, PDFStreamHandle AStream);
```

**File**

VSIImageA.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[ in ] PDF Library handle.
PDFStreamHandle AStream	[ in ] Stream handle.

**Returns**

Count of the images.

**Description**

Gets count of the images in the tiff file opened with PDF Stream.

---

## 6.13 PDFImageGetWidth Function

```
LIB_API ppUns32 PDFImageGetWidth(PDFImageHandle Image);
```

**File**

VSIImageA.h

**Parameters**

Parameters	Description
PDFImageHandle Image	[ in ] PDF Image handle.

**Returns**

Width in samples of the bitmap.

**Description**

Gets the width of the image.

---

## 6.14 PDFImageLoadFromFile Function

```
LIB_API PDFImageHandle PDFImageLoadFromFile(PDFLibHandle Lib, char * FileName, ppUns32
```

```
Index);
```

**File**

VSIImageA.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[ in ] PDF Library handle.
char * FileName	[ in ] Filename of the file where stored image
ppUns32 Index	[ in ] Index of the image if file ( Used only for Tiff files )

**Returns**

PDF Image handle.

**Description**

Create PDF image from image file.

---

## 6.15 PDFImageLoadFromStream Function

```
LIB_API PDFImageHandle PDFImageLoadFromStream(PDFLibHandle Lib, PDFStreamHandle AStream,  
ppUns32 Index);
```

**File**

VSIImageA.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[ in ] PDF Library handle.
PDFStreamHandle AStream	[ in ] Stream handle.
ppUns32 Index	[ in ] Index of the image if file ( Used only for Tiff files )

**Returns**

PDF Image handle.

**Description**

Create PDF image from image file opened with PDF Stream.

---

## 6.16 PDFImageAppendToDocFromBuffer Function

```
LIB_API ppUns32 PDFImageAppendToDocFromBuffer(PDFDocHandle Doc, void * Buffer, ppUns32  
BufferSize, ppUns32 Index, TImageCompressionType CompressionType);
```

**File**

VSIImageA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in ] PDF document handle.

void * Buffer	[ in ] Storage location for data.
ppUns32 BufferSize	[ in ] Size of the storage.
ppUns32 Index	[ in ] Index of the image if file ( Used only for Tiff files )
TImageCompressionType CompressionType	[ in ] Image compression type.

**Returns**

Index of the image in PDF document.

**Description**

Appends image to document with specified compression from memory buffer.

---

## 6.17 PDFImageCreate Function

```
LIB_API PDFImageHandle PDFImageCreate(PDFLibHandle Lib, ppUns32 Width, ppUns32 Height, ppComponentDepth Depth, TPDFColorDevice Device);
```

**File**

VSImageA.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[ in ] PDF Library handle.
ppUns32 Width	[ in ] Specifies the width in pixels of the image.
ppUns32 Height	[ in ] Specifies the height in pixels of the image.
ppComponentDepth Depth	[ in ] Indicates the bit count per component of the image.
TPDFColorDevice Device	[ in ] Indicates the color space in which image samples are specified.

**Returns**

PDF Image handle.

**Description**

Create PDF image.

---

## 6.18 PDFImageGetTIFFCountFromBuffer Function

```
LIB_API ppUns32 PDFImageGetTIFFCountFromBuffer(PDFLibHandle Lib, void * Buffer, ppUns32 BufferSize);
```

**File**

VSImageA.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[ in ] PDF Library handle.
void * Buffer	[ in ] Storage location for data.
ppUns32 BufferSize	[ in ] Size of the storage.

**Returns**

Count of the images.

**Description**

Gets count of the images in the tiff file stored in memory buffer.

---

## 6.19 PDFImageLoadFromBuffer Function

```
LIB_API PDFImageHandle PDFImageLoadFromBuffer(PDFLibHandle Lib, void * Buffer, ppUns32 BufferSize, ppUns32 Index);
```

**File**

VSIImageA.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[ in ] PDF Library handle.
void * Buffer	[ in ] Storage location for data.
ppUns32 BufferSize	[ in ] Size of the storage.
ppUns32 Index	[ in ] Index of the image if file ( Used only for Tiff files )

**Returns**

PDF Image handle.

**Description**

Create PDF image from image file stored in memory buffer.

---

## 6.20 PDFImageLoadFromHandle Function

```
LIB_API PDFImageHandle PDFImageLoadFromHandle(PDFLibHandle Lib, HBITMAP Handle);
```

**File**

VSIImageA.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[ in ] PDF Library handle.
HBITMAP Handle	[ in ] Bitmap handle.

**Returns**

PDF Image handle.

**Description**

Create image from bitmap handle.

## 7 Extended Graphic State Level

Extended State - are state to work with graphics and text on page content. It's comfortable to use Extended State if it is needed to used defined styles in painting. This state of graphical settings is stored within document. Changing of this settings is executed with function of this level. Extended State changing influences all the objects which use it.

### Functions

Function
PDFExtGraphicStateNew (see page 35)
PDFExtGraphicStateSetAlphaFill (see page 36)
PDFExtGraphicStateSetAlphalsShape (see page 36)
PDFExtGraphicStateSetAlphaStroke (see page 37)
PDFExtGraphicStateSetBlendMode (see page 37)
PDFExtGraphicStateSetCTM (see page 37)
PDFExtGraphicStateSetDashPattern (see page 38)
PDFExtGraphicStateSetFlatness (see page 38)
PDFExtGraphicStateSetLineCap (see page 39)
PDFExtGraphicStateSetLineJoin (see page 39)
PDFExtGraphicStateSetLineWidth (see page 40)
PDFExtGraphicStateSetMitterLimit (see page 40)
PDFExtGraphicStateSetOverprintFill (see page 41)
PDFExtGraphicStateSetOverprintMode (see page 41)
PDFExtGraphicStateSetOverprintStroke (see page 42)
PDFExtGraphicStateSetRenderingIntent (see page 42)
PDFExtGraphicStateSetSmoothness (see page 43)
PDFExtGraphicStateSetStrokeAdjustment (see page 43)
PDFExtGraphicStateSetTextKnockout (see page 44)

### Legend

	Method
---	--------

## 7.1 PDFExtGraphicStateNew Function

```
LIB_API ppUns32 PDFExtGraphicStateNew(PDFDocHandle Doc);
```

### File

VSGStateA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in ] Current PDF document.

### Returns

Graphic state index.

### Description

Creates graphic state in document.

## 7.2 PDFExtGraphicStateSetAlphaFill Function

```
LIB_API void PDFExtGraphicStateSetAlphaFill(PDFDocHandle Doc, ppUns32 GState, ppReal AlphaFill);
```

### File

VSGStateA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
ppReal AlphaFill	[ in ] Alpha fill.

### Returns

None.

### Description

Setting alpha fill for current graphic state.

## 7.3 PDFExtGraphicStateSetAlphaIsShape Function

```
LIB_API void PDFExtGraphicStateSetAlphaIsShape(PDFDocHandle Doc, ppUns32 GState, ppBool AlphaIsShape);
```

### File

VSGStateA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
ppBool AlphaIsShape	[ in ] Alpha is shape.

### Returns

None.

### Description

Setting "alpha is shape" for current graphic state.



---

## 7.4 PDFExtGraphicStateSetAlphaStroke Function

```
LIB_API void PDFExtGraphicStateSetAlphaStroke(PDFDocHandle Doc, ppUns32 GState, ppReal AlphaStroke);
```

### File

VSGStateA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
AlphalsShape	[ in ] Alpha stroke.

### Returns

None.

### Description

Setting alpha stroke for current graphic state.

---

## 7.5 PDFExtGraphicStateSetBlendMode Function

```
LIB_API void PDFExtGraphicStateSetBlendMode(PDFDocHandle Doc, ppUns32 GState, TPDFBlendMode BlendMode);
```

### File

VSGStateA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
TPDFBlendMode BlendMode	[ in ] Blending mode.

### Returns

None.

### Description

Setting blending mode for current graphic state.

---

## 7.6 PDFExtGraphicStateSetCTM Function

```
LIB_API void PDFExtGraphicStateSetCTM(PDFDocHandle Doc, ppUns32 GState, ppReal a, ppReal b, ppReal c, ppReal d, ppReal e, ppReal f);
```

**File**

VSGStateA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
ppReal a	[ in ] A part of the CTM.
ppReal b	[ in ] B part of the CTM.
ppReal c	[ in ] C part of the CTM.
ppReal d	[ in ] D part of the CTM.
ppReal e	[ in ] E part of the CTM.
ppReal f	[ in ] F part of the CTM.

**Returns**

None.

**Description**

Setting CTM ( current transformation matrix ) width for current graphic state.

---

## 7.7 PDFExtGraphicStateSetDashPattern Function

```
LIB_API void PDFExtGraphicStateSetDashPattern(PDFDocHandle Doc, ppUns32 GState, PDFCosHandle DashPattern);
```

**File**

VSGStateA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
PDFCosHandle DashPattern	[ in ] Dash pattern.

**Returns**

None.

**Description**

Setting dash pattern for current graphic state.

---

## 7.8 PDFExtGraphicStateSetFlatness Function

```
LIB_API void PDFExtGraphicStateSetFlatness(PDFDocHandle Doc, ppUns32 GState, ppReal Flatness);
```

**File**

VSGStateA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
ppReal Flatness	[ in ] Graphic state flatness.

**Returns**

None.

**Description**

Setting flatness for current graphic state.

---

## 7.9 PDFExtGraphicStateSetLineCap Function

```
LIB_API void PDFExtGraphicStateSetLineCap(PDFDocHandle Doc, ppUns32 GState, TPDFLineCap LineCap);
```

**File**

VSGStateA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
TPDFLineCap LineCap	[ in ] Line cap.

**Returns**

None.

**Description**

Setting line cap for current graphic state.

---

## 7.10 PDFExtGraphicStateSetLineJoin Function

```
LIB_API void PDFExtGraphicStateSetLineJoin(PDFDocHandle Doc, ppUns32 GState, TPDFLineJoin LineJoin);
```

**File**

VSGStateA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
TPDFLineJoin LineJoin	[ in ] Line join.

**Returns**

None.

**Description**

Setting line cap for current graphic state.

---

## 7.11 PDFExtGraphicStateSetLineWidth Function

```
LIB_API void PDFExtGraphicStateSetLineWidth(PDFDocHandle Doc, ppUns32 GState, ppReal LineWidth);
```

**File**

VSGStateA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
ppReal LineWidth	[ in ] Line width.

**Returns**

None.

**Description**

Setting line width for current graphic state.

---

## 7.12 PDFExtGraphicStateSetMitterLimit Function

```
LIB_API void PDFExtGraphicStateSetMitterLimit(PDFDocHandle Doc, ppUns32 GState, ppReal MitterLimit);
```

**File**

VSGStateA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
ppReal MitterLimit	[ in ] Mitter limit.

**Returns**

None.

**Description**

Setting mitter limit for current graphic state.

---

## 7.13 PDFExtGraphicStateSetOverprintFill Function

```
LIB_API void PDFExtGraphicStateSetOverprintFill(PDFDocHandle Doc, ppUns32 GState, ppBool OverprintFill);
```

### File

VSGStateA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
ppBool OverprintFill	[ in ] Overprint fill.

### Returns

None.

### Description

Setting overprint fill for current graphic state.

---

## 7.14 PDFExtGraphicStateSetOverprintMode Function

```
LIB_API void PDFExtGraphicStateSetOverprintMode(PDFDocHandle Doc, ppUns32 GState, ppInt32 OverprintMode);
```

### File

VSGStateA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
ppInt32 OverprintMode	[ in ] Overprint mode.

### Returns

None.

### Description

Setting overprint mode for current graphic state.

---

## 7.15 PDFExtGraphicStateSetOverprintStroke Function

```
LIB_API void PDFExtGraphicStateSetOverprintStroke(PDFDocHandle Doc, ppUns32 GState, ppBool OverprintStroke);
```

### File

VSGStateA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
ppBool OverprintStroke	[ in ] Overprint stroke.

### Returns

None.

### Description

Setting overprint stroke for current graphic state.

---

## 7.16 PDFExtGraphicStateSetRenderingIntent Function

```
LIB_API void PDFExtGraphicStateSetRenderingIntent(PDFDocHandle Doc, ppUns32 GState, TPDFRenderingIntents Intent);
```

### File

VSGStateA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
TPDFRenderingIntents Intent	[ in ] Rendering intents.

### Returns

None.

### Description

Setting graphic state rendering intents.

---

## 7.17 PDFExtGraphicStateSetSmoothness Function

```
LIB_API void PDFExtGraphicStateSetSmoothness(PDFDocHandle Doc, ppUns32 GState, ppReal Smoothness);
```

### File

VSGStateA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
ppReal Smoothness	[ in ] Smoothness value.

### Returns

None.

### Description

Setting smoothness for current graphic state.

---

## 7.18 PDFExtGraphicStateSetStrokeAdjustment Function

```
LIB_API void PDFExtGraphicStateSetStrokeAdjustment(PDFDocHandle Doc, ppUns32 GState, ppBool StrokeAdjustment);
```

### File

VSGStateA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
ppBool StrokeAdjustment	[ in ] Stroke adjustment.

### Returns

None.

### Description

Setting stroke adjustment for current graphic state.

## 7.19 PDFExtGraphicStateSetTextKnockout Function

```
LIB_API void PDFExtGraphicStateSetTextKnockout(PDFDocHandle Doc, ppUns32 GState, ppBool TextKnockout);
```

### File

VSGStateA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in, out ] Current PDF document.
ppUns32 GState	[ in ] Graphic state index.
ppBool TextKnockout	[ in ] Text knockout.

### Returns

None.

### Description

Setting text knockout for current graphic state.



## 8 Font Level

There are functions for working with three type of the fonts in our library: standard fonts, True Type fonts and Type1 fonts.

Standard fonts are included in viewer applications. It's necessary to get access to font body for including this font into PDF document. It's possible to include subset of the font's glyphs if TrueType font is used.

### Functions

Function
✦ PDFFontStandardAppend (see page 45)
✦ PDFFontTrueTypeAppend (see page 45)
✦ PDFFontTrueTypeAppendFromFile (see page 46)
✦ PDFFontTrueTypeAppendFromStream (see page 46)
✦ PDFFontType1AppendFromFile (see page 47)
✦ PDFFontType1AppendFromStream (see page 47)

### Legend

✦	Method
---	--------

## 8.1 PDFFontStandardAppend Function

```
LIB_API ppUns32 PDFFontStandardAppend(PDFDocHandle Doc, TPDFStdandardFont font,
TPDFEncodingType encode);
```

### File

VSFontA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in ] Current PDF document.
TPDFStdandardFont font	[ in ] One of standart font: stdfHelvetica, stdfHelveticaBold, stdfHelveticaOblique, stdfHelveticaBoldOblique, stdfTimesRoman, stdfTimesBold, stdfTimesItalic, stdfTimesBoldItalic, stdfCourier, stdfCourierBold, stdfCourierOblique, stdfCourierBoldOblique, stdfSymbol, stdfZapfDingbats
TPDFEncodingType encode	[ in ] Font encoding type.

### Returns

Font index in PDF document.

### Description

Append one of 14 standard fonts to document.

## 8.2 PDFFontTrueTypeAppend Function

```
LIB_API ppUns32 PDFFontTrueTypeAppend(PDFDocHandle Doc, char * fontname, ppBool Bold,
```

```
ppBool Italic);
```

**File**

VSTFontA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in ] Current PDF document.
char * fontname	[ in ] Font name.
ppBool Bold	[ in ] Is bold style.
ppBool Italic	[ in ] Is italic style.

**Returns**

Font index in PDF document.

**Description**

Append true type font to document by name.

---

## 8.3 PDFFontTrueTypeAppendFromFile Function

```
LIB_API ppUns32 PDFFontTrueTypeAppendFromFile(PDFDocHandle Doc, char * fontfilename);
```

**File**

VSTFontA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in ] Current PDF document.
fontname	[ in ] Font filename.

**Returns**

Font index in PDF document.

**Description**

Append true type font to document from file.

---

## 8.4 PDFFontTrueTypeAppendFromStream Function

```
LIB_API ppUns32 PDFFontTrueTypeAppendFromStream(PDFDocHandle Doc, PDFStreamHandle Strm);
```

**File**

VSTFontA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in ] Current PDF document.
PDFStreamHandle Strm	[ in ] Font stream.

**Returns**

Font index in PDF document.

**Description**

Append true type font to document from stream.

---

## 8.5 PDFFontType1AppendFromFile Function

```
LIB_API ppUns32 PDFFontType1AppendFromFile(PDFDocHandle Doc, char * fontfilename);
```

**File**

VSFontA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in ] Current PDF document.
fontname	[ in ] Font filename.

**Returns**

Font index in PDF document.

**Description**

Append Type1 font to document from file.

---

## 8.6 PDFFontType1AppendFromStream Function

```
LIB_API ppUns32 PDFFontType1AppendFromStream(PDFDocHandle Doc, PDFStreamHandle Strm);
```

**File**

VSFontA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in ] Current PDF document.
PDFStreamHandle Strm	[ in ] Font stream.

**Returns**

Font index in PDF document.

**Description**

Append Type1 font to document from stream.

# 9 Canvas Drawing Level

This function level allows to use all power of the graphic and text operators in the work with PaintBox. The work with graphics, texts, styles and images is described here. It helps to create any practical page content filling.

## 9.1 Graphic State Operations

A PDF viewer application maintains an internal data structure called the graphics state that holds current graphic control parameters. These parameters define the global framework within which the graphics operators execute. For example, the Fill operator implicitly uses the current color parameter, and the Stroke operator additionally uses the current line width parameter from the graphics state. The graphics state is initialized at the beginning of each page.

### Functions

Function
✦ PBXNoDash (see page 48)
✦ PBXSetColor (see page 49)
✦ PBXSetDash (see page 49)
✦ PBXSetFillColor (see page 49)
✦ PBXSetFlatness (see page 50)
✦ PBXSetLineWidth (see page 50)
✦ PBXSetStrokeColor (see page 50)
✦ PBXStateRestore (see page 51)
✦ PBXStateStore (see page 51)
✦ PBXSetLineCap (see page 52)
✦ PBXSetLineJoin (see page 52)
✦ PBXSetMiterLimit (see page 52)
✦ PBXSetGState (see page 53)

### Legend

✦	Method
---	--------

### 9.1.1 PBXNoDash Function

```
LIB_API void PBXNoDash(PBXHandle PaintBox);
```

#### File

VSCanvasA.h

#### Parameters

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle

#### Returns

None.

**Description**

This function resets the dash pattern back to none, i.e., solid line.

---

## 9.1.2 PBXSetColor Function

```
LIB_API void PBXSetColor(PBXHandle PaintBox, TPDFColor Color);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
TPDFColor Color	[in] The color of the filling and stroking

**Returns**

None.

**Description**

This function sets both filling and stroking color to the specified values.

---

## 9.1.3 PBXSetDash Function

```
LIB_API void PBXSetDash(PBXHandle PaintBox, char * Dash);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle.
char * Dash	[in] Dash pattern style.

**Returns**

None.

**Description**

The line dash pattern controls the pattern of dashes and gaps used to stroke paths.

---

## 9.1.4 PBXSetFillColor Function

```
LIB_API void PBXSetFillColor(PBXHandle PaintBox, TPDFColor Color);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle

TPDFColor Color	[in] The color of the filling
-----------------	-------------------------------

**Returns**

None.

**Description**

This function sets filling color to the specified values .

---

## 9.1.5 PBXSetFlatness Function

```
LIB_API void PBXSetFlatness(PBXHandle PaintBox, ppInt32 Flatness);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle.
ppInt32 Flatness	[in] The flatness tolerance value.

**Returns**

None.

**Description**

The flatness tolerance controls the maximum permitted distance in device pixels between the mathematically correct path and an approximation constructed with straight line segments.

---

## 9.1.6 PBXSetLineWidth Function

```
LIB_API void PBXSetLineWidth(PBXHandle PaintBox, ppReal LineWidth);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle.
ppReal LineWidth	[in] Current line width.

**Returns**

None.

**Description**

This procedure sets the current line width to the value specified in points.

---

## 9.1.7 PBXSetStrokeColor Function

```
LIB_API void PBXSetStrokeColor(PBXHandle PaintBox, TPDFColor Color);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
TPDFColor Color	[in] The color of the stroke

**Returns**

None.

**Description**

This function sets stroking color to the specified values.

---

## 9.1.8 PBXStateRestore Function

```
LIB_API void PBXStateRestore(PBXHandle PaintBox);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle

**Returns**

None.

**Description**

This function restores the entire graphics state to its former value by popping it from the stack.

---

## 9.1.9 PBXStateStore Function

```
LIB_API void PBXStateStore(PBXHandle PaintBox);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle

**Returns**

None.

**Description**

This function pushes a copy of the entire graphics state onto the stack.

---

## 9.1.10 PBXSetLineCap Function

```
LIB_API void PBXSetLineCap(PBXHandle PaintBox, TPDFLineCap LineCap);
```

### File

VSCanvasA.h

### Parameters

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle.
TPDFLineCap LineCap	[in] Element of the TPDFLineCap (see page 173) enumeration that specifies the line cap.

### Returns

None.

### Description

Specifies line cap style.

---

## 9.1.11 PBXSetLineJoin Function

```
LIB_API void PBXSetLineJoin(PBXHandle PaintBox, TPDFLineJoin LineJoin);
```

### File

VSCanvasA.h

### Parameters

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
TPDFLineJoin LineJoin	[in] Element of the TPDFLineJoin (see page 174) enumeration that specifies the join style used at the end of a line segment that meets another line segment.

### Returns

None.

### Description

The SetLineJoin method sets the line join for this PaintBox.

---

## 9.1.12 PBXSetMiterLimit Function

```
LIB_API void PBXSetMiterLimit(PBXHandle PaintBox, ppReal MiterLimit);
```

### File

VSCanvasA.h

### Parameters

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle



ppReal MiterLimit	[in] Real number that specifies the miter limit of this PaintBox object.
-------------------	--

Returns

None.

Description

The SetMiterLimit method sets the miter limit of this PaintBox object.

# 9.1.13 PBXSetGState Function

```
LIB_API void PBXSetGState(PBXHandle PaintBox, ppUns32 Index);
```

File

VSCanvasA.h

Parameters

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppUns32 Index	[in] Index of the created extended graphic state returned from function PDFExtGraphicStateNew (see page 35)

Returns

None.

Description



This function sets extended graphic state which can be created with function PDFExtGraphicStateNew (see page 35).

# 9.2 Path Construction Operations

One of the key concepts of the PDF imaging model is a path which, in itself, an invisible contour without any markings on the page. Paths must be acted upon by path-painting operators to produce markings. A path may be stroked with a certain color and width, producing an actual curve on the page. A path may also be filled with a color. It may also be used to define a clipping path. Functions in this section are used to construct paths that are subsequently used to provide various effects.

Functions

Function
◆ PBXNewPath (see page 54)
◆ PBXArc (see page 54)
◆ PBXArc2 (see page 55)
◆ PBXEllipse (see page 55)
◆ PBXCircle (see page 56)
◆ PBXPie (see page 56)
◆ PBXPie2 (see page 57)
◆ PBXCurveTo (see page 58)
◆ PBXLineTo (see page 58)
◆ PBXMoveTo (see page 59)
◆ PBXRectangle (see page 59)
◆ PBXRectRotated (see page 60)

 PBXRoundRect ( see page 60) PBXClosePath ( see page 61)**Legend**

	Method
---	--------

## 9.2.1 PBXNewPath Function

```
LIB_API void PBXNewPath(PBXHandle PaintBox);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle

**Returns**

None.

**Description**

Clears the current path in the PaintBox. Current point becomes undefined.

## 9.2.2 PBXArc Function

```
LIB_API void PBXArc(PBXHandle PaintBox, ppReal x1, ppReal y1, ppReal x2, ppReal y2, ppReal x3, ppReal y3, ppReal x4, ppReal y4);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle.
ppReal x1	[in] Specifies the x-coordinate of the upper-left corner of the bounding rectangle.
ppReal y1	[in] Specifies the y-coordinate of the upper-left corner of the bounding rectangle.
ppReal x2	[in] Specifies the x-coordinate of the lower-right corner of the bounding rectangle.
ppReal y2	[in] Specifies the y-coordinate of the lower-right corner of the bounding rectangle.
ppReal x3	[in] Specifies the x-coordinate of the point that defines the arc's starting point.
ppReal y3	[in] Specifies the y-coordinate of the point that defines the arc's starting point .
ppReal x4	[in] Specifies the x-coordinate of the point that defines the arc's endpoint.
ppReal y4	[in] Specifies the y-coordinate of the point that defines the arc's endpoint.

**Returns**

None.

**Description**

Use Arc to create an elliptically curved path. The arc traverses the perimeter of an PBXEllipse (see page 55) which is bounded by the points ( x1, y1 ) and ( x2, y2 ). The drawn arc is following the perimeter of the ellipse , counterclockwise, from the starting point to the ending point. The starting point is defined by the intersection of the ellipse and a line is defined by the center of the ellipse and ( x3, y3). The ending point is defined by the intersection of the ellipse and a line is defined by the center of the ellipse and ( x4, y4 ).

---

## 9.2.3 PBXArc2 Function

```
LIB_API void PBXArc2(PBXHandle PaintBox, ppReal x1, ppReal y1, ppReal x2, ppReal y2, ppReal BeginAngle, ppReal EndAngle);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle.
ppReal x1	[in] Specifies the x-coordinate of the upper-left corner of the bounding rectangle.
ppReal y1	[in] Specifies the y-coordinate of the upper-left corner of the bounding rectangle.
ppReal x2	[in] Specifies the x-coordinate of the lower-right corner of the bounding rectangle.
ppReal y2	[in] Specifies the y-coordinate of the lower-right corner of the bounding rectangle.
ppReal BeginAngle	[in] Specifies the starting angle in degrees relative to the x-axis.
ppReal EndAngle	[in] Specifies the ending angle in degrees relative to the x-axis.

**Returns**

None.

**Description**

Use Arc to create an elliptically curved path. The arc traverses the perimeter of an ellipse which is bounded by the points ( x1, y1 ) and ( x2, y2 ). The drawn arc is following the perimeter of the ellipse, counterclockwise, from the starting point to the ending point. The starting point is defined by the intersection of the ellipse and a line is defined by BegAngle and EndAngle, specified in degrees.

---

## 9.2.4 PBXEllipse Function

```
LIB_API void PBXEllipse(PBXHandle PaintBox, ppReal x1, ppReal y1, ppReal x2, ppReal y2);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle.
ppReal x1	[in] The x-coordinate of the upper-left corner of the bounding rectangle.
ppReal y1	[in] The y-coordinate of the upper-left corner of the bounding rectangle.
ppReal x2	[in] The x-coordinate of the lower-right corner of the bounding rectangle.
ppReal y2	[in] The y-coordinate of the lower-right corner of the bounding rectangle.

**Returns**

None.

**Description**

This procedure creates an ellipse path specified by top left point at pixel coordinates ( x1, y1 ) and the bottom right point at ( x2, y2 ) in the counter-clock-wise direction.

**Notes**

If you need an ellipse drawn in the clock-wise direction, please use PBXArc (see page 54). This function performs a move to angle 0 (right edge) of the ellipse. Current point also will be at the same location after the call.

---

## 9.2.5 PBXCircle Function

```
LIB_API void PBXCircle(PBXHandle PaintBox, ppReal X, ppReal Y, ppReal R);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppReal X	[in] The X-coordinate of the center of the circle.
ppReal Y	[in] The Y-coordinate of the center of the circle.
ppReal R	[in] The radius of the circle.

**Returns**

None.

**Description**

This procedure creates a circular path centered at (X, Y) with radius "R" in the counter-clock-wise direction.

**Notes**

If you need a circle drawn in the clock-wise direction, please use PBXArc (see page 54). This function performs a move to angle 0 (right edge) of the circle. Current point also will be at the same location after the call.

---

## 9.2.6 PBXPie Function

```
LIB_API void PBXPie(PBXHandle PaintBox, ppReal x1, ppReal y1, ppReal x2, ppReal y2, ppReal x3, ppReal y3, ppReal x4, ppReal y4);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppReal x1	[in] Specifies the x-coordinate of the upper-left corner of the bounding rectangle.
ppReal y1	[in] Specifies the y-coordinate of the upper-left corner of the bounding rectangle.
ppReal x2	[in] Specifies the x-coordinate of the lower-right corner of the bounding rectangle.
ppReal y2	[in] Specifies the y-coordinate of the lower-right corner of the bounding rectangle.
ppReal x3	[in] Specifies the x-coordinate of the point that defines the arc's starting point.
ppReal y3	[in] Specifies the y-coordinate of the point that defines the arc's starting point .
ppReal x4	[in] Specifies the x-coordinate of the point that defines the arc's endpoint.
ppReal y4	[in] Specifies the y-coordinate of the point that defines the arc's endpoint.

**Returns**

None.

**Description**

Use Pie to append a pie-shaped wedge on the path. The wedge is defined by the ellipse bounded by the rectangle determined by the points ( x1, y1 ) and ( x2, y2). The drawn section is determined by two lines radiating from the center of the ellipse through the points ( x3, y3 ) and ( x4, y4 )

**Notes**

Current point is center of the wedge.

---

## 9.2.7 PBXPie2 Function

```
LIB_API void PBXPie2(PBXHandle PaintBox, ppReal x1, ppReal y1, ppReal x2, ppReal y2, ppReal BeginAngle, ppReal EndAngle);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppReal x1	[in] The x-coordinate of the upper-left corner of the rectangle.
ppReal y1	[in] The y-coordinate of the upper-left corner of the rectangle.
ppReal x2	[in] The x-coordinate of the lower-right corner of the rectangle.
ppReal y2	[in] The y-coordinate of the lower-right corner of the rectangle.
ppReal BeginAngle	[in] Specifies the starting angle in degrees relative to the x-axis.

ppReal EndAngle	[in] Specifies the ending angle in degrees relative to the x-axis.
-----------------	--

**Returns**

None.

**Description**

Use Pie to append a pie-shaped wedge on the path. The wedge is defined by the ellipse bounded by the rectangle determined by the points ( x1, y1 ) and ( x2, y2). The drawn section is determined by two lines ( BegAngle and EndAngle, specified in degrees).

**Notes**

Current point is center of the wedge.

---

## 9.2.8 PBXCurveTo Function

```
LIB_API void PBXCurveTo(PBXHandle PaintBox, ppReal x1, ppReal y1, ppReal x2, ppReal y2, ppReal x3, ppReal y3);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppReal x1	[in] The logical x-coordinate of the first control point position.
ppReal y1	[in] The logical y-coordinate of the first control point position.
ppReal x2	[in] The logical x-coordinate of the second control point position.
ppReal y2	[in] The logical y-coordinate of the second control point position.
ppReal x3	[in] The logical x-coordinate of the new position.
ppReal y3	[in] The logical y-coordinate of the new position.

**Returns**

None.

**Description**

This procedure adds a Bezier cubic curve segment to the path starting at the current point as ( x0, y0 ), using two points ( x1, y1 ) and ( x2, y2 ) as control points, and terminating at point ( x3, y3 ). The new current point will be ( x3, y3 ). If there is no current point, an error will result.

---

## 9.2.9 PBXLineTo Function

```
LIB_API void PBXLineTo(PBXHandle PaintBox, ppReal X, ppReal Y);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppReal X	[in] The logical x-coordinate of the endpoint for the line.
ppReal Y	[in] The logical y-coordinate of the endpoint for the line.

**Returns**

None.

**Description**

This procedure adds a line segment to the path, starting at the current point and ending at point ( x, y ).

Current point sets to ( x, y ).

---

## 9.2.10 PBXMoveTo Function

```
LIB_API void PBXMoveTo(PBXHandle PaintBox, ppReal X, ppReal Y);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppReal X	[in] The logical x-coordinate of the new position.
ppReal Y	[in] The logical y-coordinate of the new position.

**Returns**

None.

**Description**

This procedure moves the current point to the location specified by ( x, y ).

---

## 9.2.11 PBXRectangle Function

```
LIB_API void PBXRectangle(PBXHandle PaintBox, ppReal x1, ppReal y1, ppReal x2, ppReal y2);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppReal x1	[in] The x-coordinate of the upper-left corner of the rectangle.
ppReal y1	[in] The y-coordinate of the upper-left corner of the rectangle.
ppReal x2	[in] The x-coordinate of the lower-right corner of the rectangle.
ppReal y2	[in] The y-coordinate of the lower-right corner of the rectangle.

**Returns**

None.

**Description**

This function draws a rectangle with one corner at ( x1, y1 ) and second at ( x2, y2 ).

---

## 9.2.12 PBXRectRotated Function

```
LIB_API void PBXRectRotated(PBXHandle PaintBox, ppReal X, ppReal Y, ppReal W, ppReal H, ppReal Angle);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppReal X	[in] The x-coordinate of the lower-left corner of the rectangle.
ppReal Y	[in] The y-coordinate of the lower-left corner of the rectangle.
ppReal W	[in] The width of the rectangle.
ppReal H	[in] The height of the rectangle.
ppReal Angle	[in] Specifies the angle, in degrees, between the escapement vector and the x-axis of the device.

**Returns**

None.

**Description**

This function draws a rectangle of size ( w, h ) with one corner at ( x, y ), with an orientation argument, angle, specified in degrees.

---

## 9.2.13 PBXRoundRect Function

```
LIB_API void PBXRoundRect(PBXHandle PaintBox, ppReal x1, ppReal y1, ppReal x2, ppReal y2, ppReal x3, ppReal y3);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppReal x1	[in] The x-coordinate of the upper-left corner of the rectangle.
ppReal y1	[in] The y-coordinate of the upper-left corner of the rectangle.
ppReal x2	[in] The x-coordinate of the lower-right corner of the rectangle.
ppReal y2	[in] The y-coordinate of the lower-right corner of the rectangle.
ppReal x3	[in] Specifies the width of the ellipse used to draw the rounded corners.



ppReal y3	[in] Specifies the height of the ellipse used to draw the rounded corners.
-----------	--

**Returns**

None.

**Description**

Adds a rectangle with rounded corners to path. The rectangle will have edges defined by the points ( x1, y1 ), ( x2, y1 ), ( x2, y2 ), ( x1, y2 ), but the corners will be shaved to create a rounded appearance. The curve of the rounded corners matches the curvature of an ellipse with width x3 and height y3

## 9.2.14 PBXClosePath Function

```
LIB_API void PBXClosePath(PBXHandle PaintBox);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle

**Returns**

None.

**Description**

This closes a path by connecting the first and the last point in the path currently constructed. Calling of this procedure is often needed to avoid a notch in a stroked path, and to make "line join" work correctly in joining the first and the last points.

## 9.3 Path Painting Operations

Paths constructed by functions in the "Path construction" section are invisible, i.e., constructing a path does not produce any markings on the page. They must be stroked or filled.

**Functions**

Function
⇒ PBXClip (🔗 see page 62)
⇒ PBXEoClip (🔗 see page 62)
⇒ PBXEoFill (🔗 see page 62)
⇒ PBXEoFillAndStroke (🔗 see page 63)
⇒ PBXFill (🔗 see page 63)
⇒ PBXFillAndStroke (🔗 see page 64)
⇒ PBXStroke (🔗 see page 64)

**Legend**

⇒	Method
---	--------

---

## 9.3.1 PBXClip Function

```
LIB_API void PBXClip(PBXHandle PaintBox);
```

### File

VSCanvasA.h

### Parameters

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle

### Returns

None.

### Description

This function installs the current paths as the boundary for clipping of subsequent drawing. The use of the clip operator may require some care, because clip and eoclip operators do not consume the current path.

### Notes

There is no practical way of removing a clipping path, except by save and restore a graphical state before clipping is imposed.

---

## 9.3.2 PBXEoClip Function

```
LIB_API void PBXEoClip(PBXHandle PaintBox);
```

### File

VSCanvasA.h

### Parameters

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle

### Returns

None.

### Description

This function installs the current paths as the boundary for clipping subsequent drawing and uses the "even-odd" rule for defining the "inside" that shows through the clipping window. The use of the clip operator may require some care, because clip and eoclip operators do not consume the current path.

### Notes

There is not practical way of removing a clipping path, except by saving and restoring a graphical state before clipping is imposed.

---

## 9.3.3 PBXEoFill Function

```
LIB_API void PBXEoFill(PBXHandle PaintBox);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle

**Returns**

None.

**Description**

This function uses the current path as the boundary for color filling and uses the "evenodd" rule for defining an "inside" that is painted.

---

## 9.3.4 PBXEoFillAndStroke Function

```
LIB_API void PBXEoFillAndStroke(PBXHandle PaintBox);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle

**Returns**

None.

**Description**

This function is used for the first filling the inside with the current fill color ( uses the "non-zero winding number" rule ), and then stroking the path with the current stroke color. PDF's graphics state maintains separate colors to fill and stroke operations, thus these combined operators are available.

---

## 9.3.5 PBXFill Function

```
LIB_API void PBXFill(PBXHandle PaintBox);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle

**Returns**

None.

**Description**

This function uses the current path as the boundary for color filling and uses the "non-zero winding number" rule.

## 9.3.6 PBXFillAndStroke Function

```
LIB_API void PBXFillAndStroke(PBXHandle PaintBox);
```

### File

VSCanvasA.h

### Parameters

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle

### Returns

None.

### Description

This function is used for the first filling the inside with the current fill color, ( uses the "evenodd" rule for defining an "inside" that is painted ) and then stroking the path with the current stroke color. PDF's graphics state maintains separate colors to fill and stroke operations, thus these combined operators are available.

## 9.3.7 PBXStroke Function

```
LIB_API void PBXStroke(PBXHandle PaintBox);
```

### File

VSCanvasA.h

### Parameters

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle

### Returns

None.

### Description

This function strokes the current paths by the current stroke color and current line width.

## 9.4 Text Operations

These functions set is used to place in PaintBox in defined way. Except these functions we can set such properties as name, size and encoding of the font, character spacing, horizontal scaling, text rendering mode, word spacing. Also we may find out texts width.

### Functions

Function
⇒ PBXGetTextWidth (🔗 see page 65)
⇒ PBXGetUnicodeWidth (🔗 see page 65)
⇒ PBXSetCharacterSpacing (🔗 see page 66)

◆ PBXSetHorizontalScaling ( see page 66)
◆ PBXSetTextRenderingMode ( see page 66)
◆ PBXSetWordSpacing ( see page 67)
◆ PBXTextOut ( see page 67)
◆ PBXUnicodeTextOut ( see page 68)
◆ PBXSetActiveFont ( see page 68)
◆ PBXSetActiveFontWithCharset ( see page 69)

**Legend**

◆	Method
---	--------

## 9.4.1 PBXGetTextWidth Function

```
LIB_API ppReal PBXGetTextWidth(PBXHandle PaintBox, char * Text);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
char * Text	[in] Specifies the character string for which the text width is determined. Must be zero terminated.

**Returns**

Width of the text.

**Description**

Returns the width of a text string as it would be displayed in the current font.

## 9.4.2 PBXGetUnicodeWidth Function

```
LIB_API ppReal PBXGetUnicodeWidth(PBXHandle PaintBox, PppUns16 Text, ppInt32 Len);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
PppUns16 Text	[in] Specifies the character string for which the text width is determined.
ppInt32 Len	[in] Specifies the length of the string. It is a WORD count.

**Returns**

Width of the text.

**Description**

Returns the width of a text string as it would be displayed in the current font.

---

## 9.4.3 PBXSetCharacterSpacing Function

```
LIB_API void PBXSetCharacterSpacing(PBXHandle PaintBox, ppReal Spacing);
```

### File

VSCanvasA.h

### Parameters

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppReal Spacing	[in] Size between character in points

### Returns

None.

### Description

This function sets the additional space that should be inserted between characters.

---

## 9.4.4 PBXSetHorizontalScaling Function

```
LIB_API void PBXSetHorizontalScaling(PBXHandle PaintBox, ppReal Scale);
```

### File

VSCanvasA.h

### Parameters

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppReal Scale	[in] horizontal scaling factor in a percentage.

### Returns

None.

### Description

This function sets the horizontal scaling factor in a percentage. This essentially expands or compresses the horizontal dimension of the string. The default value for this parameter is 100 (%).

---

## 9.4.5 PBXSetTextRenderingMode Function

```
LIB_API void PBXSetTextRenderingMode(PBXHandle PaintBox, ppInt32 Mode);
```

### File

VSCanvasA.h

### Parameters

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppInt32 Mode	[in] Rendering mode

**Returns**

None.

**Description**

This function sets the mode that determines how the outline character is used. By default, the outline character is used for filling operations by which inside the outline path is painted solidly with the current fill color. This may be changed by calling this function.

**Notes**

Available modes at current time:

Mode	Action
0	Fill text.
1	Stroke text.
2	Fill, then stroke, text.
3	Neither fill nor stroke text (invisible).
4	Fill text and add to path for clipping.
5	Stroke text and add to path for clipping.
6	Fill , then stroke , text and add to path for clipping.
7	Add text to path for clipping.

---

## 9.4.6 PBXSetWordSpacing Function

```
LIB_API void PBXSetWordSpacing(PBXHandle PaintBox, ppReal Spacing);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppReal Spacing	[in] Additional space (in points) that should be inserted between words

**Returns**

None.

**Description**

This procedure sets the additional space (in points) that should be inserted between words, i.e., for every space character found in the text string.

---

## 9.4.7 PBXTextOut Function

```
LIB_API void PBXTextOut(PBXHandle PaintBox, ppReal X, ppReal Y, ppReal Orientation, char * TextStr);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppReal X	[in] Specifies the x-coordinate of the starting point of the text.
ppReal Y	[in] Specifies the y-coordinate of the starting point of the text.
ppReal Orientation	[in] Specifies the angle, in degrees, between the escapement vector and the x-axis of the device.
char * TextStr	[in] Pointer to the string to be drawn. Must be zero terminated.

**Returns**

None.

**Description**

Writes a character string at the specified location using the currently selected font.

---

## 9.4.8 PBXUnicodeTextOut Function

```
LIB_API void PBXUnicodeTextOut(PBXHandle PaintBox, ppReal X, ppReal Y, ppReal Orientation, PppUns16 Text, ppInt32 Len);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppReal X	[in] Specifies the x-coordinate of the starting point of the text.
ppReal Y	[in] Specifies the y-coordinate of the starting point of the text.
ppReal Orientation	[in] Specifies the angle, in degrees, between the escapement vector and the x-axis of the device.
PppUns16 Text	[in] Pointer to string for drawing.
ppInt32 Len	[in] Specifies the length of the string. It is a WORD count.

**Returns**

None.

**Description**

Writes a character string at the specified location using the currently selected font.

---

## 9.4.9 PBXSetActiveFont Function

```
LIB_API void PBXSetActiveFont(PBXHandle PaintBox, ppUns32 Index, ppReal FontSize, ppBool UnderLine, ppBool StrikeOut);
```

**File**

VSCanvasA.h



**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppUns32 Index	[in] Index of the font appended to PDF document early.
ppReal FontSize	[in] Size of output text, in units
ppBool StrikeOut	[in] Specifies an strikeout font if set to true.
Underline	[in] Specifies an underlined font if set to true.

**Returns**

None.

**Description**

This function sets the active font for text operations.

## 9.4.10 PBXSetActiveFontWithCharset Function

```
LIB_API void PBXSetActiveFontWithCharset(PBXHandle PaintBox, ppUns32 Index, ppReal
FontSize, ppUns8 Charset, ppBool UnderLine, ppBool StrikeOut);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppUns32 Index	[in] Index of the font appended to PDF document early.
ppReal FontSize	[in] Size of output text, in units
ppUns8 Charset	[in] Charset of the output text
ppBool StrikeOut	[in] Specifies an strikeout font if set to true.
Underline	[in] Specifies an underlined font if set to true.

**Returns**

None. Platform: Windows only.

**Description**

This function sets the active font for text operations.

## 9.5 Other Drawing Operations

There are functions not included in previous levels and which have own meanings with PaintBox work. These functions help to find out PaintBox size, to play metafile and to place image and to finish work normally.

**Functions**

Function
⇒ PBXGetHeight (🔗 see page 70)
⇒ PBXGetWidth (🔗 see page 70)
⇒ PBXPlayMetaFile (🔗 see page 70)
⇒ PBXClose (🔗 see page 71)

 PBXAppendLine (  see page 71)
 PBXShowImage (  see page 72)

Legend

	Method
---	--------

# 9.5.1 PBXGetHeight Function

```
LIB_API ppReal PBXGetHeight(PBXHandle PaintBox);
```

File

VSCanvasA.h

Parameters

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle

Returns

Height of the PaintBox in logical units.

Description

This function returns height of the PaintBox.

# 9.5.2 PBXGetWidth Function

```
LIB_API ppReal PBXGetWidth(PBXHandle PaintBox);
```

File

VSCanvasA.h

Parameters

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle

Returns

Width of the PaintBox in logical units.

Description

This function returns width of the PaintBox.

# 9.5.3 PBXPlayMetaFile Function

```
LIB_API void PBXPlayMetaFile(PBXHandle PaintBox, HGDIOBJ Metafile, ppReal X, ppReal Y, ppReal XScale, ppReal YScale);
```

File

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
HGDIOBJ Metafile	[in] Handle to the enhanced metafile
ppReal X	[in] The X-coordinate of the upper-left corner of the drawing rectangle.
ppReal Y	[in] The Y-coordinate of the upper-left corner of the drawing rectangle.
ppReal XScale	[in] X-scaling factor to play metafile.
ppReal YScale	[in] Y-scaling factor to play metafile.

**Returns**

None.

**Description**

Function paints context of the metafile on the page.

**Version**

Professional only.

**Platforms**

Windows only.

---

## 9.5.4 PBXClose Function

```
LIB_API void PBXClose(PBXHandle PaintBox, ppBool Pack);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object will be disposed.
ppBool Pack	[in] Indicated whether content of the PaintBox will be packed.

**Returns**

None.

**Description**

Disposes the instance of the PaintBox and packs content.

---

## 9.5.5 PBXAppendLine Function

```
LIB_API void PBXAppendLine(PBXHandle PaintBox, char * LineCode);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle

char * LineCode	[in] Line of the code which need append to content.
-----------------	---

**Returns**

None.

**Description**

Append misc line to content.

---

## 9.5.6 PBXShowImage Function

```
LIB_API void PBXShowImage(PBXHandle PaintBox, ppUns32 Index, ppReal X, ppReal Y, ppReal Width, ppReal Height, ppReal Angle);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
PBXHandle PaintBox	[in] PaintBox object handle
ppUns32 Index	[in] Index of the image appended to PDF document early.
ppReal X	[in] The x-coordinate of the lower-left corner of the image.
ppReal Y	[in] The y-coordinate of the lower-left corner of the image.
ppReal Width	[in] The width of the image.
ppReal Height	[in] The height of the image.
ppReal Angle	[in] Specifies the angle, in degrees, between the escapement vector and the x-axis.

**Returns**

None.

**Description**

This function places the image data of size ( Width , Height ) with one corner at (x,y), and angle, specified in degrees (ImageIndex is returned by one of the function appending image to PDF document) into the current content stream for the page

# 10 PDF Acroform Level

Acroforms - are PDF documents objects such as pushbutton, checkbox, radiobuttons, editbox, combobox, listbox or digital signature, placed in page document.

Acroform - is a collection of Acro Objects for gathering information interactively from the user. A PDF document may contain any number of Objects appearing on any combination of pages, all of which make up a single, global interactive form spanning the entire document. Arbitrary subsets of object's names and values can be imported or exported from the Document by means of Actions. Each Acro Object in a document's interactive form is defined by a properties.

## Functions

Function
PDFAcroGetCount (see page 73)
PDFAcroEditBoxSetAlign (see page 74)
PDFAcroEditBoxSetMaxLen (see page 74)
PDFAcroObjectAddAction (see page 75)
PDFAcroObjectAppendItem (see page 75)
PDFAcroObjectSetBorder (see page 76)
PDFAcroObjectSetCaption (see page 76)
PDFAcroObjectSetFlag (see page 76)
PDFAcroObjectSetFont (see page 77)
PDFAcroObjectSetStyle (see page 77)
PDFAcroPushButtonSetMiter (see page 78)
PDFDocAppendSignatureFromBuffer (see page 78)
PDFDocAppendSignatureFromFile (see page 79)
PDFDocAppendSignatureFromStream (see page 80)
PDFPageAppendCheckBox (see page 80)
PDFPageAppendComboBox (see page 81)
PDFPageAppendEditBox (see page 81)
PDFPageAppendListBox (see page 82)
PDFPageAppendPushButton (see page 82)
PDFPageAppendRadioButton (see page 83)
PDFPageAppendSignatureBox (see page 83)

## Legend

	Method
---	--------

## 10.1 PDFAcroGetCount Function

```
LIB_API ppUns32 PDFAcroGetCount(PDFDocHandle Doc);
```

### File

VSAcroInfoA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[in] Source Document.

**Returns**

Number of Acro Objects in Document, integer number.

**Description**

Gets number of Acro Objects in Document.

---

## 10.2 PDFAcroEditBoxSetAlign Function

```
LIB_API void PDFAcroEditBoxSetAlign(PDFDocHandle Doc, ppUns32 AcroIndex, TPDFAcroQuadding Align);
```

**File**

VSAcroFormA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] Document object handle.
ppUns32 AcroIndex	[in] Index of acroform object in document for which need set parameters.
TPDFAcroQuadding Align	[in] Text justification which need set.

**Returns**

None.

**Description**

Sets text justification of the edit box.

---

## 10.3 PDFAcroEditBoxSetMaxLen Function

```
LIB_API void PDFAcroEditBoxSetMaxLen(PDFDocHandle Doc, ppUns32 AcroIndex, ppUns32 MaxLen);
```

**File**

VSAcroFormA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] Document object handle.
ppUns32 AcroIndex	[in] Index of acroform object in document for which need set parameters.
ppUns32 MaxLen	[in] Maximum length of the edit box text.

**Returns**

None.

**Description**

Sets maximum length of the edit box text.

# 10.4 PDFAcroObjectAddAction Function

```
LIB_API void PDFAcroObjectAddAction(PDFDocHandle Doc, ppUns32 AcroIndex, PDFActionHandle Action, TPDFAcroEventType Type);
```

File

VSAcroFormA.h

Parameters

Parameters	Description
PDFDocHandle Doc	[in] Destination document.
PDFActionHandle Action	[in] Handle on action for include to trigger.
TPDFAcroEventType Type	[in] Type of event on Control for Action
AcroObjectIndex	[in] Index of acroform object in document.

Returns

None.

Description

Adds Action OnEvent in Acro Form Object (Control).

See Also

TPDFAcroEventType (see page 168), Actions

# 10.5 PDFAcroObjectAppendItem Function

```
LIB_API void PDFAcroObjectAppendItem(PDFDocHandle Doc, ppUns32 AcroIndex, char * Item);
```

File

VSAcroFormA.h

Parameters

Parameters	Description
PDFDocHandle Doc	[in] Document object handle.
ppUns32 AcroIndex	[in] Index of acroform object in document for which need set parameters.
char * Item	[in] Item which need append to list.

Returns

None.

Description

Appends items to ListBox or to ComboBox.

---

## 10.6 PDFAcroObjectSetBorder Function

```
LIB_API void PDFAcroObjectSetBorder(PDFDocHandle Doc, ppUns32 AcroIndex, TPDFColor BorderColor, TPDFColor FillColor, ppReal BorderWidth);
```

### File

VSAcroFormA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[in] Document object handle.
ppUns32 AcroIndex	[in] Index of acroform object in document for which need set parameters.
TPDFColor BorderColor	[in] Color of the border line.
TPDFColor FillColor	[in] Color of the background.
ppReal BorderWidth	[in] Width of the border line.

### Returns

None.

### Description

Sets Acroobject border

---

## 10.7 PDFAcroObjectSetCaption Function

```
LIB_API void PDFAcroObjectSetCaption(PDFDocHandle Doc, ppUns32 AcroIndex, char * Caption);
```

### File

VSAcroFormA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[in] Document object handle.
ppUns32 AcroIndex	[in] Index of acroform object in document for which need set parameters.
char * Caption	[in] Caption which need set .

### Returns

None.

### Description

Sets caption of the acroform object.

---

## 10.8 PDFAcroObjectSetFlag Function

```
LIB_API void PDFAcroObjectSetFlag(PDFDocHandle Doc, ppUns32 AcroIndex, ppUns32 Flag);
```



**File**

VSAcroFormA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] Document object handle.
ppUns32 AcroIndex	[in] Index of acroform object in document for which need set parameters.
ppUns32 Flag	[in] Characteristics of the acroform object .

**Returns**

None.

**Description**

Sets flags specifying various characteristics of the acroform object.

---

## 10.9 PDFAcroObjectSetFont Function

```
LIB_API void PDFAcroObjectSetFont(PDFDocHandle Doc, ppUns32 AcroIndex, ppUns32 FontIndex, ppReal FontSize, TPDFColor Color);
```

**File**

VSAcroFormA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] Document object handle.
ppUns32 AcroIndex	[in] Index of acroform object in document for which need set parameters.
ppUns32 FontIndex	[in] Index of the font stored in PDF document.
ppReal FontSize	[in] Size of the used font.
TPDFColor Color	[in] Color of the used font.

**Returns**

None.

**Description**

Sets Acroobject drawing font

---

## 10.10 PDFAcroObjectSetStyle Function

```
LIB_API void PDFAcroObjectSetStyle(PDFDocHandle Doc, ppUns32 AcroIndex, TPDFCheckBoxSign Sign, TPDFCheckBoxStyle Style);
```

**File**

VSAcroFormA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] Document object handle.
ppUns32 AcroIndex	[in] Index of acroform object in document for which need set parameters.
TPDFCheckBoxSign Sign	[in] Code of Mark character in CheckBox or RadioButton.
TPDFCheckBoxStyle Style	[in] Style of CheckBox or RadioButton - rectangle or circle.

**Returns**

None.

**Description**

Sets display style for radiobutton and checkbox.

## 10.11 PDFAcroPushButtonSetMiter Function

```
LIB_API void PDFAcroPushButtonSetMiter(PDFDocHandle Doc, ppUns32 AcroIndex, ppReal Miter);
```

**File**

VSAcroFormA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] Document object handle.
ppUns32 AcroIndex	[in] Index of acroform object in document for which need set parameters.
ppReal Miter	[in] Miter of pushbutton, bevel size.

**Returns**

None.

**Description**

Sets miter of the pushbutton.

## 10.12 PDFDocAppendSignatureFromBuffer Function

```
LIB_API void PDFDocAppendSignatureFromBuffer(PDFDocHandle Doc, void * Buffer, ppUns32 Length, char * Name, char * Reason, ppBool PKCS7, char * Password);
```

**File**

VSAcroFormA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] Destination document handle.
void * Buffer	[in] Pointer to memory buffer where stored PFX Personal Signature file.

ppUns32 Length	[in] Size of the memory buffer.
char * Name	[in] Name of the digital signature acro object field.
char * Reason	[in] Reason of Sign this document. Text string, for example "I agree..."
ppBool PKCS7	[in] Boolean flag of coding type :
char * Password	[in] Owner Password for Personal Signature. Text string
true	'Adobe.PPKMS' and 'adbe.pkcs7.sha1' crypt system sub filter
false	'Adobe.PPKLite' and 'adbe.x509.rsa_sha1' crypt system sub filter

**Returns**

None.

**Description**

Appends digital signature to PDF document stored in memory buffer.

## 10.13 PDFDocAppendSignatureFromFile Function

```
LIB_API void PDFDocAppendSignatureFromFile(PDFDocHandle Doc, char * FileName, char * Name, char * Reason, ppBool PKCS7, char * Password);
```

**File**

VSAcroFormA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] Destination document handle.
char * FileName	[in] Filename of the file where stored PFX Personal Signature.
char * Name	[in] Name of the digital signature acro object field.
char * Reason	[in] Reason of Sign this document. Text string, for example "I agree..."
ppBool PKCS7	[in] Boolean flag of coding type :
char * Password	[in] Owner Password for Personal Signature. Text string
true	'Adobe.PPKMS' and 'adbe.pkcs7.sha1' crypt system sub filter
false	'Adobe.PPKLite' and 'adbe.x509.rsa_sha1' crypt system sub filter

**Returns**

None.

**Description**

Appends digital signature to PDF document stored in file.

## 10.14 PDFDocAppendSignatureFromStream Function

```
LIB_API void PDFDocAppendSignatureFromStream(PDFDocHandle Doc, PDFStreamHandle Stream, char * Name, char * Reason, ppBool PKCS7, char * Password);
```

### File

VSAcroFormA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[in] Destination document handle.
PDFStreamHandle Stream	[in] Stream where stored PFX Personal Signature file.
char * Name	[in] Name of the digital signature acro object field.
char * Reason	[in] Reason of Sign this document. Text string, for example "I agree..."
ppBool PKCS7	[in] Boolean flag of coding type :
char * Password	[in] Owner Password for Personal Signature. Text string
Length	[in] Size of the memory buffer.
true	'Adobe.PPKMS' and 'adbe.pkcs7.sha1' crypt system sub filter
false	'Adobe.PPKLite' and 'adbe.x509.rsa_sha1' crypt system sub filter

### Returns

None.

### Description

Appends digital signature to PDF document stored in memory buffer.

## 10.15 PDFPageAppendCheckBox Function

```
LIB_API ppUns32 PDFPageAppendCheckBox(PDFDocHandle Doc, ppUns32 Page, TPDFRect R, char * Name, ppBool InitialState);
```

### File

VSAcroFormA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in ] PDF Document handle.
ppUns32 Page	[ in ] Index of the page where will located created check box.
char * Name	[ in ] Name of Acroform object, Name of Acroform field is used to export when the PDF document is submitted
ppBool InitialState	[ in ] Value of the check box.
Rect	[ in ] The annotation rectangle, defining the location of the acro field on the page

**Returns**

Index of acroform object in document.

**Description**

Create new acro check box.

---

## 10.16 PDFPageAppendComboBox Function

```
LIB_API ppUns32 PDFPageAppendComboBox(PDFDocHandle Doc, ppUns32 Page, TPDFRect R, char *  
Name);
```

**File**

VSacroFormA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in ] PDF Document handle.
ppUns32 Page	[ in ] Index of the page where will located created combo box.
char * Name	[ in ] Name of Acroform object, Name of Acroform field is used to export when the PDF document is submitted
Rect	[ in ] The annotation rectangle, defining the location of the acro field on the page

**Returns**

Index of acroform object in document.

**Description**

Create new acro combo box.

---

## 10.17 PDFPageAppendEditBox Function

```
LIB_API ppUns32 PDFPageAppendEditBox(PDFDocHandle Doc, ppUns32 Page, TPDFRect R, char *  
Name);
```

**File**

VSacroFormA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in ] PDF Document handle.
ppUns32 Page	[ in ] Index of the page where will located created edit box.
char * Name	[ in ] Name of Acroform object, Name of Acroform field is used to export when the PDF document is submitted
Rect	[ in ] The annotation rectangle, defining the location of the acro field on the page

**Returns**

Index of acroform object in document.

**Description**

Create new acro edit box.

---

## 10.18 PDFPageAppendListBox Function

```
LIB_API ppUns32 PDFPageAppendListBox(PDFDocHandle Doc, ppUns32 Page, TPDFRect R, char * Name);
```

**File**

VSAcroFormA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in ] PDF Document handle.
ppUns32 Page	[ in ] Index of the page where will located created list box.
char * Name	[ in ] Name of Acroform object, Name of Acroform field is used to export when the PDF document is submitted
Rect	[ in ] The annotation rectangle, defining the location of the acro field on the page

**Returns**

Index of acroform object in document.

**Description**

Create new acro list box.

---

## 10.19 PDFPageAppendPushButton Function

```
LIB_API ppUns32 PDFPageAppendPushButton(PDFDocHandle Doc, ppUns32 Page, TPDFRect R, char * Name);
```

**File**

VSAcroFormA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[ in ] PDF Document handle.
ppUns32 Page	[ in ] Index of the page where will located created push button.
char * Name	[ in ] Name of Acroform object, Name of Acroform field is used to export when the PDF document is submitted
Rect	[ in ] The annotation rectangle, defining the location of the acro field on the page

**Returns**

Index of acroform object in document.

**Description**

Create new acro push button.

## 10.20 PDFPageAppendRadioButton Function

```
LIB_API ppUns32 PDFPageAppendRadioButton(PDFDocHandle Doc, ppUns32 Page, TPDFRect R, char *
Name, char * Group, ppBool InitialState);
```

### File

VSAcroFormA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in ] PDF Document handle.
ppUns32 Page	[ in ] Index of the page where will located created radio button.
char * Name	[ in ] Name of Acroform object, Name of Acroform field is used to export when the PDF document is submitted
char * Group	[ in ] Name of Radio Buttons Group to which RadioButton will be linked.
ppBool InitialState	[ in ] Value of the check box.
Rect	[ in ] The annotation rectangle, defining the location of the acro field on the page

### Returns

Index of acroform object in document.

### Description

Create new acro radio button.

## 10.21 PDFPageAppendSignatureBox Function

```
LIB_API ppUns32 PDFPageAppendSignatureBox(PDFDocHandle Doc, ppUns32 Page, TPDFRect R, char
* Name);
```

### File

VSAcroFormA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in ] PDF Document handle.
ppUns32 Page	[ in ] Index of the page where will located created signature box.
char * Name	[ in ] Name of Acroform object, Name of Acroform field is used to export when the PDF document is submitted
Rect	[ in ] The annotation rectangle, defining the location of the acro field on the page

### Returns

Index of acroform object in document.



















### Description

Create new acro empty signature box.


# 11 PDF Outline Level

Outlines – is structured catalog of references to objects in document. Outline items are indirect references to visual object of the document. Outlines structure must be represented as a tree.

Functions

Function
 PDFDocGetOutlineRoot ( <a href="#">see page 84</a> )
 PDFOutlineAddNewChild ( <a href="#">see page 85</a> )
 PDFOutlineAddNewNext ( <a href="#">see page 85</a> )
 PDFOutlineAddNewPrev ( <a href="#">see page 85</a> )
 PDFOutlineAddNewSibling ( <a href="#">see page 86</a> )
 PDFOutlineGetCount ( <a href="#">see page 86</a> )
 PDFOutlineGetFirstChild ( <a href="#">see page 86</a> )
 PDFOutlineGetLastChild ( <a href="#">see page 87</a> )
 PDFOutlineGetNext ( <a href="#">see page 87</a> )
 PDFOutlineGetParent ( <a href="#">see page 88</a> )
 PDFOutlineGetPrev ( <a href="#">see page 88</a> )
 PDFOutlineHasChildren ( <a href="#">see page 88</a> )
 PDFOutlineSetAction ( <a href="#">see page 89</a> )
 PDFOutlineSetColor ( <a href="#">see page 89</a> )
 PDFOutlineSetDestination ( <a href="#">see page 90</a> )
 PDFOutlineSetExpanded ( <a href="#">see page 90</a> )
 PDFOutlineSetFlags ( <a href="#">see page 90</a> )
 PDFOutlineSetTitle ( <a href="#">see page 91</a> )

Legend

	Method
---	--------

## 11.1 PDFDocGetOutlineRoot Function

```
LIB_API PDFOutlineHandle PDFDocGetOutlineRoot(PDFDocHandle Doc);
```

File

VSOOutlineA.h

Parameters

Parameters	Description
PDFDocHandle Doc	[ in ] Current PDF document.

Returns

Handle to document outlines root.

Description

Returns root of outlines.



---

## 11.2 PDFOutlineAddNewChild Function

```
LIB_API PDFOutlineHandle PDFOutlineAddNewChild(PDFOutlineHandle Outline);
```

### File

VSOOutlineA.h

### Parameters

Parameters	Description
PDFOutlineHandle Outline	[ in ] Current outline.
Doc	[ in ] Current PDF document.

### Returns

Handle to created outline.

### Description

Creates outline as child for current outline.

---

## 11.3 PDFOutlineAddNewNext Function

```
LIB_API PDFOutlineHandle PDFOutlineAddNewNext(PDFOutlineHandle Outline);
```

### File

VSOOutlineA.h

### Parameters

Parameters	Description
PDFOutlineHandle Outline	[ in ] Current outline.
Doc	[ in ] Current PDF document.

### Returns

Handle to created outline.

### Description

Adds new outline which will be next for current outline.

---

## 11.4 PDFOutlineAddNewPrev Function

```
LIB_API PDFOutlineHandle PDFOutlineAddNewPrev(PDFOutlineHandle Outline);
```

### File

VSOOutlineA.h

### Parameters

Parameters	Description
PDFOutlineHandle Outline	[ in ] Current outline.

Doc	[ in ] Current PDF document.
-----	------------------------------

**Returns**

Handle to created outline.

**Description**

Adds new outline which will be previous for current outline.

---

## 11.5 PDFOutlineAddNewSibling Function

```
LIB_API PDFOutlineHandle PDFOutlineAddNewSibling(PDFOutlineHandle Outline);
```

**File**

VSOOutlineA.h

**Parameters**

Parameters	Description
PDFOutlineHandle Outline	[ in ] Current outline.
Doc	[ in ] Current PDF document.

**Returns**

Handle to created outline.

**Description**

Creates new outline which will be parallel for current outline.

---

## 11.6 PDFOutlineGetCount Function

```
LIB_API ppInt32 PDFOutlineGetCount(PDFOutlineHandle Outline);
```

**File**

VSOOutlineA.h

**Parameters**

Parameters	Description
PDFOutlineHandle Outline	[ in ] Current outline.
Doc	[ in ] Current PDF document.

**Returns**

Count of children.

**Description**

Calculates count of the children.

---

## 11.7 PDFOutlineGetFirstChild Function

```
LIB_API PDFOutlineHandle PDFOutlineGetFirstChild(PDFOutlineHandle Outline);
```

**File**

VSOutlineA.h

**Parameters**

Parameters	Description
PDFOutlineHandle Outline	[ in ] Current outline.
Doc	[ in ] Current PDF document.

**Returns**

Handle to the first outline child.

**Description**

Returns the first outline child.

---

## 11.8 PDFOutlineGetLastChild Function

```
LIB_API PDFOutlineHandle PDFOutlineGetLastChild(PDFOutlineHandle Outline);
```

**File**

VSOutlineA.h

**Parameters**

Parameters	Description
PDFOutlineHandle Outline	[ in ] Current outline.
Doc	[ in ] Current PDF document.

**Returns**

Handle to the last outline child.

**Description**

Returns the last outline child.

---

## 11.9 PDFOutlineGetNext Function

```
LIB_API PDFOutlineHandle PDFOutlineGetNext(PDFOutlineHandle Outline);
```

**File**

VSOutlineA.h

**Parameters**

Parameters	Description
PDFOutlineHandle Outline	[ in ] Current outline.
Doc	[ in ] Current PDF document.

**Returns**

Handle to following outline.

**Description**

Returns following outline.

---

## 11.10 PDFOutlineGetParent Function

```
LIB_API PDFOutlineHandle PDFOutlineGetParent(PDFOutlineHandle Outline);
```

**File**

VSOOutlineA.h

**Parameters**

Parameters	Description
PDFOutlineHandle Outline	[ in ] Current outline.
Doc	[ in ] Current PDF document.

**Returns**

Handle to parent outline.

**Description**

Returns parent outline.

---

## 11.11 PDFOutlineGetPrev Function

```
LIB_API PDFOutlineHandle PDFOutlineGetPrev(PDFOutlineHandle Outline);
```

**File**

VSOOutlineA.h

**Parameters**

Parameters	Description
PDFOutlineHandle Outline	[ in ] Current outline.
Doc	[ in ] Current PDF document.

**Returns**

Handle to previous outline.

**Description**

Returns previous outline.

---

## 11.12 PDFOutlineHasChildren Function

```
LIB_API ppBool PDFOutlineHasChildren(PDFOutlineHandle Outline);
```

**File**

VSOOutlineA.h

**Parameters**

Parameters	Description
PDFOutlineHandle Outline	[ in ] Current outline.

Doc	[ in ] Current PDF document.
-----	------------------------------

**Returns**

If current outline has children - true, else - false.

**Description**

Inspects current outline on children presence.

---

## 11.13 PDFOutlineSetAction Function

```
LIB_API void PDFOutlineSetAction(PDFOutlineHandle Outline, PDFActionHandle Action);
```

**File**

VSOOutlineA.h

**Parameters**

Parameters	Description
PDFOutlineHandle Outline	[ in ] Current outline.
PDFActionHandle Action	[ in ] Linked action.
Doc	[ in ] Current PDF document.

**Returns**

None.

**Description**

Links action to outline.

---

## 11.14 PDFOutlineSetColor Function

```
LIB_API void PDFOutlineSetColor(PDFOutlineHandle Outline, PPDFColor Color);
```

**File**

VSOOutlineA.h

**Parameters**

Parameters	Description
PDFOutlineHandle Outline	[ in ] Current outline.
PPDFColor Color	[ in ] Outline color.
Doc	[ in ] Current PDF document.

**Returns**

None.

**Description**

Sets color of outline.

---

# 11.15 PDFOutlineSetDestination Function

```
LIB_API void PDFOutlineSetDestination(PDFOutlineHandle Outline, PDFDestinationHandle Destination);
```

**File**

VSOOutlineA.h

**Parameters**

Parameters	Description
PDFOutlineHandle Outline	[ in ] Current outline.
PDFDestinationHandle Destination	[ in ] Linked object handle.
Doc	[ in ] Current PDF document.

**Returns**

None.

**Description**

Links outline to object.

# 11.16 PDFOutlineSetExpanded Function

```
LIB_API void PDFOutlineSetExpanded(PDFOutlineHandle Outline, ppBool Expanded);
```

**File**

VSOOutlineA.h

**Parameters**

Parameters	Description
PDFOutlineHandle Outline	[ in ] Current outline.
ppBool Expanded	[ in ] Outline expand state.
Doc	[ in ] Current PDF document.

**Returns**

None.

**Description**

Sets current outline on expanding.

# 11.17 PDFOutlineSetFlags Function

```
LIB_API void PDFOutlineSetFlags(PDFOutlineHandle Outline, ppInt32 Flags);
```

**File**

VSOOutlineA.h

**Parameters**

Parameters	Description
PDFOutlineHandle Outline	[ in ] Current outline.
ppInt32 Flags	[ in ] Outline flags.
Doc	[ in ] Current PDF document.

**Returns**

None.

**Description**

Sets the outline flags.

---

## 11.18 PDFOutlineSetTitle Function

```
LIB_API void PDFOutlineSetTitle(PDFOutlineHandle Outline, char * str, ppInt32 len);
```

**File**

VSOOutlineA.h

**Parameters**

Parameters	Description
PDFOutlineHandle Outline	[ in ] Current outline.
char * str	[ in ] Outline title.
ppInt32 len	[ in ] Title length.
Doc	[ in ] Current PDF document.

**Returns**

None.

**Description**

Sets title of outline.

# 12 Thread and Bead Level

## 12.1 Thread Operations

Functions

Function
⇒ PDFThreadNew (🔗 see page 92)
⇒ PDFDocGetThread (🔗 see page 93)
⇒ PDFDocGetThreadCount (🔗 see page 93)
⇒ PDFThreadAppendBead (🔗 see page 93)
⇒ PDFThreadDelete (🔗 see page 94)
⇒ PDFThreadGetFirstBead (🔗 see page 94)
⇒ PDFThreadGetInfo (🔗 see page 95)
⇒ PDFThreadSetInfo (🔗 see page 95)

Legend

⇒	Method
---	--------

### 12.1.1 PDFThreadNew Function

```
LIB_API PDFThreadHandle PDFThreadNew(PDFDocHandle Doc);
```

File

VSThreadA.h

Parameters

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.

Returns

New Thread Handle in PDF Document.

Description

Creates New Thread in PDF Document.

See Also

PDFThreadDelete (🔗 see page 94)



---

## 12.1.2 PDFDocGetThread Function

```
LIB_API PDFThreadHandle PDFDocGetThread(PDFDocHandle Doc, ppInt32 Index);
```

### File

VSThreadA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.
ppInt32 Index	[in] Index of Thread in PDF Document.

### Returns

Thread Handle.

### Description

Gets Thread of PDF Document according to Index.

### See Also

PDFThreadHandle

---

## 12.1.3 PDFDocGetThreadCount Function

```
LIB_API ppInt32 PDFDocGetThreadCount(PDFDocHandle Doc);
```

### File

VSThreadA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[in] PDF Document Handle.

### Returns

Number of Threads Items.

### Description

Gets Threads Count in PDF Document.

---

## 12.1.4 PDFThreadAppendBead Function

```
LIB_API PDFBeadHandle PDFThreadAppendBead(PDFThreadHandle Thread, ppUns32 Page, TPDFRect Rect);
```

### File

VSThreadA.h

### Parameters

Parameters	Description
PDFThreadHandle Thread	[in] Thread Handle.

ppUns32 Page	[in] Index of Page on which this bead will be appeared.
TPDFRect Rect	[in] A rectangle specifying the location of this bead on the page.

**Returns**

Bead Handle.

**Description**

Creates new bead on the page for Thread.

---

## 12.1.5 PDFThreadDelete Function

```
LIB_API void PDFThreadDelete(PDFThreadHandle Thread);
```

**File**

VSThreadA.h

**Parameters**

Parameters	Description
PDFThreadHandle Thread	[in] Thread Handle in PDF Document for Deleting.
Doc	[in] PDF Document Handle.

**Returns**

None.

**Description**

Deletes Thread from PDF Document.

**See Also**

PDFThreadNew (🔗 see page 92)

---

## 12.1.6 PDFThreadGetFirstBead Function

```
LIB_API PDFBeadHandle PDFThreadGetFirstBead(PDFThreadHandle Thread);
```

**File**

VSThreadA.h

**Parameters**

Parameters	Description
PDFThreadHandle Thread	[in] Thread Handle in PDF Document for founding the first bead.
Doc	[in] PDF Document Handle.

**Returns**

Bead Handle on the First Bead in Thread.

**Description**

Gets Bead Handle on First Bead in Thread.

**See Also**

PDFThreadSetFirstBead

## 12.1.7 PDFThreadGetInfo Function

```
LIB_API char * PDFThreadGetInfo(PDFThreadHandle Thread, TPDFInformation Info, ppUns32 * StrLen);
```

### File

VSThreadA.h

### Parameters

Parameters	Description
PDFThreadHandle Thread	[in] Thread Handle in PDF Document.
Doc	[in] PDF Document Handle.
InfoKey	[in] Name of requesting information.
Value	[out] Text value of requesting information.
MaxLength	[in] Maximum length of value in bytes.

### Returns

Length of value in bytes ( not longer than MaxLength ).

### Description

Gets information from Thread according to Name.

### See Also

PDFThreadSetInfo (🔗 see page 95)

## 12.1.8 PDFThreadSetInfo Function

```
LIB_API void PDFThreadSetInfo(PDFThreadHandle Thread, TPDFInformation Info, char * Value, ppUns32 Length);
```

### File

VSThreadA.h

### Parameters

Parameters	Description
PDFThreadHandle Thread	[in] Thread Handle in PDF Document.
char * Value	[in] Text string information value.
ppUns32 Length	[in] Length of value in bytes.
Doc	[in] PDF Document Handle.
InfoKey	[in] Name of setting information.

### Returns

None.

### Description

Sets information to Thread according to property name.

### See Also

PDFThreadGetInfo (🔗 see page 95)

# 12.2 Bead Operations

Functions

Function
PDFBeadGetNext (see page 96)
PDFBeadGetPage (see page 96)
PDFBeadGetPrev (see page 97)
PDFBeadGetRect (see page 97)
PDFBeadGetThread (see page 98)
PDFBeadSetRect (see page 98)

Legend

	Method
--	--------

## 12.2.1 PDFBeadGetNext Function

```
LIB_API PDFBeadHandle PDFBeadGetNext(PDFBeadHandle Bead);
```

File

VSThreadA.h

Parameters

Parameters	Description
PDFBeadHandle Bead	[in] Bead whence we go onward.
Doc	[in] PDF Document Handle.

Returns

Handle of next bead.

Description

Navigates to the next bead item.

See Also

PDFBeadGetPrev (see page 97)

## 12.2.2 PDFBeadGetPage Function

```
LIB_API ppInt32 PDFBeadGetPage(PDFBeadHandle Bead);
```

File

VSThreadA.h

Parameters

Parameters	Description
PDFBeadHandle Bead	[in] Bead Handle.
Doc	[in] PDF Document Handle.

**Returns**

Index of Page on which this bead appears.

**Description**

Gets an Index of Page on which this bead appears.

---

## 12.2.3 PDFBeadGetPrev Function

```
LIB_API PDFBeadHandle PDFBeadGetPrev(PDFBeadHandle Bead);
```

**File**

VSThreadA.h

**Parameters**

Parameters	Description
PDFBeadHandle Bead	[in] Bead whence we go back.
Doc	[in] PDF Document Handle.

**Returns**

Handle of previous bead.

**Description**

Navigates to previous bead item.

**See Also**

PDFBeadGetNext ([↗](#) see page 96)

---

## 12.2.4 PDFBeadGetRect Function

```
LIB_API void PDFBeadGetRect(PDFBeadHandle Bead, TPDFRect * Rect);
```

**File**

VSThreadA.h

**Parameters**

Parameters	Description
PDFBeadHandle Bead	[in] Bead Handle.
TPDFRect * Rect	[out] Rectangle specifying the location of this bead on the page.
Doc	[in] PDF Document Handle.

**Returns**

None.

**Description**

Gets a rectangle specifying the location of this bead on the page.

**See Also**

PDFBeadSetRect ([↗](#) see page 98)

# 12.2.5 PDFBeadGetThread Function

```
LIB_API PDFThreadHandle PDFBeadGetThread(PDFBeadHandle Bead);
```

**File**

VSThreadA.h

**Parameters**

Parameters	Description
PDFBeadHandle Bead	[in] Bead Handle.
Doc	[in] PDF Document Handle.

**Returns**

Handle of Bead's Thread.

**Description**

Gets a Handle of Bead's Thread.

# 12.2.6 PDFBeadSetRect Function

```
LIB_API void PDFBeadSetRect(PDFBeadHandle Bead, TPDFRect Rect);
```

**File**

VSThreadA.h

**Parameters**

Parameters	Description
PDFBeadHandle Bead	[in] Bead Handle.
TPDFRect Rect	[out] Rectangle specifying the location of this bead on the page.
Doc	[in] PDF Document Handle.

**Returns**

None.

**Description**

Sets a rectangle specifying the location of this bead on the page.

**See Also**

PDFBeadGetRect ([↗](#) see page 97)

# 13 Annotation Level

???????? - are PDF document objects such as note, sound, movie and etc. placed on the document page.

Many standard types of the annotations may be on the page in opened or closed condition. When annotations are closed they are displayed on the page in conditional form depending on their type, for example icon, mailbox or stamp. When user activates annotation clicking the mouse on annotation will show object connected with it, for example window with text note or video clip or sound playing.

Functions

Function
PDFAnnotationSetAction ( see page 99)
PDFAnnotationSetAlphaBlending ( see page 100)
PDFAnnotationSetBorderStyle ( see page 100)
PDFAnnotationSetColor ( see page 100)
PDFAnnotationSetFlag ( see page 101)
PDFAnnotationSetName ( see page 101)
PDFAnnotationSetTitle ( see page 102)
PDFPageAppendAnnotationLinkWithAction ( see page 102)
PDFPageAppendAnnotationLinkWithDest ( see page 103)
PDFPageAppendAnnotationStamp ( see page 103)
PDFPageAppendAnnotationText ( see page 104)

Legend

	Method
--	--------

## 13.1 PDFAnnotationSetAction Function

```
LIB_API void PDFAnnotationSetAction(PDFAnnotationHandle Annotation, PDFActionHandle Action);
```

File

VSAnnotA.h

Parameters

Parameters	Description
PDFActionHandle Action	[ in ] Specifies annotation's action.
Annotaion	[ in ] PDF annotation for which need set action.

Returns

None.

Description

Sets action for PDF Annotation which will executed when annotation is activated.

## 13.2 PDFAnnotationSetAlphaBlending Function

```
LIB_API void PDFAnnotationSetAlphaBlending(PDFAnnotationHandle Annotation, ppReal AlphaBlending);
```

### File

VSAnnotA.h

### Parameters

Parameters	Description
ppReal AlphaBlending	[ in ] Specifies annotation's alpha blending.
Annotaion	[ in ] PDF annotation for which need set alpha blending.

### Returns

None.

### Description

Sets alpha blending for PDF Annotation.

## 13.3 PDFAnnotationSetBorderStyle Function

```
LIB_API void PDFAnnotationSetBorderStyle(PDFAnnotationHandle Annotation, ppReal Width, TBorderStyle Style, PDFCosHandle Dash);
```

### File

VSAnnotA.h

### Parameters

Parameters	Description
ppReal Width	[ in ] Specifies border's width.
TBorderStyle Style	[ in ] Specifies border's style.
PDFCosHandle Dash	[ in ] Specifies dash of the border. Used only with style equal to bsDashed.
Annotaion	[ in ] PDF annotation for which need set border style.

### Returns

None.

### Description

Sets border style for PDF Annotation.

## 13.4 PDFAnnotationSetColor Function

```
LIB_API void PDFAnnotationSetColor(PDFAnnotationHandle Annotation, TPDFColor Color);
```

### File

VSAnnotA.h



**Parameters**

Parameters	Description
TPDFColor Color	[ in ] Specifies annotation's name.
Annotaion	[ in ] PDF annotation for which need set color.

**Returns**

None.

**Description**

Sets color for PDF Annotation.

---

## 13.5 PDFAnnotationSetFlag Function

```
LIB_API void PDFAnnotationSetFlag(PDFAnnotationHandle Annotation, ppUns16 AnnotationFlag);
```

**File**

VSAnnotA.h

**Parameters**

Parameters	Description
Annotaion	[ in ] PDF annotation for which need set flags.
AnnotationFlags	[ in ] Specifies annotation's flags.

**Returns**

None.

**Description**

Sets flag for PDF Annotation.

---

## 13.6 PDFAnnotationSetName Function

```
LIB_API void PDFAnnotationSetName(PDFAnnotationHandle Annotation, char * Name, ppUns32 NameLength);
```

**File**

VSAnnotA.h

**Parameters**

Parameters	Description
char * Name	[ in ] Specifies annotation's name.
ppUns32 NameLength	[ in ] Specifies title's name.
Annotaion	[ in ] PDF annotation for which need set name.

**Returns**

None.

**Description**

Sets Name for PDF Annotation.

## 13.7 PDFAnnotationSetTitle Function

```
LIB_API void PDFAnnotationSetTitle(PDFAnnotationHandle Annotation, char * Title, ppUns32 TitleLength);
```

### File

VSAnnotA.h

### Parameters

Parameters	Description
char * Title	[ in ] Specifies annotation's title.
ppUns32 TitleLength	[ in ] Specifies title's length.
Annotaion	[ in ] PDF annotation for which need set title.

### Returns

None.

### Description

Sets Title for PDF Annotation.

## 13.8 PDFPageAppendAnnotationLinkWithAction Function

```
LIB_API PDFAnnotationHandle PDFPageAppendAnnotationLinkWithAction(PDFDocHandle Doc, ppUns32 Page, TPDFRect Rect, PDFActionHandle Action, THighLightMode Mode, ppBool Visible);
```

### File

VSAnnotA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in ] PDF Document handle.
ppUns32 Page	[ in ] Index of the page where will located created link annotation.
TPDFRect Rect	[ in ] The annotation rectangle, defining the location of the annotation on the page.
PDFActionHandle Action	[ in ] An action to be performed when the annotation is activated.
THighLightMode Mode	[ in ] The annotation's highlighting mode.
ppBool Visible	[ in ] No border is drawn if set to false.

### Returns

Link annotation handle.

### Description

Create new link annotation.

## 13.9 PDFPageAppendAnnotationLinkWithDest Function

```
LIB_API PDFAnnotationHandle PDFPageAppendAnnotationLinkWithDest(PDFDocHandle Doc, ppUns32 Page, TPDFRect Rect, PDFDestinationHandle Destination, THighLightMode Mode, ppBool Visible);
```

### File

VSAnnotA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in ] PDF Document handle.
ppUns32 Page	[ in ] Index of the page where will located created link annotation.
TPDFRect Rect	[ in ] The annotation rectangle, defining the location of the annotation on the page.
PDFDestinationHandle Destination	[ in ] A destination to be displayed when the annotation is activated.
THighLightMode Mode	[ in ] The annotation's highlighting mode.
ppBool Visible	[ in ] No border is drawn if set to false.

### Returns

Link annotation handle.

### Description

Create new link annotation.

## 13.10 PDFPageAppendAnnotationStamp Function

```
LIB_API PDFAnnotationHandle PDFPageAppendAnnotationStamp(PDFDocHandle Doc, ppUns32 Page, TPDFRect Rect, char * Content, ppUns32 ContentLength, TAnnotationStampName Name);
```

### File

VSAnnotA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[ in ] PDF Document handle.
ppUns32 Page	[ in ] Index of the page where will located created stamp annotation.
TPDFRect Rect	[ in ] The annotation rectangle, defining the location of the annotation on the page.
char * Content	[ in ] Text to be displayed for the annotation
ppUns32 ContentLength	[ in ] Length of the content.
TAnnotationStampName Name	[ in ] The name of an icon to be used in displaying the annotation.

Returns

Link annotation handle.

Description

Create new stamp annotaion.

# 13.11 PDFPageAppendAnnotationText Function

```
LIB_API PDFAnnotationHandle PDFPageAppendAnnotationText(PDFDocHandle Doc, ppUns32 Page,
TPDFRect Rect, char * Content, ppUns32 ContentLength, ppBool IsOpened, TAnnotationTextName
Name);
```

File

VSAnnotA.h

Parameters

Parameters	Description
PDFDocHandle Doc	[ in ] PDF Document handle.
ppUns32 Page	[ in ] Index of the page where will located created text annotation.
TPDFRect Rect	[ in ] The annotation rectangle, defining the location of the annotation on the page
char * Content	[ in ] Text to be displayed for the annotation
ppUns32 ContentLength	[ in ] Length of the content.
ppBool IsOpened	[ in ] Specifying whether the annotation should initially be displayed open.
TAnnotationTextName Name	[ in ] The name of an icon to be used in displaying the annotation.

Returns

Text annotation handle.









Description

Create new text annotation.


# 14 Action Level

Instead of simply jumping to a destination in the document, an annotation or outline item can specify an action for the viewer application to perform, such as launching an application, playing a sound, or changing an annotation's appearance state. The optional action entry in the annotation or outline item dictionary specifies an action to be performed when the annotation or outline item is activated; A variety of other circumstances may trigger an action as well. PDF includes a wide variety of standard action types.

Functions

Function
 PDFActionSetNext (  see page 105)
 PDFDestinationFromExplit (  see page 105)
 PDFDestinationFromString (  see page 106)
 PDFDestinationNamedNew (  see page 106)

Legend

	Method
---	--------

## 14.1 PDFActionSetNext Function

```
LIB_API void PDFActionSetNext(PDFActionHandle Action, PDFActionHandle Next);
```

File

VSActionA.h

Parameters

Parameters	Description
PDFActionHandle Action	[in] Handle of the action object.
PDFActionHandle Next	[in] Handle of the next action object.

Returns

None.

Description

This function sets action which will be executed after current action.

## 14.2 PDFDestinationFromExplit Function

```
LIB_API PDFDestinationHandle PDFDestinationFromExplit(PDFDocHandle Doc, PDFExplicitDest Dest, ppBool OtherDocument, ppBool IsIndirect);
```

File

VSActionA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document handle
PDFExplicitDest Dest	[in] Explicit destination
ppBool OtherDocument	[in] Specifies will destination is this document or other.
ppBool IsIndirect	[in] Specifies store this destination in memory or append to PDFDocument.

**Returns**

Destination Handle.

**Description**

Creates new destination handle from string.

**Notes**

Memory Leak will present if IsIndirect is false and destination not appended to any Action

---

## 14.3 PDFDestinationFromString Function

```
LIB_API PDFDestinationHandle PDFDestinationFromString(PDFDocHandle Doc, char * String, ppInt32 Length, ppBool IsIndirect);
```

**File**

VSActionA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document handle
char * String	[in] Specifies the name for which will created destination handle.
ppInt32 Length	[in] Length of the name
ppBool IsIndirect	[in] Specifies store this destination in memory or append to PDFDocument.

**Returns**

Destination Handle.

**Description**

Creates new destination handle from string.

**Notes**

Memory Leak will present if IsIndirect is false and destination not appended to any Action

---

## 14.4 PDFDestinationNamedNew Function

```
LIB_API void PDFDestinationNamedNew(PDFDocHandle Doc, char * String, ppInt32 Length, PDFExplicitDest Destination);
```

**File**

VSActionA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document handle
char * String	[in] Specifies the name which will be appended to name table.
ppInt32 Length	[in] Length of the name
PDFExplicitDest Destination	[in] Explicit destination which will be respected to this name

**Returns**

None.

**Description**

Creates new destination name in name table and assigns to it explicit destination

---


## 14.5 Goto Action

A Go-to action changes the view to a specified destination (page, location, and magnification factor).

**Functions**

Function
 PDFActionNewGoToDestination (  see page 107)

**Legend**

	Method
---	--------

---

### 14.5.1 PDFActionNewGoToDestination Function

```
LIB_API PDFActionHandle PDFActionNewGoToDestination(PDFDocHandle Doc, PDFDestinationHandle Destination);
```

**File**

VSActionA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document handle
PDFDestinationHandle Destination	[in] Destination handle.

**Returns**

The return value is a handle to a PDF action.

**Description**

GoTo Action

Creates new "GoTo" action and sets destination to explicit destination.

## 14.6 Goto Remote Action

A remote go-to action is similar to an ordinary go-to action, but jumps to a destination in another PDF file instead of the current file.

### Functions

#### Function

PDFActionNewGoToRemote (see page 108)

### Legend

	Method
---	--------

## 14.6.1 PDFActionNewGoToRemote Function

```
LIB_API PDFActionHandle PDFActionNewGoToRemote(PDFDocHandle Doc, char * FileName, ppInt32
FileNameLength, PDFDestinationHandle Destination, ppBool InNewWindow);
```

### File

VSActionA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[in] PDF Document handle
char * FileName	[in] Filename of the PDF document which need open
ppInt32 FileNameLength	[in] Length of the filename.
PDFDestinationHandle Destination	[in] Destination Handle.
ppBool InNewWindow	[in] Specifying whether to open the destination document in a new window. If this flag is false, the destination document will replace the current document in the same window.

### Returns

The return value is a handle to a PDF action.

### Description

Creates new "GoToRemote" action and sets destination to name destination.

## 14.7 Launch Action


A launch action launches an application or opens or prints a document.

### Functions

#### Function

PDFActionNewLaunch (see page 109)

### Legend

	Method
---	--------



# 14.7.1 PDFActionNewLaunch Function

```
LIB_API PDFActionHandle PDFActionNewLaunch(PDFDocHandle Doc, char * FileName, char *
DefaultDir, char * Operation, char * Params, ppBool InNewWindow);
```

**File**

VSActionA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document handle
ppBool InNewWindow	[in] Specifying whether to open the destination document in a new window. If this flag is false, the destination document will replace the current document in the same window.
Launch	[in] Point to a <b>PDFLaunch</b> structure that defines the characteristics of the launch action.

**Returns**

The return value is a handle to a PDF action.

**Description**

Creates new "Launch" action.

# 14.8 Hide Action

A hide action hides or shows one or more annotations on the screen by setting or clearing their Hidden flags. This type of action can be used in combination with appearance streams and trigger events to display pop-up help information on the screen.

**Functions**

Function
⇒ PDFActionNewHide (📖 see page 109)
⇒ PDFActionHideAddAnnotation (📖 see page 110)
⇒ PDFActionHideAddAnnotationName (📖 see page 110)

**Legend**

⇒	Method
---	--------

# 14.8.1 PDFActionNewHide Function

```
LIB_API PDFActionHandle PDFActionNewHide(PDFDocHandle Doc, ppBool IsHide);
```

**File**

VSActionA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document handle.

ppBool IsHide	[in] Type of the action. Execution of this action will hide selected annotations if value sets in "true". In other case selected annotations will be shown.
---------------	---

**Returns**

The return value is a handle to a PDF action.

**Description**

Creates new "Hide" action.

---

## 14.8.2 PDFActionHideAddAnnotation Function

```
LIB_API void PDFActionHideAddAnnotation(PDFActionHandle Action, PDFAnnotationHandle Annotation);
```

**File**

VSActionA.h

**Parameters**

Parameters	Description
PDFActionHandle Action	[in] Handle of the PDF action.
PDFAnnotationHandle Annotation	[in] Handle of the annotation which is needed to be appended to list.
Doc	[in] PDF Document handle.

**Returns**

None.

**Description**

Appends annotation to list in the hide action.

**Notes**

Operation will be performed for all annotations in the PDF document if any annotation for this action is not selected.

---

## 14.8.3 PDFActionHideAddAnnotationName Function

```
LIB_API void PDFActionHideAddAnnotationName(PDFActionHandle Action, char * AnnotationName, ppInt32 Length);
```

**File**

VSActionA.h

**Parameters**

Parameters	Description
PDFActionHandle Action	[in] Handle of the PDF action.
char * AnnotationName	[in] Name of the annotation which is needed to be appended to the list.
ppInt32 Length	[in] Length of the name.
Doc	[in] PDF Document handle.

**Returns**

None.

Description

Appends annotation to list in the hide action.

Notes

Operation will be performed for all annotations in the PDF document if any annotation for this action is not selected.


# 14.9 URI Action

A uniform resource identifier (URI) is a string that identifies (resolves to) a resource on the Internet — typically a file that is the destination of a hypertext link, although it can also resolve to a query or other entity. A URI action causes a URI to be resolved.

Functions

Function
 PDFActionNewURI (🔗 see page 111)

Legend

	Method
---	--------

## 14.9.1 PDFActionNewURI Function

```
LIB_API PDFActionHandle PDFActionNewURI(PDFDocHandle Doc, char * URI, ppInt32 Length, ppBool IsMap);
```

File

VSActionA.h

Parameters

Parameters	Description
PDFDocHandle Doc	[in] PDF Document handle.
char * URI	[in] The uniform resource identifier to resolve, encoded in 7-bit ASCII.
ppInt32 Length	[in] Length of the URI.
ppBool IsMap	[in] A flag specifying whether to track the mouse position when the URI is resolved.

Returns

The return value is a handle to a PDF action.

Description

Creates new "URI" action.

# 14.10 Thread Action


A thread action jumps to a specified bead on an article, in either the current document or a different one.

## Functions

### Function

PDFActionNewThread (see page 112)

## Legend

	Method
---	--------

## 14.10.1 PDFActionNewThread Function

```
LIB_API PDFActionHandle PDFActionNewThread(PDFDocHandle Doc, char * FileName, ppInt32
FileNameLength, PDFThreadActionParam Thread);
```

### File

VSActionA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[in] PDF Document handle
char * FileName	[in] Filename of the PDF documents where destination thread may be the desired.
ppInt32 FileNameLength	[in] Length of the filename.
Launch	[in] Point to a <b>PDFThreadActionParam</b> (see page 161) structure that defines the characteristics of the thread action.

### Returns

The return value is a handle to a PDF action.

### Description

Creates new "Thread" action.

### Notes

Destination thread is in current PDF document if filename is NULL. In other case PDFBeadHandle or PDFThreadHandle impossible to use in **PDFThreadActionParam** (see page 161) structure.

## 14.11 Named Action

Viewer applications supports several named actions.

## Functions

### Function

PDFActionNewNamed (see page 112)

## Legend

	Method
---	--------

## 14.11.1 PDFActionNewNamed Function

```
LIB_API PDFActionHandle PDFActionNewNamed(PDFDocHandle Doc, PDFNamedActionType NamedType);
```

**File**

VSActionA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document handle.
PDFNamedActionType NamedType	[in] Operation for the named action.

**Returns**

The return value is a handle to a PDF action.

**Description**

Creates new "Named" action.

## 14.12 Import Action

An import-data action imports Forms Data Format (FDF) data into the document's interactive form from a specified file.

**Functions**

Function
 PDFActionNewImportData (  see page 113)

**Legend**

	Method
---	--------

### 14.12.1 PDFActionNewImportData Function

```
LIB_API PDFActionHandle PDFActionNewImportData(PDFDocHandle Doc, char * FileName, ppInt32 Length);
```

**File**

VSActionA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document handle.
char * FileName	[in] The FDF filename from which to import the data.
ppInt32 Length	[in] The length of the filename.

**Returns**

The return value is a handle to a PDF action.

**Description**

Creates new "Import Data" action.

## 14.13 Submit Action

A submit-form action transmits the names and values of selected interactive form fields to a specified uniform resource locator (URL), presumably the address of a World Wide Web server that will process them and send back a response.

### Functions

Function
PDFActionNewSubmitForm (see page 114)
PDFActionSubmitFormAddAnnotation (see page 115)
PDFActionSubmitFormAddAnnotationName (see page 115)

### Legend

	Method
---	--------

### 14.13.1 PDFActionNewSubmitForm Function

```
LIB_API PDFActionHandle PDFActionNewSubmitForm(PDFDocHandle Doc, char * URI, ppInt32 Length, ppInt32 Flags);
```

#### File

VSActionA.h

#### Parameters

Parameters	Description
PDFDocHandle Doc	[in] PDF Document handle.
char * URI	[in] A URL file specification giving the uniform resource locator of the script at the Web server that will process the submission.
ppInt32 Length	[in] Length of the URI string
Flag	[in] A set of flags specifying various characteristics of the action.

#### Returns

The return value is a handle to a PDF action.

#### Description

Creates new "SubmitForm" action.

#### Notes

Below you can find flag meaning table:

Flag	Meaning
PDF_SUBMIT_FORM_FLAG_EXCLUDE	The list of the acroform objects exclude from the submission.
PDF_SUBMIT_FORM_FLAG_INCLUDE_NO_VALUE_FIELDS	All acroform object will included in submission ( With empty values too )
PDF_SUBMIT_FORM_FLAG_EXPORT_FORMAT	If this flag set, export will execute in HTML form format, else in FDF format

PDF_SUBMIT_FORM_FLAG_GET_METHOD	If this flag set, field names and values are submitted using an HTTP GET request; if clear, they are submitted using a POST request.
PDF_SUBMIT_FORM_FLAG_SUBMIT_COORDINATES	If set, the coordinates of the mouse click that caused the submitform action are transmitted as part of the form data. The coordinate values are relative to the upper-left corner of the acroform object rectangle.
PDF_SUBMIT_FORM_FLAG_XML	If set, field names and values are submitted in XML format; if clear, they are submitted in HTML Form format or Forms Data Format (FDF), according to the value of the PDF_SUBMIT_FORM_FLAG_EXPORT_FORMAT flag.
PDF_SUBMIT_FORM_FLAG_SUBMIT_PDF	If set, the document is submitted in PDF format, using the MIME content type application/pdf (described in Internet RFC 2045, Multipurpose Internet Mail Extensions (MIME), Part One: Format of Internet Message Bodies; see the Bibliography). If this flag is set, all other flags are ignored except PDF_SUBMIT_FORM_FLAG_GET_METHOD.

### 14.13.2 PDFActionSubmitFormAddAnnotation Function

```
LIB_API void PDFActionSubmitFormAddAnnotation(PDFActionHandle Action, PDFAnnotationHandle Annotation);
```

File

VSActionA.h

Parameters

Parameters	Description
PDFActionHandle Action	[in] Handle of the PDF action.
PDFAnnotationHandle Annotation	[in] Handle of the annotation which need append to list.
Doc	[in] PDF Document handle.

Returns

None.

Description

Appends annotation to list in the submitform action.

Notes

Operation will be performed for all acroform object in the PDF document (flag PDF\_SUBMIT\_FORM\_FLAG\_EXCLUDE not used ) if its not selected any annotation for this action.

### 14.13.3 PDFActionSubmitFormAddAnnotationName Function

```
LIB_API void PDFActionSubmitFormAddAnnotationName(PDFActionHandle Action, char * AnnotationName, ppInt32 Length);
```

File

VSActionA.h

**Parameters**

Parameters	Description
PDFActionHandle Action	[in] Handle of the PDF action.
char * AnnotationName	[in] Name of the annotation which is needed to be appended to the list.
ppInt32 Length	[in] Length of the name.
Doc	[in] PDF Document handle.

**Returns**

None.

**Description**

Appends annotation to list in the submitform action.







**Notes**

Operation will be performed for all acroform object in the PDF document (flag PDF\_SUBMIT\_FORM\_FLAG\_EXCLUDE not used ) if its not selected any annotation for this action.


## 14.14 Reset Form Action

A reset-form action resets selected interactive form fields to their default values. For fields that can have no value (such as pushbuttons), the action has no effect.

**Functions**

Function
 PDFActionNewResetForm (  see page 116)
 PDFActionResetFormAddAnnotation (  see page 117)
 PDFActionResetFormAddAnnotationName (  see page 117)

**Legend**

	Method
---	--------

### 14.14.1 PDFActionNewResetForm Function

```
LIB_API PDFActionHandle PDFActionNewResetForm(PDFDocHandle Doc, ppBool Exclude);
```

**File**

VSActionA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] PDF Document handle.
ppBool Exclude	[in] If false, the list specifies which fields to reset. If true, the list of the acroform objects informs which fields to be excluded from resetting; all fields in the document's interactive form are reset excepting those listed.

**Returns**

The return value is a handle to a PDF action.



**Description**

Creates new "Reset" action.

---

## 14.14.2 PDFActionResetFormAddAnnotation Function

```
LIB_API void PDFActionResetFormAddAnnotation(PDFActionHandle Action, PDFAnnotationHandle Annotation);
```

**File**

VSActionA.h

**Parameters**

Parameters	Description
PDFActionHandle Action	[in] Handle of the PDF action.
PDFAnnotationHandle Annotation	[in] Handle of the annotation which is needed to be appended to list.
Doc	[in] PDF Document handle.

**Returns**

None.

**Description**

Appends annotation to list in the resetform action.

**Notes**

Reset action will be performed for all annotations in the PDF document if its not selected any annotation for this action.

---

## 14.14.3 PDFActionResetFormAddAnnotationName Function

```
LIB_API void PDFActionResetFormAddAnnotationName(PDFActionHandle Action, char * AnnotationName, ppInt32 Length);
```

**File**

VSActionA.h

**Parameters**

Parameters	Description
PDFActionHandle Action	[in] Handle of the PDF action.
char * AnnotationName	[in] Name of the annotation which is needed to be appended to the list.
ppInt32 Length	[in] Length of the name.
Doc	[in] PDF Document handle.

**Returns**

None.

**Description**

Appends annotation to list in the resetform action.





Notes

Reset action will be performed for all annotations in the PDF document if its not selected any annotation for this action.


# 14.15 Javascript Action

A JavaScript action causes a script to be compiled and executed by the JavaScript interpreter. Depending on the nature of the script, this can cause various interactive form fields in the document to update their values or change their visual appearances.

Functions

Function
 PDFActionNewJavaScriptStream (  see page 118)
 PDFActionNewJavaScript (  see page 118)

Legend

	Method
---	--------

## 14.15.1 PDFActionNewJavaScriptStream Function

```
LIB_API PDFActionHandle PDFActionNewJavaScriptStream(PDFDocHandle Doc, PDFCosHandle JavaScript);
```

File

VSActionA.h

Parameters

Parameters	Description
PDFDocHandle Doc	[in] PDF Document handle
PDFCosHandle JavaScript	[in] Cos Stream where this JavaScript is stored

Returns

The return value is a handle to a PDF action.

Description

Creates new "JavaScript" action from CosStream where this javascript is stored.

## 14.15.2 PDFActionNewJavaScript Function

```
LIB_API PDFActionHandle PDFActionNewJavaScript(PDFDocHandle Doc, char * JavaScript, ppInt32 Length);
```

File

VSActionA.h

Parameters

Parameters	Description
PDFDocHandle Doc	[in] PDF Document handle
char * JavaScript	[in] JavaScript string which will be executed

ppInt32 Length	[in] Length of javascript string.
----------------	-----------------------------------

**Returns**

The return value is a handle to a PDF action.

**Description**

Creates new "JavaScript" action from string.

# 15 CosObject Level

## 15.1 Common Cos Object Functions

Functions

Function
✦ CosGetType (see page 120)
✦ CosCopy (see page 120)
✦ CosFree (see page 121)
✦ CosGetFromDoc (see page 121)
✦ CosGetGeneration (see page 122)
✦ CosGetID (see page 122)
✦ CosIsIndirect (see page 122)

Legend

✦	Method
---	--------

### 15.1.1 CosGetType Function

```
LIB_API CosType CosGetType(PDFCosHandle CosObject);
```

File

VSCosA.h

Parameters

Parameters	Description
PDFCosHandle CosObject	[in] The object whose type is obtained.

Returns

The object's type.

Description

Gets an object's type.

### 15.1.2 CosCopy Function

```
LIB_API PDFCosHandle CosCopy(PDFDocHandle Doc, PDFCosHandle CosObject);
```

File

VSCosA.h

**Parameters**

Parameters	Description
PDFCosHandle CosObject	[in] The Cos object from which information will be received for copy.
PDFDoc	[in] Document for which need create new cos object.

**Returns**

New Cos object which has all information from source Cos object.

**Description**

Creates new Cos object and copies all data from source Cos object excluding indirect information.

---

## 15.1.3 CosFree Function

```
LIB_API void CosFree(PDFCosHandle CosObject);
```

**File**

VSCosA.h

**Parameters**

Parameters	Description
PDFCosHandle CosObject	[in] The object to free.

**Returns**

None.

**Description**

Gets free Cos object. If it's a composite object (array, dictionary or stream) :

- all the direct objects in it will be automatically destroyed
- the indirect objects in it will be not destroyed

---

## 15.1.4 CosGetFromDoc Function

```
LIB_API PDFCosHandle CosGetFromDoc(PDFDocHandle Doc, ppUns32 ID);
```

**File**

VSCosA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] The document from which Cos object will be loaded.
ppUns32 ID	[in] The index of the indirect Cos object which is to be returned.

**Returns**

Either Cos object or the null object returns if there is no object with this ID.

**Description**

Gets the indirect Cos object from document.

# 15.1.5 CosGetGeneration Function

```
LIB_API ppUns16 CosGetGeneration(PDFCosHandle CosObject);
```

**File**

VSCosA.h

**Parameters**

Parameters	Description
PDFCosHandle CosObject	[in] The indirect CosObj for which the generation number is obtained. A CosObj can be determined as indirect using CosIsIndirect (see page 122) function.

**Returns**

The generation number of CosObj.

**Description**

Gets the generation number of an indirect Cos object.

**See Also**

CosIsIndirect (see page 122) CosGetID (see page 122)

# 15.1.6 CosGetID Function

```
LIB_API ppUns32 CosGetID(PDFCosHandle CosObject);
```

**File**

VSCosA.h

**Parameters**

Parameters	Description
PDFCosHandle CosObject	[in] The indirect CosObj for which the ID is obtained. A CosObj can be determined as indirect using CosIsIndirect (see page 122) function.

**Returns**

The ID of CosObj.

**Description**

Gets the index for an indirect object.

**See Also**

CosIsIndirect (see page 122) CosGetGeneration (see page 122)

# 15.1.7 CosIsIndirect Function

```
LIB_API ppBool CosIsIndirect(PDFCosHandle CosObject);
```

**File**

VSCosA.h

Parameters

Parameters	Description
PDFCosHandle CosObject	[in] The object to test.

Returns

true if Cos Object is indirect, false if Cos Object is direct.

Description

Tests object if it's indirect or direct.

See Also

CosGetID (🔗 see page 122) CosGetGeneration (🔗 see page 122)

# 15.2 Cos Null Object

The null object has a type and value that are unequal to those of any other object. There is only one object of type null, denoted by the keyword null.

Functions

Function
🔗 CosNullNew (🔗 see page 123)

Legend

🔗	Method
---	--------

## 15.2.1 CosNullNew Function

```
LIB_API PDFCosHandle CosNullNew(PDFDocHandle Doc);
```

File

VSCosA.h

Parameters

Parameters	Description
PDFDocHandle Doc	[in] The document in which the null object is used.

Returns

The newly-created null Cos object.

Description

Creates a new direct null object.

See Also

CosFree (🔗 see page 121) CosGetType (🔗 see page 120)

# 15.3 Cos Boolean Object

Boolean object - PDF provides boolean objects identified by the keywords true and false. Boolean objects can be used as the values of array elements and dictionary entries.

Functions

Function
⇒ CosBoolGetValue (🔗 see page 124)
⇒ CosBoolNew (🔗 see page 124)
⇒ CosBoolSetValue (🔗 see page 125)

Legend

⇒	Method
---	--------

## 15.3.1 CosBoolGetValue Function

```
LIB_API ppBool CosBoolGetValue(PDFCosHandle CosObject);
```

File

VSCosA.h

Parameters

Parameters	Description
PDFCosHandle CosObject	[in] The boolean Cos object whose value is obtained.

Returns

Value of the boolean Cos object.

Description

Gets the value of the specified boolean object.

See Also

CosBoolNew (🔗 see page 124) CosBoolSetValue (🔗 see page 125)

## 15.3.2 CosBoolNew Function

```
LIB_API PDFCosHandle CosBoolNew(PDFDocHandle Doc, ppBool IsIndirect, ppBool Value);
```

File

VSCosA.h

Parameters

Parameters	Description
PDFDocHandle Doc	[in] The document in which the boolean is used.
ppBool IsIndirect	[in] If true, creates the boolean object as an indirect object.
ppBool Value	[in] The value which new boolean will have.



Returns

The newly-created boolean Cos object.

Description

Creates a new boolean object and sets the specified value.

See Also

CosFree (🔗 see page 121) CosGetType (🔗 see page 120) CosBoolGetValue (🔗 see page 124) CosBoolSetValue (🔗 see page 125)

### 15.3.3 CosBoolSetValue Function

```
LIB_API void CosBoolSetValue(PDFCosHandle CosObject, ppBool Value);
```

File

VSCosA.h

Parameters

Parameters	Description
PDFCosHandle CosObject	[in] The boolean Cos object whose value is assigned.
ppBool Value	[in] New value of the Cos boolean object.

Returns

None

Description

Sets the value of the specified boolean object.

See Also

CosBoolNew (🔗 see page 124) CosBoolGetValue (🔗 see page 124)

## 15.4 Cos Number Objects

PDF provides two types of numeric object: integer and real. Integer objects represent mathematical integers within a certain interval centered at 0. Real objects approximate mathematical real numbers, but with limited range and precision; they are typically represented in fixed-point, rather than floating-point, form. The range and precision of numbers are limited by the internal representations used in the machine on which the PDF viewer application is running.

Functions

Function
🔗 CosNumberGetValue (🔗 see page 129)

Legend

🔗	Method
---	--------

## 15.4.1 Cos Real Object

### Functions

Function
✦ CosRealGetValue (see page 126)
✦ CosRealNew (see page 126)
✦ CosRealSetValue (see page 127)

### Legend

✦	Method
---	--------

### 15.4.1.1 CosRealGetValue Function

```
LIB_API ppReal CosRealGetValue(PDFCosHandle CosObject);
```

#### File

VSCosA.h

#### Parameters

Parameters	Description
PDFCosHandle CosObject	[in] The real Cos object whose value is obtained.

#### Returns

Value of the real Cos object.

#### Description

Gets the value of the specified real object.

#### See Also

CosRealNew (see page 126) CosRealSetValue (see page 127)

### 15.4.1.2 CosRealNew Function

```
LIB_API PDFCosHandle CosRealNew(PDFDocHandle Doc, ppBool IsIndirect, ppReal Value);
```

#### File

VSCosA.h

#### Parameters

Parameters	Description
PDFDocHandle Doc	[in] The document in which the real is used.
ppBool IsIndirect	[in] If true, creates the real object as an indirect object.
ppReal Value	[in] The value the new real will have.

#### Returns

The newly-created real Cos object.

#### Description

Creates a new real object and sets the specified value.

See Also

[CosFree](#) (see page 121) [CosGetType](#) (see page 120) [CosRealGetValue](#) (see page 126) [CosRealSetValue](#) (see page 127)

15.4.1.3 CosRealSetValue Function

```
LIB_API void CosRealSetValue(PDFCosHandle CosObject, ppReal Value);
```

File

VSCosA.h

Parameters

Parameters	Description
PDFCosHandle CosObject	[in] The real Cos object whose value is assigned.
ppReal Value	[in] New value of the real Cos object.

Returns

None

Description




Sets the value of the specified real object.

See Also


[CosRealNew](#) (see page 126) [CosRealGetValue](#) (see page 126)

15.4.2 Cos Integer Object

Functions

Function
 <a href="#">CosIntGetValue</a> (see page 127)
 <a href="#">CosIntNew</a> (see page 128)
 <a href="#">CosIntSetValue</a> (see page 128)

Legend

	Method
---	--------

15.4.2.1 CosIntGetValue Function

```
LIB_API ppInt32 CosIntGetValue(PDFCosHandle CosObject);
```

File

VSCosA.h

Parameters

Parameters	Description
PDFCosHandle CosObject	[in] The integer Cos object whose value is obtained.

Returns

Value of the integer Cos object.

**Description**

Gets the value of the specified integer object.

**See Also**

CosIntNew (🔗 see page 128) CosIntSetValue (🔗 see page 128)

# 15.4.2.2 CosIntNew Function

```
LIB_API PDFCosHandle CosIntNew(PDFDocHandle Doc, ppBool IsIndirect, ppInt32 Value);
```

**File**

VSCosA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] The document in which the integer is used.
ppBool IsIndirect	[in] If true, creates the integer object as an indirect object.
ppInt32 Value	[in] The value the new integer will have.

**Returns**

The newly-created integer Cos object.

**Description**

Creates a new integer object and sets the specified value.

**See Also**

CosFree (🔗 see page 121) CosGetType (🔗 see page 120) CosIntGetValue (🔗 see page 127) CosIntSetValue (🔗 see page 128)

# 15.4.2.3 CosIntSetValue Function

```
LIB_API void CosIntSetValue(PDFCosHandle CosObject, ppInt32 Value);
```

**File**

VSCosA.h

**Parameters**

Parameters	Description
PDFCosHandle CosObject	[in] The integer Cos object whose value is assigned.
ppInt32 Value	[in] New value of the integer Cos object.

**Returns**

None

**Description**

Sets the value of the specified integer object.

**See Also**

CosIntNew (🔗 see page 128) CosIntGetValue (🔗 see page 127)

# 15.4.3 CosNumberGetValue Function

```
LIB_API ppReal CosNumberGetValue(PDFCosHandle CosObject);
```

**File**

VSCosA.h

**Parameters**

Parameters	Description
PDFCosHandle CosObject	[in] The real or integer Cos object whose value is obtained.

**Returns**

Value of the real or integer Cos object.




**Description**

Gets the value of the specified real or integer object.


# 15.5 Cos Name Object

A name object is an atomic symbol uniquely defined by a sequence of characters. Uniquely defined means that any two name objects made up of the same sequence of characters are identically the same object. Atomic means that a name has no internal structure; although it is defined by a sequence of characters, those characters are not “elements” of the name.

**Functions**

Function
 CosNameGetValue (see page 129)
 CosNameNew (see page 130)
 CosNameSetValue (see page 130)

**Legend**

	Method
---	--------

# 15.5.1 CosNameGetValue Function

```
LIB_API ppAtom CosNameGetValue(PDFCosHandle CosObject);
```

**File**

VSCosA.h

**Parameters**

Parameters	Description
PDFCosHandle CosObject	[in] The name Cos object whose value is obtained.

**Returns**

Value of the name Cos object.

**Description**

Gets the value of the specified name object.

See Also

CosNameNew (↗ see page 130) CosNameSetValue (↗ see page 130)

# 15.5.2 CosNameNew Function

```
LIB_API PDFCosHandle CosNameNew(PDFDocHandle Doc, ppBool IsIndirect, ppAtom Value);
```

File

VSCosA.h

Parameters

Parameters	Description
PDFDocHandle Doc	[in] The document in which the name object is used.
ppBool IsIndirect	[in] If true, creates the name object as an indirect object.
ppAtom Value	[in] The value the new name will have.

Returns

The newly-created name Cos object.

Description

Creates a new name object and sets the specified value.

See Also

CosFree (↗ see page 121) CosGetType (↗ see page 120) CosNameGetValue (↗ see page 129) CosNameSetValue (↗ see page 130)

# 15.5.3 CosNameSetValue Function

```
LIB_API void CosNameSetValue(PDFCosHandle CosObject, ppAtom Value);
```

File

VSCosA.h

Parameters

Parameters	Description
PDFCosHandle CosObject	[in] The boolean Cos object whose value is assigned.
ppAtom Value	[in] New value of the name Cos object.

Returns

None

Description

Sets the value of the specified name object.

See Also

CosNameNew (↗ see page 130) CosNameGetValue (↗ see page 129)

# 15.6 Cos String Object

A string object consists of a series of bytes—unsigned integer values in the range 0 to 255. The string elements are not integer objects, but are stored in a more compact format. The length of a string is subject to an implementation limit.

Functions

Function
✦ CosStringGetValue (🔗 see page 131)
✦ CosStringNew (🔗 see page 131)
✦ CosStringSetValue (🔗 see page 132)

Legend

✦	Method
---	--------

## 15.6.1 CosStringGetValue Function

```
LIB_API char * CosStringGetValue(PDFCosHandle CosObject, ppUns32 * Length);
```

File

VSCosA.h

Parameters

Parameters	Description
PDFCosHandle CosObject	[in] The string Cos object whose value is obtained.
ppUns32 * Length	[out] Length of the value in bytes.

Returns

The value of string Cos object.

Description

Gets the value of string Cos object and the string's length.

See Also

CosStringNew (🔗 see page 131) CosStringSetValue (🔗 see page 132)

## 15.6.2 CosStringNew Function

```
LIB_API PDFCosHandle CosStringNew(PDFDocHandle Doc, ppBool IsIndirect, char * String, ppUns32 Length);
```

File

VSCosA.h

Parameters

Parameters	Description
PDFDocHandle Doc	[in] The document in which the string is used.
ppBool IsIndirect	[in] If true, creates the string object as an indirect object.

char * String	[in] The value that the new string will have. It is not a C string, since Cos strings can contain NULL characters. The data in String is copied, that is, if String was dynamically allocated, it can be free after this call.
ppUns32 Length	[in] The length of String.

Returns

The newly-created string Cos object.

Description

Creates a new string object and sets the specified value.

See Also

CosFree (🔗 see page 121) CosGetType (🔗 see page 120) CosStringGetValue (🔗 see page 131) CosStringSetValue (🔗 see page 132)

### 15.6.3 CosStringSetValue Function

```
LIB_API void CosStringSetValue(PDFCosHandle CosObject, char * String, ppUns32 Length);
```

File

VSCosA.h

Parameters

Parameters	Description
PDFCosHandle CosObject	[in] The string Cos object whose value is assigned.
char * String	[in] The new value that the string Cos object will have. It is not a C string, since Cos strings can contain NULL characters. The data in String is copied, that is, if String was dynamically allocated, it can be free after this call.
ppUns32 Length	[in] The new length of String.

Returns

None.

Description

Sets the new value for string Cos object.

See Also

CosStringNew (🔗 see page 131) CosStringGetValue (🔗 see page 131)







## 15.7 Cos Array Object

An array object is a one-dimensional collection of objects arranged sequentially. Unlike arrays in many other computer languages, PDF arrays may be heterogeneous; that is, an array’s elements may be any combination of numbers, strings, dictionaries, or any other objects, including other arrays. The number of elements in an array is subject to an implementation limit.


Functions

Function
🔗 CosArrayAppend (🔗 see page 133)



 <a href="#">CosArrayClear</a> ( <a href="#">see page 133</a> )
 <a href="#">CosArrayCount</a> ( <a href="#">see page 134</a> )
 <a href="#">CosArrayInsert</a> ( <a href="#">see page 134</a> )
 <a href="#">CosArrayItem</a> ( <a href="#">see page 135</a> )
 <a href="#">CosArrayNew</a> ( <a href="#">see page 135</a> )
 <a href="#">CosArrayRemove</a> ( <a href="#">see page 136</a> )

Legend

	Method
---	--------

# 15.7.1 CosArrayAppend Function

```
LIB_API ppInt32 CosArrayAppend(PDFCosHandle CosObject, PDFCosHandle NewCosObject);
```

File

VSCosA.h

Parameters

Parameters	Description
PDFCosHandle CosObject	[in] The array into which the object is appended.
PDFCosHandle NewCosObject	[in] The object to append.

Returns

Position in which Cos object was inserted.

Description

Appends an cos object into an array.

See Also

[CosArrayNew](#) ([see page 135](#)) [CosArrayCount](#) ([see page 134](#)) [CosArrayInsert](#) ([see page 134](#)) [CosArrayAppend](#) [CosArrayRemove](#) ([see page 136](#)) [CosArrayClear](#) ([see page 133](#))

# 15.7.2 CosArrayClear Function

```
LIB_API void CosArrayClear(PDFCosHandle CosObject);
```

File

VSCosA.h

Parameters

Parameters	Description
PDFCosHandle CosObject	[in] The array from which an elements are removed.

Returns

None

Description

Clears and gets free all elements from an array.

See Also

[CosArrayNew](#) ([see page 135](#)) [CosArrayCount](#) ([see page 134](#)) [CosArrayInsert](#) ([see page 134](#)) [CosArrayAppend](#) ([see page 134](#))

see page 133) CosArrayRemove (↗ see page 136) CosArrayClear

### 15.7.3 CosArrayCount Function

```
LIB_API ppUns32 CosArrayCount(PDFCosHandle CosObject);
```

**File**

VSCosA.h

**Parameters**

Parameters	Description
PDFCosHandle CosObject	[in] The array for which the number of elements are determined.

**Returns**

The number of elements in array.

**Description**

Gets the number of elements in array.

**See Also**

CosArrayNew (↗ see page 135) CosArrayInsert (↗ see page 134) CosArrayAppend (↗ see page 133) CosArrayRemove (↗ see page 136) CosArrayClear (↗ see page 133)

### 15.7.4 CosArrayInsert Function

```
LIB_API ppInt32 CosArrayInsert(PDFCosHandle CosObject, PDFCosHandle NewCosObject, ppUns32 pos);
```

**File**

VSCosA.h

**Parameters**

Parameters	Description
PDFCosHandle CosObject	[in] The array into which the object is inserted.
PDFCosHandle NewCosObject	[in] The object to insert.
Position	[in] The location in the array to insert the cos object. The cos object is inserted before the specified location. The first element in an array has a pos of zero. If pos >= CosArrayCount (↗ see page 134) ( CosObject ), it appends obj to array (increasing the array count by 1).

**Returns**

Position in which Cos object was inserted.

**Description**

Inserts an cos object into an array.

**See Also**

CosArrayNew (↗ see page 135) CosArrayCount (↗ see page 134) CosArrayInsert CosArrayAppend (↗ see page 133) CosArrayRemove (↗ see page 136) CosArrayClear (↗ see page 133)

---

## 15.7.5 CosArrayItem Function

```
LIB_API PDFCosHandle CosArrayItem(PDFCosHandle CosObject, ppUns32 Index);
```

### File

VSCosA.h

### Parameters

Parameters	Description
PDFCosHandle CosObject	[in] The array from which an element is obtained.
ppUns32 Index	[in] The Index for the array member to obtain. Array indices start at 0.

### Returns

The Cos object occupying the index element of array. Returns a null Cos object if Index is outside the array bounds. If specified element is referenced Cos object function returns Cos object with ID equal to value of referenced Cos object.

### Description

Gets the specified element from an array.

### See Also

CosArrayNew ([↗](#) see page 135) CosArrayCount ([↗](#) see page 134) CosArrayInsert ([↗](#) see page 134) CosArrayAppend ([↗](#) see page 133) CosArrayRemove ([↗](#) see page 136) CosArrayClear ([↗](#) see page 133)

---

## 15.7.6 CosArrayNew Function

```
LIB_API PDFCosHandle CosArrayNew(PDFDocHandle Doc, ppBool IsIndirect, ppUns32 Entries);
```

### File

VSCosA.h

### Parameters

Parameters	Description
PDFDocHandle Doc	[in] The document in which the array is used.
ppBool IsIndirect	[in] If true, creates the array object as an indirect object.
ppUns32 Entries	[in] The number of elements that will be in the array. This value only a hint; Cos arrays grow dynamically as needed.

### Returns

The newly-created array Cos object.

### Description

Creates and returns a new array Cos object.

### See Also

CosFree ([↗](#) see page 121) CosGetType ([↗](#) see page 120) CosArrayCount ([↗](#) see page 134) CosArrayInsert ([↗](#) see page 134) CosArrayAppend ([↗](#) see page 133) CosArrayRemove ([↗](#) see page 136) CosArrayClear ([↗](#) see page 133)

# 15.7.7 CosArrayRemove Function

Removes element from array.

```
LIB_API void CosArrayRemove(PDFCosHandle CosObject, ppUns32 Index);
```

File

VSCosA.h

Parameters

Parameters	Description
PDFCosHandle CosObject	[in] The array Cos object to remove the member from it.
ppUns32 Index	[in] The Index for the array member to remove. Array indices start at 0.

Returns

None

Description

Checks whether the position is within the array bounds and then removes it from the array and moves each subsequent element to the slot with the next smaller Index and decrements the array's length by 1. Removed element will be free.

See Also

CosArrayNew (🔗 see page 135) CosArrayCount (🔗 see page 134) CosArrayInsert (🔗 see page 134) CosArrayAppend (🔗 see page 133) CosArrayRemove CosArrayClear (🔗 see page 133)

# 15.8 Cos Dictionary Object

A dictionary object is an associative table containing pairs of objects, known as the dictionary's entries. The first element of each entry is the key and the second element is the value. The key must be a name (unlike dictionary keys in Post-Script, which may be objects of any type). The value can be any kind of object, including another dictionary. A dictionary entry whose value is null is equivalent to an absent entry.

Functions

Function
🔗 CosDictAppend (🔗 see page 137)
🔗 CosDictClear (🔗 see page 137)
🔗 CosDictCount (🔗 see page 137)
🔗 CosDictRemoveKey (🔗 see page 138)
🔗 CosDictValueByName (🔗 see page 138)
🔗 CosDictGetPair (🔗 see page 139)
🔗 CosDictNew (🔗 see page 139)

Legend

🔗	Method
---	--------

---

## 15.8.1 CosDictAppend Function

```
LIB_API void CosDictAppend(PDFCosHandle CosObject, ppAtom Key, PDFCosHandle KeyValue);
```

### File

VSCosA.h

### Parameters

Parameters	Description
PDFCosHandle CosObject	[in] The dictionary or stream in which a value is set.
ppAtom Key	[in] The key which value is set.
Value	[in] The value to set.

### Returns

None

### Description

Sets the value of a dictionary key, adding the key to the dictionary. This method can also be used with a stream object. In that case, the key-value pair is added to the stream's attributes dictionary.

### See Also

CosDictNew ([see page 139](#)) CosDictCount ([see page 137](#)) CosDictGetPair ([see page 139](#)) CosDictRemoveKey ([see page 138](#)) CosDictValueByName ([see page 138](#)) CosDictClear ([see page 137](#))

---

## 15.8.2 CosDictClear Function

```
LIB_API void CosDictClear(PDFCosHandle CosObject);
```

### File

VSCosA.h

### Parameters

Parameters	Description
PDFCosHandle CosObject	[in] The dictionary or stream from which elements are removed.

### Returns

None

### Description

Clears and gets free all keys and values from the dictionary or stream.

### See Also

CosDictNew ([see page 139](#)) CosDictCount ([see page 137](#)) CosDictGetPair ([see page 139](#)) CosDictAppend ([see page 137](#)) CosDictRemoveKey ([see page 138](#)) CosDictValueByName ([see page 138](#))

---

## 15.8.3 CosDictCount Function

```
LIB_API ppInt32 CosDictCount(PDFCosHandle CosObject);
```

**File**

VSCosA.h

**Parameters**

Parameters	Description
PDFCosHandle CosObject	[in] The dictionary or stream for which the number of key-value pair is determined.

**Returns**

The number of key-value pair in the dictionary.

**Description**

Gets the number of key-value pair in the dictionary. This method can also be used with a stream object. In that case, returns number the key-value pair from the stream's attributes dictionary.

**See Also**

CosDictNew ([↗](#) see page 139) CosDictGetPair ([↗](#) see page 139) CosDictAppend ([↗](#) see page 137) CosDictRemoveKey ([↗](#) see page 138) CosDictValueByName ([↗](#) see page 138) CosDictClear ([↗](#) see page 137)

---

## 15.8.4 CosDictRemoveKey Function

```
LIB_API void CosDictRemoveKey(PDFCosHandle CosObject, ppAtom Key);
```

**File**

VSCosA.h

**Parameters**

Parameters	Description
PDFCosHandle CosObject	[in] The dictionary from which the key-value pair is removed.
ppAtom Key	[in] The key to remove.

**Returns**

None

**Description**

Removes and gets free a key-value pair from a dictionary. This method can also be used with a stream object. In that case, the key-value pair is removed from the stream's attributes dictionary.

**See Also**

CosDictNew ([↗](#) see page 139) CosDictCount ([↗](#) see page 137) CosDictGetPair ([↗](#) see page 139) CosDictAppend ([↗](#) see page 137) CosDictValueByName ([↗](#) see page 138) CosDictClear ([↗](#) see page 137)

---

## 15.8.5 CosDictValueByName Function

```
LIB_API PDFCosHandle CosDictValueByName(PDFCosHandle CosObject, ppAtom Key);
```

**File**

VSCosA.h

**Parameters**

Parameters	Description
PDFCosHandle CosObject	[in] The dictionary or stream from which a value is obtained.
ppAtom Key	[in] The key whose value is obtained.

**Returns**

The object associated with the specified key. Returns a null Cos object if key is not present. If value is referenced Cos object returns Cos object with ID equal to value of referenced Cos object.

**Description**

Gets the value of the specified key in the specified dictionary. If it's called with a stream object instead of a dictionary object, this method gets the value of the specified key from the stream's attributes dictionary.

**See Also**

CosDictNew ([↗](#) see page 139) CosDictCount ([↗](#) see page 137) CosDictGetPair ([↗](#) see page 139) CosDictAppend ([↗](#) see page 137) CosDictRemoveKey ([↗](#) see page 138) CosDictClear ([↗](#) see page 137)

**Example**

```
PDFCosHandle dict, obj;  
obj = CosDicValueByName ( dict, ULStringToAtom ( Lib, "Pages" ) );
```

---

## 15.8.6 CosDictGetPair Function

```
LIB_API void CosDictGetPair(PDFCosHandle CosObject, ppUns32 Index, ppAtom * Key,  
PDFCosHandle * Value);
```

**File**

VSCosA.h

**Parameters**

Parameters	Description
PDFCosHandle CosObject	[in] The dictionary or stream for which the key-value pair is determined.
ppUns32 Index	[in] Index of the pair for which is needed to obtain key and value.
ppAtom * Key	[out] Key from pair.
PDFCosHandle * Value	[out] Value from pair.

**Description**

Gets the key-value pair in the dictionary. This method can also be used with a stream object. In that case, returns the key-value pair from the stream's attributes dictionary.

**See Also**

CosDictNew ([↗](#) see page 139) CosDictCount ([↗](#) see page 137) CosDictAppend ([↗](#) see page 137) CosDictRemoveKey ([↗](#) see page 138) CosDictValueByName ([↗](#) see page 138) CosDictClear ([↗](#) see page 137)

---

## 15.8.7 CosDictNew Function

```
LIB_API PDFCosHandle CosDictNew(PDFDocHandle Doc, ppBool IsIndirect, ppUns32 Entries);
```

**File**

VSCosA.h

**Parameters**

Parameters	Description
PDFDocHandle Doc	[in] The document in which the dictionary is used.
ppBool IsIndirect	[in] If true, creates the dictionary object as an indirect object.
ppUns32 Entries	[in] Number of entries in the dictionary. This value is only a hint - Cos dictionaries grow dynamically as needed.

**Returns**

The newly-created dictionary Cos object.

**Description**

Creates a new dictionary.

**See Also**

CosGetType (see page 120) CosFree (see page 121) CosDictCount (see page 137) CosDictGetPair (see page 139) CosDictAppend (see page 137) CosDictRemoveKey (see page 138) CosDictValueByName (see page 138) CosDictClear (see page 137)

## 15.9 Cos Stream Object

A stream object, like a string object, is a sequence of bytes. However, a PDF application can read a stream incrementally, while a string must be read in its entirety. Furthermore, a stream can be of unlimited length, whereas a string is subject to an implementation limit. For this reason, objects with potentially large amounts of data, such as images and page descriptions, are represented as streams.

**Functions**

Function
✦ CosStreamGetValue (see page 140)
✦ CosStreamGetAttr (see page 141)
✦ CosStreamNew (see page 141)

**Legend**

✦	Method
---	--------

### 15.9.1 CosStreamGetValue Function

```
LIB_API PDFStreamHandle CosStreamGetValue(PDFCosHandle CosObject);
```

**File**

VSCosA.h

**Parameters**

Parameters	Description
PDFCosHandle CosObject	[in] The stream whose attributes value is obtained.

**Returns**

The value of the stream Cos object.

**Description**

Gets a stream's value.



See Also

CosStreamNew (🔗 see page 141) CosStreamGetAttr (🔗 see page 141)

# 15.9.2 CosStreamGetAttr Function

```
LIB_API PDFCosHandle CosStreamGetAttr(PDFCosHandle CosObject);
```

File

VSCosA.h

Parameters

Parameters	Description
PDFCosHandle CosObject	[in] The stream whose attributes dictionary is obtained.

Returns

The stream's attributes dictionary Cos object.

Description

Gets a stream's attributes dictionary.

See Also

CosStreamNew (🔗 see page 141) CosStreamGetValue (🔗 see page 140)

# 15.9.3 CosStreamNew Function

```
LIB_API PDFCosHandle CosStreamNew(PDFDocHandle Doc, ppBool IsIndirect, ppInt32 Entries);
```

File

VSCosA.h

Parameters

Parameters	Description
PDFDocHandle Doc	[in] The document in which the dictionary is used.
ppBool IsIndirect	[in] Must always be true, specifying that the Cos stream is created as an indirect object.
ppInt32 Entries	[in] Number of entries in the attribute dictionary. This value is only a hint - Cos dictionaries grow dynamically as needed.

Returns

The newly-created stream Cos object.

Description

Creates a new stream.

See Also

CosFree (🔗 see page 121) CosGetType (🔗 see page 120) CosSteamGetAttr CosStreamGetValue (🔗 see page 140)

# 16 Underline Level

There are work functions with basic objects such as Color, Atoms, Files and Streams in this level. They are used to convert these objects from usual types to PDF objects or structures.

## 16.1 Color Level

### Functions

Function
⇒ ULCMYKToColor (🔗 see page 142)
⇒ ULGrayToColor (🔗 see page 142)
⇒ ULRGBToColor (🔗 see page 143)

### Legend

⇒	Method
---	--------

### 16.1.1 ULCMYKToColor Function

```
LIB_API TPDFColor ULCMYKToColor(ppReal c, ppReal m, ppReal y, ppReal k);
```

#### File

VSCanvasA.h

#### Parameters

Parameters	Description
ppReal c	[in] Specifies the intensity of the cyan color.
ppReal m	[in] Specifies the intensity of the magenta color.
ppReal y	[in] Specifies the intensity of the yellow color.
ppReal k	[in] Specifies the intensity of the black color.

#### Returns

The return value is the resultant CMYK color.

#### Description

Creates a TPDFColor (🔗 see page 172) structure from intensity of the CMYK.

#### Remarks

The intensity for each argument is in the range 0 through 1. If all four intensities are zero, the result is white. If all four intensities are 1, the result is black.

### 16.1.2 ULGrayToColor Function

```
LIB_API TPDFColor ULGrayToColor(ppReal g);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
ppReal g	[in] Specifies the intensity of the gray color.

**Returns**

The return value is the resultant gray color.

**Description**

Creates a TPDFColor (see page 172) structure from intensity of the gray.

**Remarks**

The intensity for argument is in the range 0 through 1. If intensity are zero, the result is black. If intensity are 1, the result is white.

---

## 16.1.3 ULRGBToColor Function

```
LIB_API TPDFColor ULRGBToColor(ppReal r, ppReal g, ppReal b);
```

**File**

VSCanvasA.h

**Parameters**

Parameters	Description
ppReal r	[in] Specifies the intensity of the red color.
ppReal g	[in] Specifies the intensity of the green color.
ppReal b	[in] Specifies the intensity of the blue color.

**Returns**

The return value is the resultant RGB color.

**Description**

Creates a TPDFColor (see page 172) structure from triple of the values.

**Remarks**

The intensity for each argument is in the range 0 through 1. If all three intensities are zero, the result is black. If all three intensities are 1, the result is white.





---

## 16.2 Atom Level


There are objects which characteristics are identified by names in PDF document. There is namespace in the library to simplify work with names. So, each atom defines unique name in document's namespace. There are functions for converting atoms to names and names to atoms, for namespace clearing, receiving count of the atoms in namespace and checking on existence name in namespace.

**Functions**

Function
ULAtomToString (see page 144)

 <a href="#">ULClearAtoms</a> ( <a href="#">see page 144</a> )
 <a href="#">ULExistsAtomForString</a> ( <a href="#">see page 144</a> )
 <a href="#">ULGetAtomCount</a> ( <a href="#">see page 145</a> )
 <a href="#">ULStringToAtom</a> ( <a href="#">see page 145</a> )

Legend

	Method
---	--------

## 16.2.1 ULAtomToString Function

```
LIB_API char * ULAtomToString(PDFLibHandle Lib, ppAtom Atom);
```

File

VSBaseA.h

Parameters

Parameters	Description
PDFLibHandle Lib	[in] PDF Library Handle.
ppAtom Atom	[in] Atom Key.

Returns

Text String Name in PDF Library

Description

Gets Text String Name by Atom Key in PDF Library.

See Also

[ULStringToAtom](#) ([see page 145](#))

## 16.2.2 ULClearAtoms Function

```
LIB_API void ULClearAtoms(PDFLibHandle Lib);
```

File

VSBaseA.h

Parameters

Parameters	Description
PDFLibHandle Lib	[in] PDF Library Handle.

Returns

None.

Description

Clears atoms in PDF Library. Gets free namespace.

## 16.2.3 ULEExistsAtomForString Function

```
LIB_API ppBool ULEExistsAtomForString(PDFLibHandle Lib, char * String);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[in] PDF Library Handle.
char * String	[in] Text String Name.

**Returns**

Boolean : true - exists, false - name not found.

**Description**

Tests if atom exists in PDF Library for searching text string.

**See Also**ULStringToAtom ([↗](#) see page 145)

---

## 16.2.4 ULGetAtomCount Function

```
LIB_API ppInt32 ULGetAtomCount(PDFLibHandle Lib);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[in] PDF Library Handle.

**Returns**

Atom count in PDF Library.

**Description**

Gets atom count in PDF Library.

---

## 16.2.5 ULStringToAtom Function

```
LIB_API ppAtom ULStringToAtom(PDFLibHandle Lib, char * String);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[in] PDF Library Handle.
char * String	[in] Text String Name.

**Returns**

Atom Key. Name Index in PDF Library

**Description**

Gets atom key by String in PDF Library.

See Also

ULAtomToString (🔗 see page 144)

# 16.3 File Level

There are functions for work with file, allowing to get data from files and to write data to file in our library.

Functions

Function
🔗 ULFileClose (🔗 see page 146)
🔗 ULFileGetChar (🔗 see page 146)
🔗 ULFileGetPosition (🔗 see page 147)
🔗 ULFileGetSize (🔗 see page 147)
🔗 ULFileLookChar (🔗 see page 148)
🔗 ULFileOpen (🔗 see page 148)
🔗 ULFileRead (🔗 see page 148)
🔗 ULFileSetPosition (🔗 see page 149)
🔗 ULFileWrite (🔗 see page 149)

Legend

🔗	Method
---	--------

## 16.3.1 ULFileClose Function

```
LIB_API void ULFileClose(PDFFileHandle FileHandle);
```

File

VSBaseA.h

Parameters

Parameters	Description
PDFFileHandle FileHandle	[in] PDF File Handle.

Returns

None.

Description

Closes PDF File.

See Also

ULFileOpen (🔗 see page 148)

## 16.3.2 ULFileGetChar Function

```
LIB_API ppInt32 ULFileGetChar(PDFFileHandle FileHandle);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFFileHandle FileHandle	[in] PDF File Handle.

**Returns**

One character from file. If it returns -1 than it is EOF ( end of file ).

**Description**

Gets one character from file.

---

## 16.3.3 ULFileGetPosition Function

```
LIB_API ppUns32 ULFileGetPosition(PDFFileHandle FileHandle);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFFileHandle FileHandle	[in] PDF File Handle.

**Returns**

File offset in bytes.

**Description**

Gets file cursor position ( byte offset from beginning of the file ).

**See Also**

ULFileSetPosition ( see page 149)

---

## 16.3.4 ULFileGetSize Function

```
LIB_API ppUns32 ULFileGetSize(PDFFileHandle FileHandle);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFFileHandle FileHandle	[in] PDF File Handle.

**Returns**

File size in bytes.

**Description**

Gets file size in bytes.

---

## 16.3.5 ULFileLookChar Function

```
LIB_API ppInt32 ULFileLookChar(PDFFileHandle FileHandle);
```

### File

VSBaseA.h

### Parameters

Parameters	Description
PDFFileHandle FileHandle	[in] PDF File Handle.

### Returns

One character from file. If it returns -1 than it is EOF ( end of file ).

### Description

Gets one character from file. Same as ULFileGetChar ( see page 146), only file cursor stays on that place.

---

## 16.3.6 ULFileOpen Function

```
LIB_API PDFFileHandle ULFileOpen(PDFLibHandle Lib, char * FileName, ppFileOpenMode OpenMode);
```

### File

VSBaseA.h

### Parameters

Parameters	Description
PDFLibHandle Lib	[in] PDF Library Handle.
char * FileName	[in] Filename, text string.
ppFileOpenMode OpenMode	[in] Open Mode : read or write.

### Returns

PDF File Handle.

### Description

Opens file and returns PDF File Handle.

### See Also

ULFileClose ( see page 146), ppFileOpenMode ( see page 165)

---

## 16.3.7 ULFileRead Function

```
LIB_API ppUns32 ULFileRead(PDFFileHandle FileHandle, void * Buffer, ppUns32 Length);
```

### File

VSBaseA.h



**Parameters**

Parameters	Description
PDFFileHandle FileHandle	[in] PDF File Handle.
void * Buffer	[in] Destination buffer in memory for data.
ppUns32 Length	[in] Size of read block in bytes.

**Returns**

Size of real read block in bytes.

**Description**

Reads data from file to buffer. Length is in bytes.

**See Also**

ULFileWrite ([↗](#) see page 149)

---

## 16.3.8 ULFileSetPosition Function

```
LIB_API ppUns32 ULFileSetPosition(PDFFileHandle FileHandle, ppUns32 Position);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFFileHandle FileHandle	[in] PDF File Handle.
ppUns32 Position	[in] File offset in bytes.

**Returns**

Position which is set.

**Description**

Sets file cursor to position ( byte offset ).

**See Also**

ULFileGetPosition ([↗](#) see page 147)

---

## 16.3.9 ULFileWrite Function

```
LIB_API ppUns32 ULFileWrite(PDFFileHandle FileHandle, void * Buffer, ppUns32 Length);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFFileHandle FileHandle	[in] PDF File Handle.
void * Buffer	[in] Source data buffer in memory.
ppUns32 Length	[in] Size of write block in bytes.

**Returns**

Size of real write block in bytes.

**Description**

Writes data from buffer to file. Length is in bytes.

**See Also**

ULFileRead ([↗](#) see page 148)

## 16.4 Stream Level

There are streams for work with data in our library. Streams are just ways of reading and writing data. Steams provide a common interface for reading and writing to different media such as memory, files and other.

**Functions**

Function
↗ ULStreamClear ( <a href="#">↗</a> see page 150)
↗ ULStreamClose ( <a href="#">↗</a> see page 151)
↗ ULStreamCopyToStream ( <a href="#">↗</a> see page 151)
↗ ULStreamCustomNew ( <a href="#">↗</a> see page 151)
↗ ULStreamFileHandleNew ( <a href="#">↗</a> see page 152)
↗ ULStreamFileNew ( <a href="#">↗</a> see page 152)
↗ ULStreamGetPosition ( <a href="#">↗</a> see page 153)
↗ ULStreamGetSize ( <a href="#">↗</a> see page 153)
↗ ULStreamLookChar ( <a href="#">↗</a> see page 153)
↗ ULStreamMemNew ( <a href="#">↗</a> see page 154)
↗ ULStreamMemReadOnlyNew ( <a href="#">↗</a> see page 154)
↗ ULStreamReadBuffer ( <a href="#">↗</a> see page 155)
↗ ULStreamReadChar ( <a href="#">↗</a> see page 155)
↗ ULStreamReadLine ( <a href="#">↗</a> see page 156)
↗ ULStreamSetPosition ( <a href="#">↗</a> see page 156)
↗ ULStreamSetSize ( <a href="#">↗</a> see page 156)
↗ ULStreamWriteBuffer ( <a href="#">↗</a> see page 157)
↗ ULStreamWriteChar ( <a href="#">↗</a> see page 157)

**Legend**

↗	Method
---	--------

### 16.4.1 ULStreamClear Function

```
LIB_API void ULStreamClear(PDFStreamHandle Stream, ppUns32 Size);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFStreamHandle Stream	[in] PDF Stream Handle.
ppUns32 Size	[in] Initializing size of PDF Stream in bytes.

**Returns**

None.

**Description**

Clears PDF Stream with initializing size ( maybe zero ).

---

## 16.4.2 ULStreamClose Function

```
LIB_API void ULStreamClose(PDFStreamHandle Stream);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFStreamHandle Stream	[in] PDF Stream Handle.

**Returns**

None.

**Description**

Closes PDF Stream.

**See Also**

ULStreamMemNew ( see page 154), ULStreamFileNew ( see page 152)

---

## 16.4.3 ULStreamCopyToStream Function

```
LIB_API ppUns32 ULStreamCopyToStream(PDFStreamHandle FromStream, PDFStreamHandle ToStream);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFStreamHandle FromStream	[in] PDF Stream Handle.
PDFStreamHandle ToStream	[in] PDF Stream Handle.

**Returns**

Size of bytes which is copied.

**Description**

Copies from one Stream in another Stream.

---

## 16.4.4 ULStreamCustomNew Function

```
LIB_API PDFStreamHandle ULStreamCustomNew(PDFLibHandle Lib, StreamReadBufferProc ReadBuffer, StreamWriteBufferProc WriteBuffer, StreamGetPositionProc GetPosition, StreamSetPositionProc SetPosition, StreamGetSizeProc GetSize, StreamGetCharProc GetChar, StreamLookCharProc LookChar, void * AStream);
```

**File**

VSBaseA.h

**Description**

This is function ULStreamCustomNew.

---

## 16.4.5 ULStreamFileHandleNew Function

```
LIB_API PDFStreamHandle ULStreamFileHandleNew(PDFLibHandle Lib, PDFFileHandle FileHandle);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[in] PDF Library Handle.
PDFFileHandle FileHandle	[in] PDF File Handle.

**Returns**

File PDF Stream Handle.

**Description**

Creates new file PDF Stream by PDF File Handle.

**See Also**

ULStreamClose ([↗](#) see page 151)

---

## 16.4.6 ULStreamFileNew Function

```
LIB_API PDFStreamHandle ULStreamFileNew(PDFLibHandle Lib, char * FileName, ppFileOpenMode OpenMode);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[in] PDF Library Handle.
char * FileName	[in] PDF File Name, text string.
ppFileOpenMode OpenMode	[in] Open Mode : read or write.

**Returns**

File PDF Stream Handle.

**Description**

Creates new file PDF Stream by filename and open mode.

**See Also**

ULStreamClose ([↗](#) see page 151)

---

## 16.4.7 ULStreamGetPosition Function

```
LIB_API ppUns32 ULStreamGetPosition(PDFStreamHandle Stream);
```

### File

VSBASEA.h

### Parameters

Parameters	Description
PDFStreamHandle Stream	[in] PDF Stream Handle.

### Returns

Stream position.

### Description

Gets Stream position ( offset from start of stream ).

### See Also

ULStreamSetPosition ( see page 156)

---

## 16.4.8 ULStreamGetSize Function

```
LIB_API ppUns32 ULStreamGetSize(PDFStreamHandle Stream);
```

### File

VSBASEA.h

### Parameters

Parameters	Description
PDFStreamHandle Stream	[in] PDF Stream Handle.

### Returns

Stream size in bytes.

### Description

Gets Stream size in bytes.

### See Also

ULStreamSetSize ( see page 156)

---

## 16.4.9 ULStreamLookChar Function

```
LIB_API ppInt32 ULStreamLookChar(PDFStreamHandle Stream);
```

### File

VSBASEA.h

**Parameters**

Parameters	Description
PDFStreamHandle Stream	[in] PDF Stream Handle.

**Returns**

One character form Stream. If it returns -1 than it is EOF ( end of stream ).

**Description**

Reads one character from Stream. Same as ULStreamReadChar ( see page 155), only stream position stays on that place.

**See Also**

ULStreamWriteChar ( see page 157)

---

## 16.4.10 ULStreamMemNew Function

```
LIB_API PDFStreamHandle ULStreamMemNew(PDFLibHandle Lib, ppUns32 Size);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[in] PDF Library Handle.
ppUns32 Size	[in] Initializing size of memory Stream in bytes.

**Returns**

Memory PDF Stream Handle.

**Description**

Creates new Memory PDF Stream with initializing size.

**See Also**

ULStreamClose ( see page 151)

---

## 16.4.11 ULStreamMemReadOnlyNew Function

```
LIB_API PDFStreamHandle ULStreamMemReadOnlyNew(PDFLibHandle Lib, void * Buffer, ppUns32 Length);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFLibHandle Lib	[in] PDF Library Handle.
void * Buffer	[in] Existed memory buffer.
ppUns32 Length	[in] Size of buffer in bytes.

**Returns**

Memory PDF Stream Handle.

**Description**

Converts memory buffer to PDF Stream.

**See Also**

ULStreamClose ([↗](#) see page 151)

---

## 16.4.12 ULStreamReadBuffer Function

```
LIB_API ppUns32 ULStreamReadBuffer(PDFStreamHandle Stream, void * Buffer, ppUns32 Count);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFStreamHandle Stream	[in] PDF Stream Handle.
void * Buffer	[out] Memory buffer for data.
ppUns32 Count	[in] Count of bytes which we want to read from Stream.

**Returns**

Count of bytes which read from Stream.

**Description**

Reads from PDF Stream to memory buffer some count of the bytes.

**See Also**

ULStreamWriteBuffer ([↗](#) see page 157)

---

## 16.4.13 ULStreamReadChar Function

```
LIB_API ppInt32 ULStreamReadChar(PDFStreamHandle Stream);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFStreamHandle Stream	[in] PDF Stream Handle.

**Returns**

One character form Stream. If it returns -1 than it is EOF ( end of stream ).

**Description**

Reads one character from Stream.

**See Also**

ULStreamWriteChar ([↗](#) see page 157)

---

## 16.4.14 ULStreamReadLine Function

```
LIB_API char * ULStreamReadLine(PDFStreamHandle Stream);
```

### File

VSBaseA.h

### Parameters

Parameters	Description
PDFStreamHandle Stream	[in] PDF Stream Handle.

### Returns

Line from Stream. Text string terminated by zero. Must be free after use.

### Description

Reads one line from Stream. Line is text string to character EOL ( end of line )

---

## 16.4.15 ULStreamSetPosition Function

```
LIB_API ppUns32 ULStreamSetPosition(PDFStreamHandle Stream, ppUns32 NewPosition);
```

### File

VSBaseA.h

### Parameters

Parameters	Description
PDFStreamHandle Stream	[in] PDF Stream Handle.
ppUns32 NewPosition	[in] New position in bytes ( offset from beginning ).

### Returns

Stream position after setting.

### Description

Sets new Stream position.

### See Also

ULStreamGetPosition ([🔗](#) see page 153)

---

## 16.4.16 ULStreamSetSize Function

```
LIB_API ppUns32 ULStreamSetSize(PDFStreamHandle Stream, ppUns32 Size);
```

### File

VSBaseA.h

### Parameters

Parameters	Description
PDFStreamHandle Stream	[in] PDF Stream Handle.
ppUns32 Size	[in] New size of PDF Stream.



**Returns**

Stream size in bytes.

**Description**

Sets new Stream size in bytes. Enlarges stream capacity.

**See Also**

ULStreamGetSize ([↗](#) see page 153)

---

## 16.4.17 ULStreamWriteBuffer Function

```
LIB_API ppUns32 ULStreamWriteBuffer(PDFStreamHandle Stream, void * Buffer, ppUns32 Count);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFStreamHandle Stream	[out] PDF Stream Handle.
void * Buffer	[in] Memory buffer with data.
ppUns32 Count	[in] Count of bytes which we want to write in Stream.

**Returns**

Count of written bytes in Stream.

**Description**

Writes from memory buffer to PDF Stream some count of the bytes.

**See Also**

ULStreamReadBuffer ([↗](#) see page 155)

---

## 16.4.18 ULStreamWriteChar Function

```
LIB_API ppUns32 ULStreamWriteChar(PDFStreamHandle Stream, char Character);
```

**File**

VSBaseA.h

**Parameters**

Parameters	Description
PDFStreamHandle Stream	[in] PDF Stream Handle.
char Character	[in] Writing data in size of one byte.

**Returns**

Count of written bytes in Stream.

**Description**

Writes one character to Stream.







**See Also**

ULStreamReadChar ([↗](#) see page 155)

# 17 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.



## Enumerations

Enumeration
 PDFBeadActionType ( <a href="#">see page 159</a> )
 PDFDestinationType ( <a href="#">see page 159</a> )
 PDFExplicitDestType ( <a href="#">see page 160</a> )
 PDFThreadActionType ( <a href="#">see page 161</a> )
 PDFVersion ( <a href="#">see page 162</a> )
 ppComponentDepth ( <a href="#">see page 162</a> )


## Types

Type
PDFExplicitDest ( <a href="#">see page 163</a> )
PDFNamedActionType ( <a href="#">see page 163</a> )
PDFThreadActionParamP ( <a href="#">see page 164</a> )
PPDFExplicitDest ( <a href="#">see page 164</a> )
ppFileOpenMode ( <a href="#">see page 165</a> )
TAnnotationStampName ( <a href="#">see page 165</a> )
TAnnotationTextName ( <a href="#">see page 166</a> )
TBorderStyle ( <a href="#">see page 167</a> )
THighLightMode ( <a href="#">see page 167</a> )
TPDFAcroEventType ( <a href="#">see page 168</a> )
TPDFAcroQuadding ( <a href="#">see page 168</a> )
TPDFAcroType ( <a href="#">see page 169</a> )
TPDFBlendMode ( <a href="#">see page 170</a> )
TPDFCheckBoxSign ( <a href="#">see page 170</a> )
TPDFCheckBoxStyle ( <a href="#">see page 171</a> )
TPDFCMYKColor ( <a href="#">see page 172</a> )
TPDFColor ( <a href="#">see page 172</a> )
TPDFColorDevice ( <a href="#">see page 173</a> )
TPDFHorJust ( <a href="#">see page 173</a> )
TPDFLineCap ( <a href="#">see page 173</a> )
TPDFLineJoin ( <a href="#">see page 174</a> )
TPDFPageBoxType ( <a href="#">see page 174</a> )
TPDFPageRotateAngle ( <a href="#">see page 175</a> )
TPDFRealPoint ( <a href="#">see page 175</a> )
TPDFRect ( <a href="#">see page 176</a> )
TPDFRenderingIntents ( <a href="#">see page 176</a> )
TPDFRGBColor ( <a href="#">see page 177</a> )
TPDFVersion ( <a href="#">see page 177</a> )
TPDFVertJust ( <a href="#">see page 178</a> )

Legend

	Enumeration
	Structure

Structures

Structure
 PDFThreadActionParam (see page 161)

# 17.1 PDFBeadActionType Enumeration

```
enum PDFBeadActionType {
    taBeadHandle,
    taBeadIndex
};
```

File

VSActionA.h

Members

Members	Description
taBeadHandle	Destination stored in PDFBeadHandle
taBeadIndex	Destination stored in index of the bead within its thread. The first bead in a thread has index 0.

Description

This is record PDFBeadActionType.

# 17.2 PDFDestinationType Enumeration

```
enum PDFDestinationType {
    pdfdtExplicit,
    pdfdtNamed
};
```

File

VSActionA.h

Members

Members	Description
pdfdtExplicit	Directly via explicit destination
pdfdtNamed	Indirectly via name destination

Description

Type of the PDF destination

## 17.3 PDFExplicitDestType Enumeration

```
enum PDFExplicitDestType {
    edtXYZ,
    edtFit,
    edtFitH,
    edtFitV,
    edtFitR,
    edtFitB,
    edtFitBH,
    edtFitBV
};
```

### File

VSActionA.h

### Members

Members	Description
edtXYZ	Display the page, with the coordinates ( <i>left</i> , <i>top</i> ) positioned at the top-left corner of the window and the contents of the page magnified by the factor <i>zoom</i> . A null value for any of the parameters <i>left</i> , <i>top</i> , or <i>zoom</i> specifies that the current value of that parameter is to be retained unchanged.
edtFit	Display the page, with its contents magnified just enough to fit the entire page within the window both horizontally and vertically. If the required horizontal and vertical magnification factors are different, use the smaller of the two, centering the page within the window in the other dimension.
edtFitH	Display the page, with the vertical coordinate <i>top</i> positioned at the <i>top</i> edge of the window and the contents of the page magnified just enough to fit the entire width of the page within the window.
edtFitV	Display the page, with the horizontal coordinate <i>left</i> positioned at the <i>left</i> edge of the window and the contents of the page magnified just enough to fit the entire height of the page within the window.
edtFitR	Display the page, with its contents magnified just enough to fit the rectangle specified by the coordinates <i>left</i> , <i>bottom</i> , <i>right</i> , and <i>top</i> entirely within the window both horizontally and vertically. If the required horizontal and vertical magnification factors are different, use the smaller of the two, centering the rectangle within the window in the other dimension.
edtFitB	Display the page, with its contents magnified just enough to fit its bounding box entirely within the window both horizontally and vertically. If the required horizontal and vertical magnification factors are different, use the smaller of the two, centering the bounding box within the window in the other dimension.
edtFitBH	Display the page, with the vertical coordinate <i>top</i> positioned at the <i>top</i> edge of the window and the contents of the page magnified just enough to fit the entire width of its bounding box within the window.
edtFitBV	Display the page, with the horizontal coordinate <i>left</i> positioned at the <i>left</i> edge of the window and the contents of the page magnified just enough to fit the entire height of its bounding box within the window.

Description

PDF explicit destination types

# 17.4 PDFThreadActionParam Structure

```
struct PDFThreadActionParam {
    PDFThreadActionType DestThreadType;
    union {
        PDFThreadHandle ThreadHandle;
        ppInt32 ThreadIndex;
        struct {
            char* String;
            ppUns32 Length;
        } ThreadTitle;
    } DestThread;
    PDFBeadActionType DestBeadType;
    union {
        PDFBeadHandle BeadHandle;
        ppInt32 BeadIndex;
    } DestBead;
};
```

File

VSActionA.h

Members

Members	Description
PDFThreadActionType DestThreadType;	Depends type of the thread
union { PDFThreadHandle ThreadHandle; ppInt32 ThreadIndex; struct { char* String; ppUns32 Length; } ThreadTitle; } DestThread;	The desired destination thread
PDFBeadActionType DestBeadType;	Depends type of the bead
union { PDFBeadHandle BeadHandle; ppInt32 BeadIndex; } DestBead;	The desired bead in the destination thread

Description

This structure used by PDFActionNewThread (🔗 see page 112) to create new Thread action

# 17.5 PDFThreadActionType Enumeration

```
enum PDFThreadActionType {
    taThreadHandle,
    taThreadIndex,
    taThreadTitle
};
```

**File**

VSActionA.h

**Members**

Members	Description
taThreadHandle	Destination is stored in PDFThreadHandle
taThreadIndex	Destination is stored in index of the threads in PDF document. The first thread in the document has index 0.
taThreadTitle	Destination is stored in the title of the thread.

**Description**

This enumeration defines type of the store thread info in thread action

---

## 17.6 PDFVersion Enumeration

```
enum PDFVersion {  
    pdfver10 = 0,  
    pdfver11 = 1,  
    pdfver12 = 2,  
    pdfver13 = 3,  
    pdfver14 = 4,  
    pdfver15 = 5,  
    pdfver16 = 6,  
    pdfver17 = 7  
};
```

**File**

VSDocA.h

**Members**

Members	Description
pdfver10 = 0	PDF Document Version is 1.0
pdfver11 = 1	PDF Document Version is 1.1
pdfver12 = 2	PDF Document Version is 1.2
pdfver13 = 3	PDF Document Version is 1.3
pdfver14 = 4	PDF Document Version is 1.4
pdfver15 = 5	PDF Document Version is 1.5
pdfver16 = 6	PDF Document Version is 1.6
pdfver17 = 7	PDF Document Version is 1.7

**Description**

Type of supported PDF Document Versions

---

## 17.7 ppComponentDepth Enumeration

```
enum ppComponentDepth {  
    it1bit = 1,  
    it2bit = 2,  
    it4bit = 4,  
    it8bit = 8,  
    it16bit = 16  
};
```

```
};
```

**File**

VSIImageA.h

**Description**

This is record ppComponentDepth.

## 17.8 PDFExplicitDest Type

```
typedef struct {
    PDFExplicitDestType Type;
    ppUns32 Page;
    ppReal a;
    ppReal b;
    ppReal c;
    ppReal d;
} PDFExplicitDest, * PPDFExplicitDest;
```

**File**

VSActionA.h

**Members**

Members	Description
PDFExplicitDestType Type;	Explicit destination type
ppUns32 Page;	Index of the page in PDF document (Begin from 0 )
ppReal a;	Value depended from type of the destination left for edtXYZ, top for edtFitH and edtFitBH, left for edtFitV,edtFitBV, and edtFitR
ppReal b;	Value depended from type of the destination top for edtXYZ, bottom for edtFitR
ppReal c;	Value depended from type of the destination zoom for edtXYZ, right for edtFitR
ppReal d;	Value depended from type of the destination top for edtFitR

**Description**

This struct for specifying a destination explicitly in a PDF file

## 17.9 PDFNamedActionType Type

```
typedef enum {
    naNextPage,
    naPrevPage,
    naFirstPage,
    naLastPage
} PDFNamedActionType;
```

**File**

VSActionA.h

**Members**

Members	Description
naNextPage	Go to the next page of the document.

naPrevPage	Go to the previous page of the document.
naFirstPage	Go to the first page of the document.
naLastPage	Go to the last page of the document.

Description

This enumeration defines type of the operation for named action

# 17.10 PDFThreadActionParamP Type

```
typedef struct PDFThreadActionParam {
    PDFThreadActionType DestThreadType;
    union {
        PDFThreadHandle ThreadHandle;
        ppInt32 ThreadIndex;
        struct {
            char* String;
            ppUns32 Length;
        } ThreadTitle;
    } DestThread;
    PDFBeadActionType DestBeadType;
    union {
        PDFBeadHandle BeadHandle;
        ppInt32 BeadIndex;
    } DestBead;
} * PDFThreadActionParamP;
```

File

VSActionA.h

Members

Members	Description
PDFThreadActionType DestThreadType;	Depends type of the thread
union { PDFThreadHandle ThreadHandle; ppInt32 ThreadIndex; struct { char* String; ppUns32 Length; } ThreadTitle; } DestThread;	The desired destination thread
PDFBeadActionType DestBeadType;	Depends type of the bead
union { PDFBeadHandle BeadHandle; ppInt32 BeadIndex; } DestBead;	The desired bead in the destination thread

Description

This structure used by PDFActionNewThread ( see page 112) to create new Thread action

# 17.11 PPDFExplicitDest Type

```
typedef struct {
```



```
PDFExplicitDestType Type;
ppUns32 Page;
ppReal a;
ppReal b;
ppReal c;
ppReal d;
} PDFExplicitDest, * PPDFExplicitDest;
```

File

VSActionA.h

Members

Members	Description
PDFExplicitDestType Type;	Explicit destination type
ppUns32 Page;	Index of the page in PDF document (Begin from 0 )
ppReal a;	Value depended from type of the destination left for edtXYZ, top for edtFitH and edtFitBH, left for edtFitV,edtFitBV, and edtFitR
ppReal b;	Value depended from type of the destination top for edtXYZ, bottom for edtFitR
ppReal c;	Value depended from type of the destination zoom for edtXYZ, right for edtFitR
ppReal d;	Value depended from type of the destination top for edtFitR

Description

This struct for specifying a destination explicitly in a PDF file

# 17.12 ppFileOpenMode Type

```
typedef enum {
    ppFileReadMode = 0,
    ppFileWriteMode
} ppFileOpenMode;
```

File

VSBaseA.h

Members

Members	Description
ppFileReadMode = 0	Read File Mode
ppFileWriteMode	Write File Mode

Description

File Open Mode Type

# 17.13 TAnnotationStampName Type

```
typedef enum {
    asnApproved,
    asnAsIs,
    asnConfidential,
    asnDepartmental,
    asnDraft,
```

```
    asnExperimental,
    asnExpired,
    asnFinal,
    asnForComment,
    asnForPublicRelease,
    asnNotApproved,
    asnNotForPublicRelease,
    asnSold,
    asnTopSecret,
    asnOther
} TAnnotationStampName;
```

File

VSAnnotA.h

Members

Members	Description
asnApproved	Approved icon type
asnAsIs	AsIs icon type
asnConfidential	Confidential icon type
asnDepartmental	Departmental icon type
asnDraft	Draft icon type
asnExperimental	Experimental icon type
asnExpired	Expired icon type
asnFinal	Final icon type
asnForComment	For comment icon type
asnForPublicRelease	For public release icon type
asnNotApproved	Not approved icon type
asnNotForPublicRelease	Not for public release icon type
asnSold	Sold icon type
asnTopSecret	Top secret icon type
asnOther	Other icon type

Description

Available stamp icon types

# 17.14 TAnnotationTextName Type

```
typedef enum {
    atnNote,
    atnComment,
    atnHelp,
    atnInsert,
    atnKey,
    atnNewParagraph,
    atnParagraph
} TAnnotationTextName;
```

File

VSAnnotA.h

Members

Members	Description
atnNote	Note annotation
atnComment	Comment annotation

atnHelp	Help annotation
atnInsert	Insert annotation
atnKey	Key annotation
atnNewParagraph	New paragraph annotation
atnParagraph	Paragraph annotation

**Description**

Available annotations names

---

## 17.15 TBorderStyle Type

```
typedef enum {  
    bsSolid,  
    bsDashed,  
    bsBeveled,  
    bsInset,  
    bsUnderline  
} TBorderStyle;
```

**File**

VSAnnotA.h

**Members**

Members	Description
bsSolid	A solid rectangle surrounding the annotation.
bsDashed	A dashed rectangle surrounding the annotation.
bsBeveled	A simulated embossed rectangle that appears to be raised above the surface of the page.
bsInset	A simulated engraved rectangle that appears to be recessed below the surface of the page.
bsUnderline	A single line along the bottom of the annotation rectangle.

**Description**

The annotation border styles

---

## 17.16 THighLightMode Type

```
typedef enum {  
    hlmNoHightLight,  
    hlmInvert,  
    hlmOutline,  
    hlmPush  
} THighLightMode;
```

**File**

VSAnnotA.h

**Members**

Members	Description
hlmNoHightLight	No highlighting.
hlmInvert	Invert the contents of the annotation rectangle.

hlmOutline	Invert the annotation's border.
hlmPush	Display the annotation's down appearance.

**Description**

The annotation's highlighting mode, the visual effect to be used when the mouse button is pressed or held down inside its active area.

## 17.17 TPDFAcroEventType Type

```
typedef enum {
    PDFAcroEventTypeActivate = 0,
    PDFAcroEventTypeEnter,
    PDFAcroEventTypeExit,
    PDFAcroEventTypePress,
    PDFAcroEventTypeRelease,
    PDFAcroEventTypeFocusOn,
    PDFAcroEventTypeFocusOff,
    PDFAcroEventTypeKeystroke,
    PDFAcroEventTypeFormat,
    PDFAcroEventTypeValidate,
    PDFAcroEventTypeCalculate
} TPDFAcroEventType;
```

**File**

VSAcroFormA.h

**Members**

Members	Description
PDFAcroEventTypeActivate = 0	event on activate, primary action
PDFAcroEventTypeEnter	event on enter in the active area
PDFAcroEventTypeExit	event on exit from the active area
PDFAcroEventTypePress	event on press mouse button inside it
PDFAcroEventTypeRelease	event on release mouse button inside
PDFAcroEventTypeFocusOn	event on receive the input focus
PDFAcroEventTypeFocusOff	event on lose the input focus
PDFAcroEventTypeKeystroke	event on change text value in field
PDFAcroEventTypeFormat	event on format value in the field
PDFAcroEventTypeValidate	event on change field's value in field
PDFAcroEventTypeCalculate	event on recalculate value

**Description**

Action Event Type for Acro Form Objects. Set of events.

## 17.18 TPDFAcroQuadding Type

```
typedef enum {
    PDFAcroQuaddingLeftTop = 0,
    PDFAcroQuaddingTop,
    PDFAcroQuaddingRightTop,
    PDFAcroQuaddingLeft,
    PDFAcroQuaddingCenter,
    PDFAcroQuaddingRight,
}
```

```

    PDFAcroQuaddingLeftBottom,
    PDFAcroQuaddingBottom,
    PDFAcroQuaddingRightBottom
} TPDFAcroQuadding;

```

**File**

VSAcroFormA.h

**Members**

Members	Description
PDFAcroQuaddingLeftTop = 0	attach text to left top corner of field
PDFAcroQuaddingTop	attach text to top central site of field
PDFAcroQuaddingRightTop	attach text to right top corner of field
PDFAcroQuaddingLeft	attach text to left central site of field
PDFAcroQuaddingCenter	attach text to center of field
PDFAcroQuaddingRight	attach text to right central site of field
PDFAcroQuaddingLeftBottom	attach text to left bottom corner of field
PDFAcroQuaddingBottom	attach text to bottom central site of field
PDFAcroQuaddingRightBottom	attach text to right bottom corner of field

**Description**

Acro Form Object Quadding Type. Text justification style.

## 17.19 TPDFAcroType Type

```

typedef enum {
    PDFAcroTypeUnknown = 0,
    PDFAcroTypePushButton,
    PDFAcroTypeCheckBox,
    PDFAcroTypeRadioButton,
    PDFAcroTypeEditBox,
    PDFAcroTypeComboBox,
    PDFAcroTypeListBox,
    PDFAcroTypeSignature
} TPDFAcroType;

```

**File**

VSAcroInfoA.h

**Members**

Members	Description
PDFAcroTypeUnknown = 0	Unknown Type, in failure case
PDFAcroTypePushButton	Button for select single action
PDFAcroTypeCheckBox	Button for check single item
PDFAcroTypeRadioButton	Button from group for select only one item from ensemble
PDFAcroTypeEditBox	Variable text edit field for change text item
PDFAcroTypeComboBox	Field for select one text item from list
PDFAcroTypeListBox	Box for select item(s) from list
PDFAcroTypeSignature	Field for sign in document, maybe invisible

**Description**

Acro Form Object Type. Interactive Control Type.

## 17.20 TPDFBlendMode Type

```
typedef enum {
    blmoNormal,
    blmoMultiply,
    blmoScreen,
    blmoOverlay,
    blmoDarken,
    blmoLighten,
    blmoColorDodge,
    blmoColorBurn,
    blmoHardLight,
    blmoSoftLight,
    blmoDifference,
    blmoExclusion
} TPDFBlendMode;
```

### File

VSGStateA.h

### Members

Members	Description
blmoNormal	Selects the source color, ignoring the backdrop
blmoMultiply	Multiplies the backdrop and source color values
blmoScreen	Multiplies the complements of the backdrop and source color values, then complements the result
blmoOverlay	Multiplies or screens the colors, depending on the backdrop color.
blmoDarken	Selects the darker of the backdrop and source colors.
blmoLighten	Selects the lighter of the backdrop and source colors.
blmoColorDodge	Brightens the backdrop color to reflect the source color. Painting with black produces no change.
blmoColorBurn	Darkens the backdrop color to reflect the source color. Painting with white produces no change.
blmoHardLight	Multiplies or screens the colors, depending on the source color value.
blmoSoftLight	Darkens or lightens the colors, depending on the source color value.
blmoDifference	Subtracts the darker of the two constituent colors from the lighter.
blmoExclusion	Produces an effect similar to that of the Difference mode, but lower in contrast.

### Description

Blending mode

## 17.21 TPDFCheckBoxSign Type

```
typedef enum {
    cbsFinger = 0053,
    cbsPen = 0062,
    cbsVmark = 0063,
    cbsNike = 0064,
```

```
cbsCross = 0065,  
cbsX = 0066,  
cbsCheck = 0070,  
cbsPlus = 0072,  
cbsDiamond = 0106,  
cbsStar = 0110,  
cbsFlower = 0137,  
cbsSnow = 0144,  
cbsCircle = 0154,  
cbsRectangle = 0156,  
cbsRhombus = 0165  
} TPDFCheckBoxSign;
```

**File**

VSAcroFormA.h

**Members**

Members	Description
cbsFinger = 0053	Finger' mark
cbsPen = 0062	Pen' mark
cbsVmark = 0063	V-style mark
cbsNike = 0064	V-style mark
cbsCross = 0065	Cross-style mark
cbsX = 0066	Cross-style mark
cbsCheck = 0070	Cross-style mark
cbsPlus = 0072	Cross-style mark
cbsDiamond = 0106	Rhombus-style mark
cbsStar = 0110	Star' mark
cbsFlower = 0137	Flower' mark
cbsSnow = 0144	Snowflake' mark
cbsCircle = 0154	Circle' mark
cbsRectangle = 0156	Rectangle' mark
cbsRhombus = 0165	Rhombus-style mark

**Description**

Type of CheckBox Mark - code of Zapf Dingbats sign ( 32 - 255 ). Here is definition in octal system

## 17.22 TPDFCheckBoxStyle Type

```
typedef enum {  
    cbfRectangle = 0,  
    cbfCircle  
} TPDFCheckBoxStyle;
```

**File**

VSAcroFormA.h

**Members**

Members	Description
cbfRectangle = 0	Rectangle style
cbfCircle	Circle style

**Description**

Type of CheckBox Style.

# 17.23 TPDFCMYKColor Type

```
typedef struct {
    ppReal C;
    ppReal M;
    ppReal Y;
    ppReal K;
} TPDFCMYKColor;
```

File

VSTypes.h

Members

Members	Description
ppReal C;	Cyan component of Color
ppReal M;	Magenta component of Color
ppReal Y;	Yellow component of Color
ppReal K;	Black component of Color

Description

CMYK Color Type

# 17.24 TPDFColor Type

```
typedef struct {
    TPDFColorDevice Device;
    union {
        ppReal Gray;
        TPDFRGBColor RGB;
        TPDFCMYKColor CMYK;
    } Color;
} TPDFColor;
```

File

VSTypes.h

Members

Members	Description
TPDFColorDevice Device;	Color Device
union { ppReal Gray; TPDFRGBColor RGB; TPDFCMYKColor CMYK; } Color;	Color Value
ppReal Gray;	Gray scale value
TPDFRGBColor RGB;	RGB color value
TPDFCMYKColor CMYK;	CMYK color value

Description

PDF Color Type



# 17.25 TPDFColorDevice Type

```
typedef enum {
    cgGray = 1,
    cgRGB = 3,
    cgCMYK = 4
} TPDFColorDevice;
```

**File**

VSTypes.h

**Description**

Color Device Type

# 17.26 TPDFHorJust Type

```
typedef enum {
    hjLeft,
    hjCenter,
    hjRight
} TPDFHorJust;
```

**File**

VSCanvasA.h

# 17.27 TPDFLineCap Type

```
typedef enum {
    lcButtEnd,
    lcRound,
    lcProjectingSquare
} TPDFLineCap;
```

**File**

VSCanvasA.h

**Members**

Members	Description
lcButtEnd	The stroke is squared off at the endpoint of the path. There is no projection beyond the end of the path.
lcRound	A semicircular arc with a diameter equal to the line width is drawn around the endpoint and filled in.
lcProjectingSquare	The stroke continues beyond the endpoint of the path for a distance equal to half the line width and is then squared off.

**Description**

The line cap style specifies the shape to be used at the ends of opened subpaths (and dashes, if any) when they are stroked.

## 17.28 TPDFLineJoin Type

```
typedef enum {  
    ljMiter,  
    ljRound,  
    ljBevel  
} TPDFLineJoin;
```

### File

VSCanvasA.h

### Members

Members	Description
ljMiter	The outer edges of the strokes for the two segments are extended until they meet at an angle, as in a picture frame. If the segments meet at too sharp an angle, a bevel join is used instead.
ljRound	A circle with a diameter equal to the line width is drawn around the point where the two segments meet and is filled in, producing a rounded corner.
ljBevel	The two segments are finished with butt caps and the resulting notch beyond the ends of the segments is filled with a triangle

### Description

The line join style specifies the shape to be used at the corners of paths that are stroked.

## 17.29 TPDFPageBoxType Type

```
typedef enum {  
    pbnMediaBox,  
    pbnCropBox,  
    pbnBleedBox,  
    pbnTrimBox,  
    pbnArtBox  
} TPDFPageBoxType;
```

### File

VSPageA.h

### Members

Members	Description
pbnMediaBox	A rectangle, expressed in default user space units, defining the boundaries of the physical medium on which the page is intended to be displayed or printed
pbnCropBox	A rectangle, expressed in default user space units, defining the visible region of default user space. When the page is displayed or printed, its contents are to be clipped (cropped) to this rectangle and then imposed on the output medium in some implementation defined manner.
pbnBleedBox	A rectangle, expressed in default user space units, defining the region to which the contents of the page should be clipped when output in a production environment

pbnTrimBox	A rectangle, expressed in default user space units, defining the intended dimensions of the finished page after trimming
pbnArtBox	A rectangle, expressed in default user space units, defining the extent of the page's meaningful content (including potential white space) as intended by the page's creator

**Description**

Page Box Type

---

## 17.30 TPDFPageRotateAngle Type

```
typedef enum {  
    pra0 = 0,  
    pra90,  
    pra180,  
    pra270  
} TPDFPageRotateAngle;
```

**File**

VSPageA.h

**Members**

Members	Description
pra0 = 0	0 deg. - rotation angle
pra90	90 deg. - rotation angle
pra180	180 deg. - rotation angle
pra270	270 deg. - rotation angle

**Description**

Page Rotation Angle. The number of degrees by which the page should be rotated clockwise when displayed or printed. The value must be a multiple of 90. Default value: 0.

---

## 17.31 TPDFRealPoint Type

```
typedef struct {  
    ppReal x;  
    ppReal y;  
} TPDFRealPoint;
```

**File**

VSTypes.h

**Members**

Members	Description
ppReal x;	horizontal axis coordinate
ppReal y;	vertical axis coordinate

**Description**

Point Type

# 17.32 TPDFRect Type

```
typedef struct {
    ppReal xl;
    ppReal yl;
    ppReal xr;
    ppReal yr;
} TPDFRect;
```

File

VSTypes.h

Members

Members	Description
ppReal xl;	Left border coordinate
ppReal yl;	Top border coordinate
ppReal xr;	Right border coordinate
ppReal yr;	Bottom border coordinate

Description

Rectangle Type

# 17.33 TPDFRenderingIntents Type

```
typedef enum {
    ReIntAbsoluteColormetric,
    ReIntRelativeColorMetrics,
    ReIntSaturation,
    ReIntPerceptual
} TPDFRenderingIntents;
```

File

VSGStateA.h

Members

Members	Description
ReIntAbsoluteColormetric	Colors are represented solely with respect to the light source; no correction is made for the output medium's white point (such as the color of unprinted paper).
ReIntRelativeColorMetrics	Colors are represented with respect to the combination of the light source and the output medium's white point (such as the color of unprinted paper).
ReIntSaturation	Colors are represented in a manner that preserves or emphasizes saturation.
ReIntPerceptual	Colors are represented in a manner that provides a pleasing perceptual appearance.

Description

Converting CIE-based colors to device colors

# 17.34 TPDFRGBColor Type

```
typedef struct {
    ppReal R;
    ppReal G;
    ppReal B;
} TPDFRGBColor;
```

**File**

VSTypes.h

**Members**

Members	Description
ppReal R;	Red component of Color
ppReal G;	Green component of Color
ppReal B;	Blue component of Color

**Description**

RGB Color Type

# 17.35 TPDFVersion Type

```
typedef enum PDFVersion {
    pdfver10 = 0,
    pdfver11 = 1,
    pdfver12 = 2,
    pdfver13 = 3,
    pdfver14 = 4,
    pdfver15 = 5,
    pdfver16 = 6,
    pdfver17 = 7
} TPDFVersion;
```

**File**

VSDocA.h

**Members**

Members	Description
pdfver10 = 0	PDF Document Version is 1.0
pdfver11 = 1	PDF Document Version is 1.1
pdfver12 = 2	PDF Document Version is 1.2
pdfver13 = 3	PDF Document Version is 1.3
pdfver14 = 4	PDF Document Version is 1.4
pdfver15 = 5	PDF Document Version is 1.5
pdfver16 = 6	PDF Document Version is 1.6
pdfver17 = 7	PDF Document Version is 1.7

**Description**

Type of supported PDF Document Versions

---

## 17.36 TPDFVertJust Type

```
typedef enum {  
    vjTop,  
    vjCenter,  
    vjBottom  
} TPDFVertJust;
```

### File

VSCanvasA.h

# 18 Symbol Reference

## 18.1 Types

The following table lists types in this documentation.

Types

Type
CosType
PDFActionHandle
PDFAnnotationHandle
PDFBeadHandle
PDFCosHandle
PDFDestinationHandle
PDFOutlineHandle
PDFThreadHandle
TImageCompressionType
TKeyValidType
TPDFEncodingType
TPDFInformation
TPDFPageOrientation
TPDFPageSize
TPDFProtectionType
TPDFStandardFont

### 18.1.1 CosType Type

```
typedef _EXP enum CosType@1 CosType;
```

File

VSCosA.h

Description

Available Cos objects

### 18.1.2 PDFActionHandle Type

```
typedef struct _t_PDFCosHandle PCosObj PDFHandle PDFActionHandle;
```

File

VSTypes.h

**Description**

This is type PDFActionHandle.

---

## 18.1.3 PDFAnnotationHandle Type

```
typedef struct _t_PDFCosHandle PCosObj PDFHandle PDFAnnotationHandle;
```

**File**

VSTypes.h

**Description**

This is type PDFAnnotationHandle.

---

## 18.1.4 PDFBeadHandle Type

```
typedef struct _t_PDFCosHandle PCosObj PDFHandle PDFBeadHandle;
```

**File**

VSTypes.h

**Description**

This is type PDFBeadHandle.

---

## 18.1.5 PDFCosHandle Type

```
typedef struct _t_PDFCosHandle PCosObj PDFHandle PDFCosHandle;
```

**File**

VSTypes.h

**Description**

This is type PDFCosHandle.

---

## 18.1.6 PDFDestinationHandle Type

```
typedef struct _t_PDFCosHandle PCosObj PDFHandle PDFDestinationHandle;
```

**File**

VSTypes.h

**Description**

This is type PDFDestinationHandle.



---

## 18.1.7 PDFOutlineHandle Type

```
typedef struct _t_PDFCosHandle PCosObj PDFHandle PDFOutlineHandle;
```

**File**

VSTypes.h

**Description**

This is type PDFOutlineHandle.

---

## 18.1.8 PDFThreadHandle Type

```
typedef struct _t_PDFCosHandle PCosObj PDFHandle PDFThreadHandle;
```

**File**

VSTypes.h

**Description**

Bead Handle

---

## 18.1.9 TImageCompressionType Type

```
typedef _EXP enum _t_TImageCompressionType TImageCompressionType;
```

**File**

VSImageA.h

**Description**

Available image compression types

---

## 18.1.10 TKeyValidType Type

```
typedef _EXP enum _t_TKeyValidType TKeyValidType;
```

**File**

VSDocA.h

**Description**

Password Type of Crypted PDF Document. Password Validity.

---

## 18.1.11 TPDFEncodingType Type

```
typedef _EXP enum _t_TPDFEncodingType TPDFEncodingType;
```

**File**

VSTFontA.h

**Description**

Font encoding

---

## 18.1.12 TPDFInformation Type

```
typedef _EXP enum _t_TPDFInformation TPDFInformation;
```

**File**

VSDocA.h

---

## 18.1.13 TPDFPageOrientation Type

```
typedef _EXP enum _t_TPDFPageOrientation TPDFPageOrientation;
```

**File**

VSDocA.h

**Description**

Page Orientation Type

---

## 18.1.14 TPDFPageSize Type

```
typedef _EXP enum _t_TPDFPageSize TPDFPageSize;
```

**File**

VSDocA.h

**Description**

Type of usual PDF Document's Page Sizes

---

## 18.1.15 TPDFProtectionType Type

```
typedef _EXP enum _t_TPDFProtectionType TPDFProtectionType;
```

**File**

VSDocA.h

**Description**

Protection Key-Length Type of Crypted PDF Document

## 18.1.16 TPDFStdandardFont Type

```
typedef _EXP enum _t_TPDFStdandardFont TPDFStdandardFont;
```

### File

VSFontA.h

### Description

Standard 14 fonts enum

## Index

### A

Action Level 105  
Annotation Level 99  
Appending Of The Pages To PDF 10  
Atom Level 143

### B

Bead Operations 96

### C

Canvas Drawing Level 48  
Color Level 142  
Common Cos Object Functions 120  
Cos Array Object 132  
Cos Boolean Object 124  
Cos Dictionary Object 136  
Cos Integer Object 127  
Cos Name Object 129  
Cos Null Object 123  
Cos Number Objects 125  
Cos Real Object 126  
Cos Stream Object 140  
Cos String Object 131  
CosArrayAppend 133  
CosArrayAppend function 133  
CosArrayClear 133  
CosArrayClear function 133  
CosArrayCount 134  
CosArrayCount function 134  
CosArrayInsert 134  
CosArrayInsert function 134  
CosArrayItem 135  
CosArrayItem function 135  
CosArrayNew 135  
CosArrayNew function 135  
CosArrayRemove 136  
CosArrayRemove function 136  
CosBoolGetValue 124  
CosBoolGetValue function 124  
CosBoolNew 124  
CosBoolNew function 124  
CosBoolSetValue 125  
CosBoolSetValue function 125  
CosCopy 120  
CosCopy function 120  
CosDictAppend 137  
CosDictAppend function 137  
CosDictClear 137  
CosDictClear function 137  
CosDictCount 137  
CosDictCount function 137  
CosDictGetPair 139  
CosDictGetPair function 139  
CosDictNew 139  
CosDictNew function 139  
CosDictRemoveKey 138  
CosDictRemoveKey function 138  
CosDictValueByName 138  
CosDictValueByName function 138  
CosFree 121  
CosFree function 121  
CosGetFromDoc 121  
CosGetFromDoc function 121  
CosGetGeneration 122  
CosGetGeneration function 122  
CosGetID 122  
CosGetID function 122  
CosGetType 120  
CosGetType function 120  
CosIntGetValue 127  
CosIntGetValue function 127  
CosIntNew 128  
CosIntNew function 128  
CosIntSetValue 128  
CosIntSetValue function 128  
CosIsIndirect 122  
CosIsIndirect function 122  
CosNameGetValue 129  
CosNameGetValue function 129  
CosNameNew 130  
CosNameNew function 130

CosNameSetValue 130  
CosNameSetValue function 130  
CosNullNew 123  
CosNullNew function 123  
CosNumberGetValue 129  
CosNumberGetValue function 129  
CosObject Level 120  
CosRealGetValue 126  
CosRealGetValue function 126  
CosRealNew 126  
CosRealNew function 126  
CosRealSetValue 127  
CosRealSetValue function 127  
CosStreamGetAttr 141  
CosStreamGetAttr function 141  
CosStreamGetValue 140  
CosStreamGetValue function 140  
CosStreamNew 141  
CosStreamNew function 141  
CosStringGetValue 131  
CosStringGetValue function 131  
CosStringNew 131  
CosStringNew function 131  
CosStringSetValue 132  
CosStringSetValue function 132  
CosType 179  
CosType type 179

## D

Document Properties 11  
DonePDFLibrary 3  
DonePDFLibrary function 3

## E

edtFit enumeration member 160  
edtFitB enumeration member 160  
edtFitBH enumeration member 160  
edtFitBV enumeration member 160  
edtFitH enumeration member 160  
edtFitR enumeration member 160  
edtFitV enumeration member 160  
edtXYZ enumeration member 160

Extended Graphic State Level 35

## F

File Level 146  
Font Level 45

## G

Goto Action 107  
Goto Remote Action 108  
Graphic State Operations 48

## H

Hide Action 109

## I

Import Action 113  
InitPDFLibrary 2  
InitPDFLibrary function 2  
InitPDFLibraryWithParams 2  
InitPDFLibraryWithParams function 2  
it16bit enumeration member 162  
it1bit enumeration member 162  
it2bit enumeration member 162  
it4bit enumeration member 162  
it8bit enumeration member 162

## J

Javascript Action 118

## L

Launch Action 108  
Load And Save PDF Documents 6

## N

Named Action 112

## O

Other Drawing Operations 69

## P

Path Construction Operations 53

Path Painting Operations 61

PBXAppendLine 71

PBXAppendLine function 71

PBXArc 54

PBXArc function 54

PBXArc2 55

PBXArc2 function 55

PBXCircle 56

PBXCircle function 56

PBXClip 62

PBXClip function 62

PBXClose 71

PBXClose function 71

PBXClosePath 61

PBXClosePath function 61

PBXCurveTo 58

PBXCurveTo function 58

PBXEllipse 55

PBXEllipse function 55

PBXEoClip 62

PBXEoClip function 62

PBXEoFill 62

PBXEoFill function 62

PBXEoFillAndStroke 63

PBXEoFillAndStroke function 63

PBXFill 63

PBXFill function 63

PBXFillAndStroke 64

PBXFillAndStroke function 64

PBXGetHeight 70

PBXGetHeight function 70

PBXGetTextWidth 65

PBXGetTextWidth function 65

PBXGetUnicodeWidth 65

PBXGetUnicodeWidth function 65

PBXGetWidth 70

PBXGetWidth function 70

PBXLineTo 58

PBXLineTo function 58

PBXMoveTo 59

PBXMoveTo function 59

PBXNewPath 54

PBXNewPath function 54

PBXNoDash 48

PBXNoDash function 48

PBXPie 56

PBXPie function 56

PBXPie2 57

PBXPie2 function 57

PBXPlayMetaFile 70

PBXPlayMetaFile function 70

PBXRectangle 59

PBXRectangle function 59

PBXRectRotated 60

PBXRectRotated function 60

PBXRoundRect 60

PBXRoundRect function 60

PBXSetActiveFont 68

PBXSetActiveFont function 68

PBXSetActiveFontWithCharset 69

PBXSetActiveFontWithCharset function 69

PBXSetCharacterSpacing 66

PBXSetCharacterSpacing function 66

PBXSetColor 49

PBXSetColor function 49

PBXSetDash 49

PBXSetDash function 49

PBXSetFillColor 49

PBXSetFillColor function 49

PBXSetFlatness 50

PBXSetFlatness function 50

PBXSetGState 53

PBXSetGState function 53

PBXSetHorizontalScaling 66

PBXSetHorizontalScaling function 66

PBXSetLineCap 52

PBXSetLineCap function 52

PBXSetLineJoin 52

PBXSetLineJoin function 52

PBXSetLineWidth 50

PBXSetLineWidth function 50

PBXSetMiterLimit 52	PDFAcroObjectSetFlag 76
PBXSetMiterLimit function 52	PDFAcroObjectSetFlag function 76
PBXSetStrokeColor 50	PDFAcroObjectSetFont 77
PBXSetStrokeColor function 50	PDFAcroObjectSetFont function 77
PBXSetTextRenderingMode 66	PDFAcroObjectSetStyle 77
PBXSetTextRenderingMode function 66	PDFAcroObjectSetStyle function 77
PBXSetWordSpacing 67	PDFAcroPushButtonSetMiter 78
PBXSetWordSpacing function 67	PDFAcroPushButtonSetMiter function 78
PBXShowImage 72	PDFActionHandle 179
PBXShowImage function 72	PDFActionHandle type 179
PBXStateRestore 51	PDFActionHideAddAnnotation 110
PBXStateRestore function 51	PDFActionHideAddAnnotation function 110
PBXStateStore 51	PDFActionHideAddAnnotationName 110
PBXStateStore function 51	PDFActionHideAddAnnotationName function 110
PBXStroke 64	PDFActionNewGoToDestination 107
PBXStroke function 64	PDFActionNewGoToDestination function 107
PBXTextOut 67	PDFActionNewGoToRemote 108
PBXTextOut function 67	PDFActionNewGoToRemote function 108
PBXUnicodeTextOut 68	PDFActionNewHide 109
PBXUnicodeTextOut function 68	PDFActionNewHide function 109
PDF Acroform Level 73	PDFActionNewImportData 113
PDF Document Level 6	PDFActionNewImportData function 113
PDF Image Level 26	PDFActionNewJavaScript 118
PDF Library Exception Level 3	PDFActionNewJavaScript function 118
PDF Library Level 2	PDFActionNewJavaScriptStream 118
PDF Outline Level 84	PDFActionNewJavaScriptStream function 118
PDF Page Copier Level 24	PDFActionNewLaunch 109
PDF Page Level 18	PDFActionNewLaunch function 109
PDFAcroEditBoxSetAlign 74	PDFActionNewNamed 112
PDFAcroEditBoxSetAlign function 74	PDFActionNewNamed function 112
PDFAcroEditBoxSetMaxLen 74	PDFActionNewResetForm 116
PDFAcroEditBoxSetMaxLen function 74	PDFActionNewResetForm function 116
PDFAcroGetCount 73	PDFActionNewSubmitForm 114
PDFAcroGetCount function 73	PDFActionNewSubmitForm function 114
PDFAcroObjectAddAction 75	PDFActionNewThread 112
PDFAcroObjectAddAction function 75	PDFActionNewThread function 112
PDFAcroObjectAppendItem 75	PDFActionNewURI 111
PDFAcroObjectAppendItem function 75	PDFActionNewURI function 111
PDFAcroObjectSetBorder 76	PDFActionResetFormAddAnnotation 117
PDFAcroObjectSetBorder function 76	PDFActionResetFormAddAnnotation function 117
PDFAcroObjectSetCaption 76	PDFActionResetFormAddAnnotationName 117
PDFAcroObjectSetCaption function 76	PDFActionResetFormAddAnnotationName function 117

PDFActionSetNext 105	PDFDestinationFromExplit 105
PDFActionSetNext function 105	PDFDestinationFromExplit function 105
PDFActionSubmitFormAddAnnotation 115	PDFDestinationFromString 106
PDFActionSubmitFormAddAnnotation function 115	PDFDestinationFromString function 106
PDFActionSubmitFormAddAnnotationName 115	PDFDestinationHandle 180
PDFActionSubmitFormAddAnnotationName function 115	PDFDestinationHandle type 180
PDFAnnotationHandle 180	PDFDestinationNamedNew 106
PDFAnnotationHandle type 180	PDFDestinationNamedNew function 106
PDFAnnotationSetAction 99	PDFDestinationType 159
PDFAnnotationSetAction function 99	PDFDestinationType enumeration 159
PDFAnnotationSetAlphaBlending 100	PDFDocAppendPage 10
PDFAnnotationSetAlphaBlending function 100	PDFDocAppendPage function 10
PDFAnnotationSetBorderStyle 100	PDFDocAppendPage2 10
PDFAnnotationSetBorderStyle function 100	PDFDocAppendPage2 function 10
PDFAnnotationSetColor 100	PDFDocAppendSignatureFromBuffer 78
PDFAnnotationSetColor function 100	PDFDocAppendSignatureFromBuffer function 78
PDFAnnotationSetFlag 101	PDFDocAppendSignatureFromFile 79
PDFAnnotationSetFlag function 101	PDFDocAppendSignatureFromFile function 79
PDFAnnotationSetName 101	PDFDocAppendSignatureFromStream 80
PDFAnnotationSetName function 101	PDFDocAppendSignatureFromStream function 80
PDFAnnotationSetTitle 102	PDFDocCheckPassword 12
PDFAnnotationSetTitle function 102	PDFDocCheckPassword function 12
PDFBeadActionType 159	PDFDocClose 9
PDFBeadActionType enumeration 159	PDFDocClose function 9
PDFBeadGetNext 96	PDFDocCreate 6
PDFBeadGetNext function 96	PDFDocCreate function 6
PDFBeadGetPage 96	PDFDocGetInfo 16
PDFBeadGetPage function 96	PDFDocGetInfo function 16
PDFBeadGetPrev 97	PDFDocGetOutlineRoot 84
PDFBeadGetPrev function 97	PDFDocGetOutlineRoot function 84
PDFBeadGetRect 97	PDFDocGetPageCount 11
PDFBeadGetRect function 97	PDFDocGetPageCount function 11
PDFBeadGetThread 98	PDFDocGetPermission 13
PDFBeadGetThread function 98	PDFDocGetPermission function 13
PDFBeadHandle 180	PDFDocGetThread 93
PDFBeadHandle type 180	PDFDocGetThread function 93
PDFBeadSetRect 98	PDFDocGetThreadCount 93
PDFBeadSetRect function 98	PDFDocGetThreadCount function 93
PDFCopyPagesToDestinationDocument 24	PDFDocGetVersion 17
PDFCopyPagesToDestinationDocument function 24	PDFDocGetVersion function 17
PDFCosHandle 180	PDFDocIsCrypted 12
PDFCosHandle type 180	PDFDocIsCrypted function 12



PDFDocLoadFromBuffer 7	PDFExtGraphicStateSetAlphalsShape 36
PDFDocLoadFromBuffer function 7	PDFExtGraphicStateSetAlphalsShape function 36
PDFDocLoadFromFile 7	PDFExtGraphicStateSetAlphaStroke 37
PDFDocLoadFromFile function 7	PDFExtGraphicStateSetAlphaStroke function 37
PDFDocLoadFromStream 8	PDFExtGraphicStateSetBlendMode 37
PDFDocLoadFromStream function 8	PDFExtGraphicStateSetBlendMode function 37
PDFDocSaveToBuffer 9	PDFExtGraphicStateSetCTM 37
PDFDocSaveToBuffer function 9	PDFExtGraphicStateSetCTM function 37
PDFDocSaveToFile 8	PDFExtGraphicStateSetDashPattern 38
PDFDocSaveToFile function 8	PDFExtGraphicStateSetDashPattern function 38
PDFDocSaveToStream 8	PDFExtGraphicStateSetFlatness 38
PDFDocSaveToStream function 8	PDFExtGraphicStateSetFlatness function 38
PDFDocSetAutoLaunch 15	PDFExtGraphicStateSetLineCap 39
PDFDocSetAutoLaunch function 15	PDFExtGraphicStateSetLineCap function 39
PDFDocSetEMFBWImagesAsJBIG2 15	PDFExtGraphicStateSetLineJoin 39
PDFDocSetEMFBWImagesAsJBIG2 function 15	PDFExtGraphicStateSetLineJoin function 39
PDFDocSetEMFColorImagesAsJpeg 16	PDFExtGraphicStateSetLineWidth 40
PDFDocSetEMFColorImagesAsJpeg function 16	PDFExtGraphicStateSetLineWidth function 40
PDFDocSetInfo 17	PDFExtGraphicStateSetMitterLimit 40
PDFDocSetInfo function 17	PDFExtGraphicStateSetMitterLimit function 40
PDFDocSetJpegImageQuality 13	PDFExtGraphicStateSetOverprintFill 41
PDFDocSetJpegImageQuality function 13	PDFExtGraphicStateSetOverprintFill function 41
PDFDocSetLinearized 14	PDFExtGraphicStateSetOverprintMode 41
PDFDocSetLinearized function 14	PDFExtGraphicStateSetOverprintMode function 41
PDFDocSetPacked 14	PDFExtGraphicStateSetOverprintStroke 42
PDFDocSetPacked function 14	PDFExtGraphicStateSetOverprintStroke function 42
PDFDocSetRemoveUnUsed 15	PDFExtGraphicStateSetRenderingIntent 42
PDFDocSetRemoveUnUsed function 15	PDFExtGraphicStateSetRenderingIntent function 42
PDFDocSetUsedDC 13	PDFExtGraphicStateSetSmoothness 43
PDFDocSetUsedDC function 13	PDFExtGraphicStateSetSmoothness function 43
PDFDocSetVersion 17	PDFExtGraphicStateSetStrokeAdjustment 43
PDFDocSetVersion function 17	PDFExtGraphicStateSetStrokeAdjustment function 43
pdfdtExplicit enumeration member 159	PDFExtGraphicStateSetTextKnockout 44
pdfdtNamed enumeration member 159	PDFExtGraphicStateSetTextKnockout function 44
PDFExplicitDest 163	PDFFontStandardAppend 45
PDFExplicitDest type 163	PDFFontStandardAppend function 45
PDFExplicitDestType 160	PDFFontTrueTypeAppend 45
PDFExplicitDestType enumeration 160	PDFFontTrueTypeAppend function 45
PDFExtGraphicStateNew 35	PDFFontTrueTypeAppendFromFile 46
PDFExtGraphicStateNew function 35	PDFFontTrueTypeAppendFromFile function 46
PDFExtGraphicStateSetAlphaFill 36	PDFFontTrueTypeAppendFromStream 46
PDFExtGraphicStateSetAlphaFill function 36	PDFFontTrueTypeAppendFromStream function 46

---

PDFFontType1AppendFromFile 47	PDFImageLoadFromHandle 34
PDFFontType1AppendFromFile function 47	PDFImageLoadFromHandle function 34
PDFFontType1AppendFromStream 47	PDFImageLoadFromStream 32
PDFFontType1AppendFromStream function 47	PDFImageLoadFromStream function 32
PDFFreeDocumentConnection 25	PDFNamedActionType 163
PDFFreeDocumentConnection function 25	PDFNamedActionType type 163
PDFImageAppendToDoc 26	PDFOutlineAddNewChild 85
PDFImageAppendToDoc function 26	PDFOutlineAddNewChild function 85
PDFImageAppendToDocAsMask 27	PDFOutlineAddNewNext 85
PDFImageAppendToDocAsMask function 27	PDFOutlineAddNewNext function 85
PDFImageAppendToDocFromBuffer 32	PDFOutlineAddNewPrev 85
PDFImageAppendToDocFromBuffer function 32	PDFOutlineAddNewPrev function 85
PDFImageAppendToDocFromFile 27	PDFOutlineAddNewSibling 86
PDFImageAppendToDocFromFile function 27	PDFOutlineAddNewSibling function 86
PDFImageAppendToDocFromStream 28	PDFOutlineGetCount 86
PDFImageAppendToDocFromStream function 28	PDFOutlineGetCount function 86
PDFImageAppendToDocWithMask 28	PDFOutlineGetFirstChild 86
PDFImageAppendToDocWithMask function 28	PDFOutlineGetFirstChild function 86
PDFImageCreate 33	PDFOutlineGetLastChild 87
PDFImageCreate function 33	PDFOutlineGetLastChild function 87
PDFImageFree 28	PDFOutlineGetNext 87
PDFImageFree function 28	PDFOutlineGetNext function 87
PDFImageGetColorDevice 29	PDFOutlineGetParent 88
PDFImageGetColorDevice function 29	PDFOutlineGetParent function 88
PDFImageGetDepth 29	PDFOutlineGetPrev 88
PDFImageGetDepth function 29	PDFOutlineGetPrev function 88
PDFImageGetHeight 30	PDFOutlineHandle 181
PDFImageGetHeight function 30	PDFOutlineHandle type 181
PDFImageGetScanLine 30	PDFOutlineHasChildren 88
PDFImageGetScanLine function 30	PDFOutlineHasChildren function 88
PDFImageGetTIFFCountFromBuffer 33	PDFOutlineSetAction 89
PDFImageGetTIFFCountFromBuffer function 33	PDFOutlineSetAction function 89
PDFImageGetTIFFCountFromFile 30	PDFOutlineSetColor 89
PDFImageGetTIFFCountFromFile function 30	PDFOutlineSetColor function 89
PDFImageGetTIFFCountFromStream 31	PDFOutlineSetDestination 90
PDFImageGetTIFFCountFromStream function 31	PDFOutlineSetDestination function 90
PDFImageGetWidth 31	PDFOutlineSetExpanded 90
PDFImageGetWidth function 31	PDFOutlineSetExpanded function 90
PDFImageLoadFromBuffer 34	PDFOutlineSetFlags 90
PDFImageLoadFromBuffer function 34	PDFOutlineSetFlags function 90
PDFImageLoadFromFile 31	PDFOutlineSetTitle 91
PDFImageLoadFromFile function 31	PDFOutlineSetTitle function 91

---

PDFPageAddContent 18	PDFPageSetRotateAngle 23
PDFPageAddContent function 18	PDFPageSetRotateAngle function 23
PDFPageAppendAnnotationLinkWithAction 102	PDFSelectPageFromSourceDocument 25
PDFPageAppendAnnotationLinkWithAction function 102	PDFSelectPageFromSourceDocument function 25
PDFPageAppendAnnotationLinkWithDest 103	PDFThreadActionParam 161
PDFPageAppendAnnotationLinkWithDest function 103	PDFThreadActionParam structure 161
PDFPageAppendAnnotationStamp 103	PDFThreadActionParamP 164
PDFPageAppendAnnotationStamp function 103	PDFThreadActionParamP type 164
PDFPageAppendAnnotationText 104	PDFThreadActionType 161
PDFPageAppendAnnotationText function 104	PDFThreadActionType enumeration 161
PDFPageAppendCheckBox 80	PDFThreadAppendBead 93
PDFPageAppendCheckBox function 80	PDFThreadAppendBead function 93
PDFPageAppendComboBox 81	PDFThreadDelete 94
PDFPageAppendComboBox function 81	PDFThreadDelete function 94
PDFPageAppendEditBox 81	PDFThreadGetFirstBead 94
PDFPageAppendEditBox function 81	PDFThreadGetFirstBead function 94
PDFPageAppendListBox 82	PDFThreadGetInfo 95
PDFPageAppendListBox function 82	PDFThreadGetInfo function 95
PDFPageAppendPushButton 82	PDFThreadHandle 181
PDFPageAppendPushButton function 82	PDFThreadHandle type 181
PDFPageAppendRadioButton 83	PDFThreadNew 92
PDFPageAppendRadioButton function 83	PDFThreadNew function 92
PDFPageAppendSignatureBox 83	PDFThreadSetInfo 95
PDFPageAppendSignatureBox function 83	PDFThreadSetInfo function 95
PDFPageCreatePaintBox 19	pdfver10 enumeration member 162
PDFPageCreatePaintBox function 19	pdfver11 enumeration member 162
PDFPageCreatePaintBoxFromContent 19	pdfver12 enumeration member 162
PDFPageCreatePaintBoxFromContent function 19	pdfver13 enumeration member 162
PDFPageGetBox 20	pdfver14 enumeration member 162
PDFPageGetBox function 20	pdfver15 enumeration member 162
PDFPageGetContentCount 20	pdfver16 enumeration member 162
PDFPageGetContentCount function 20	pdfver17 enumeration member 162
PDFPageGetCosObject 20	PDFVersion 162
PDFPageGetCosObject function 20	PDFVersion enumeration 162
PDFPageGetRotateAngle 21	PPCallException 4
PDFPageGetRotateAngle function 21	PPCallException function 4
PDFPageInsertContent 21	ppComponentDepth 162
PDFPageInsertContent function 21	ppComponentDepth enumeration 162
PDFPageRemoveContent 22	PPDFExplicitDest 164
PDFPageRemoveContent function 22	PPDFExplicitDest type 164
PDFPageSetBox 22	ppFileOpenMode 165
PDFPageSetBox function 22	ppFileOpenMode type 165

PPGetLastError 4  
 PPGetLastError function 4  
 PPPopExceptionBuffer 5  
 PPPopExceptionBuffer function 5  
 PPPushExceptionBuffer 5  
 PPPushExceptionBuffer function 5

## R

Reset Form Action 116

## S

Stream Level 150  
 Structs, Records, Enums 158  
 Submit Action 114

## T

taBeadHandle enumeration member 159  
 taBeadIndex enumeration member 159  
 TAnnotationStampName 165  
 TAnnotationStampName type 165  
 TAnnotationTextName 166  
 TAnnotationTextName type 166  
 taThreadHandle enumeration member 161  
 taThreadIndex enumeration member 161  
 taThreadTitle enumeration member 161  
 TBorderStyle 167  
 TBorderStyle type 167  
 Text Operations 64  
 THighLightMode 167  
 THighLightMode type 167  
 Thread Action 111  
 Thread and Bead Level 92  
 Thread Operations 92  
 TImageCompressionType 181  
 TImageCompressionType type 181  
 TKeyValidType 181  
 TKeyValidType type 181  
 TPDFAcroEventType 168  
 TPDFAcroEventType type 168  
 TPDFAcroQuadding 168  
 TPDFAcroQuadding type 168  
 TPDFAcroType 169

TPDFAcroType type 169  
 TPDFBlendMode 170  
 TPDFBlendMode type 170  
 TPDFCheckBoxSign 170  
 TPDFCheckBoxSign type 170  
 TPDFCheckBoxStyle 171  
 TPDFCheckBoxStyle type 171  
 TPDFCMYKColor 172  
 TPDFCMYKColor type 172  
 TPDFColor 172  
 TPDFColor type 172  
 TPDFColorDevice 173  
 TPDFColorDevice type 173  
 TPDFEncodingType 181  
 TPDFEncodingType type 181  
 TPDFHorJust 173  
 TPDFHorJust type 173  
 TPDFInformation 182  
 TPDFInformation type 182  
 TPDFLineCap 173  
 TPDFLineCap type 173  
 TPDFLineJoin 174  
 TPDFLineJoin type 174  
 TPDFPageBoxType 174  
 TPDFPageBoxType type 174  
 TPDFPageOrientation 182  
 TPDFPageOrientation type 182  
 TPDFPageRotateAngle 175  
 TPDFPageRotateAngle type 175  
 TPDFPageSize 182  
 TPDFPageSize type 182  
 TPDFProtectionType 182  
 TPDFProtectionType type 182  
 TPDFRealPoint 175  
 TPDFRealPoint type 175  
 TPDFRect 176  
 TPDFRect type 176  
 TPDFRenderingIntents 176  
 TPDFRenderingIntents type 176  
 TPDFRGBColor 177  
 TPDFRGBColor type 177  
 TPDFStandardFont 183

TPDFStdandardFont type 183

TPDFVersion 177

TPDFVersion type 177

TPDFVertJust 178

TPDFVertJust type 178

## U

ULAtomToString 144

ULAtomToString function 144

ULClearAtoms 144

ULClearAtoms function 144

ULCMYKToColor 142

ULCMYKToColor function 142

ULExistsAtomForString 144

ULExistsAtomForString function 144

ULFileClose 146

ULFileClose function 146

ULFileGetChar 146

ULFileGetChar function 146

ULFileGetPosition 147

ULFileGetPosition function 147

ULFileGetSize 147

ULFileGetSize function 147

ULFileLookChar 148

ULFileLookChar function 148

ULFileOpen 148

ULFileOpen function 148

ULFileRead 148

ULFileRead function 148

ULFileSetPosition 149

ULFileSetPosition function 149

ULFileWrite 149

ULFileWrite function 149

ULGetAtomCount 145

ULGetAtomCount function 145

ULGrayToColor 142

ULGrayToColor function 142

ULRGBToColor 143

ULRGBToColor function 143

ULStreamClear 150

ULStreamClear function 150

ULStreamClose 151

ULStreamClose function 151

ULStreamCopyToStream 151

ULStreamCopyToStream function 151

ULStreamCustomNew 151

ULStreamCustomNew function 151

ULStreamFileHandleNew 152

ULStreamFileHandleNew function 152

ULStreamFileNew 152

ULStreamFileNew function 152

ULStreamGetPosition 153

ULStreamGetPosition function 153

ULStreamGetSize 153

ULStreamGetSize function 153

ULStreamLookChar 153

ULStreamLookChar function 153

ULStreamMemNew 154

ULStreamMemNew function 154

ULStreamMemReadOnlyNew 154

ULStreamMemReadOnlyNew function 154

ULStreamReadBuffer 155

ULStreamReadBuffer function 155

ULStreamReadChar 155

ULStreamReadChar function 155

ULStreamReadLine 156

ULStreamReadLine function 156

ULStreamSetPosition 156

ULStreamSetPosition function 156

ULStreamSetSize 156

ULStreamSetSize function 156

ULStreamWriteBuffer 157

ULStreamWriteBuffer function 157

ULStreamWriteChar 157

ULStreamWriteChar function 157

ULStringToAtom 145

ULStringToAtom function 145

Underline Level 142

URI Action 111

## V

VersyPDF Library 1

VSGetErrorStr 3

VSGetErrorStr function 3