VersyPDF Library

Table of Contents

Index

1 VersyPDF Library

What is VersyPDF Library? VersyPDF – it's library which allows to create 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 (⧉ see page 2)
DonePDFLibrary (⧉ see page 2)

2.1 InitPDFLibrary Function

```
PDFLibHandle InitPDFLibrary(char * username, char * key);
```

File

VSLibA.h

Returns

None.

Description

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

2.2 DonePDFLibrary Function

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

File

VSLibA.h

Parameters

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

Returns

None.

Description

Close VersyPDF library.

3 PDF Document Level

This level describes function interfaces for working with PDF document. All the functions a 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 3)
PDFDocLoadFromFile (↗ see page 4)
PDFDocLoadFromBuffer (↗ see page 4)
PDFDocLoadFromStream (↗ see page 4)
PDFDocSaveToFile (↗ see page 5)
PDFDocSaveToMemory (↗ see page 5)
PDFDocSaveToStream (↗ see page 6)
PDFDocClose (↗ see page 6)

3.1.1 PDFDocCreate Function

```
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 6).

3.1.2 PDFDocLoadFromFile Function

```
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 5).

3.1.3 PDFDocLoadFromBuffer Function

```
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 Stream.

See Also

PDFDocSaveToMemory (see page 5).

3.1.4 PDFDocLoadFromStream Function

```
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 6), PDFStreamHandle, PDFDocLoadFromFile (see page 4), PDFDocLoadFromBuffer (see page 4).

3.1.5 PDFDocSaveToFile Function

```
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 4).

3.1.6 PDFDocSaveToMemory Function

```
void * PDFDocSaveToMemory(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 4).

3.1.7 PDFDocSaveToStream Function

```
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 4), PDFStreamHandle, PDFDocSaveToMemory (⧉ see page 5), PDFDocSaveToFile (⧉ see page 5).

3.1.8 PDFDocClose Function

```
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 3).

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 7)
PDFDocAppendPage2 (↗ see page 7)
PDFDocGetPageCount (↗ see page 8)

3.2.1 PDFDocAppendPage Function

```
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 7).

3.2.2 PDFDocAppendPage2 Function

```
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 (↗ see page 233).
`TPDFPageOrientation Orientation`	[in] Orientation of Page, see TPDFPageOrientation (↗ see page 232).

Returns

Index of New Page in Document.

Description

Add Page in PDF Document.

See Also

PDFDocAppendPage (↗ see page 7), TPDFPageSize (↗ see page 233), TPDFPageOrientation (↗ see page 232).

3.2.3 PDFDocGetPageCount Function

```
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
PDFDocGetInfo (⧉ see page 8)
PDFDocSetInfo (⧉ see page 9)
PDFDocIsCrypted (⧉ see page 9)
PDFDocCheckPassword (⧉ see page 10)
PDFDocGetPermission (⧉ see page 10)
PDFDocSetSecurity (⧉ see page 10)
PDFDocSetEMFImagesAsJpeg (⧉ see page 11)
PDFDocSetUsedDC (⧉ see page 11)
PDFDocSetJpegImageQuality (⧉ see page 12)
PDFDocSetLinearized (⧉ see page 12)
PDFDocSetPacked (⧉ see page 12)
PDFDocSetRemoveUnUsed (⧉ see page 13)
PDFDocSetAutoLaunch (⧉ see page 13)

3.3.1 PDFDocGetInfo Function

```
PDFCosHandle PDFDocGetInfo(PDFDocHandle Doc, TPDFInformation Info);
```

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 9), TPDFInformation (see page 229).

3.3.2 PDFDocSetInfo Function

```
void PDFDocSetInfo(PDFDocHandle Doc, TPDFInformation Info, PDFCosHandle Value);
```

File

VSDocA.h

Parameters

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

Returns

None.

Description

Save information in Document description.

See Also

PDFDocGetInfo (see page 8), TPDFInformation (see page 229).

3.3.3 PDFDocIsCrypted Function

```
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 10), PDFDocSetSecurity (see page 10).

3.3.4 PDFDocCheckPassword Function

```
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 kvtInvalid then Password is invalid.

Description

Checking the PDF Document on Validity of Password.

See Also

PDFDocIsCrypted (↗ see page 9), PDFDocSetSecurity (↗ see page 10), TKeyValidType (↗ see page 215).

3.3.5 PDFDocGetPermission Function

```
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 (↗ see page 10).

3.3.6 PDFDocSetSecurity Function

```
void PDFDocSetSecurity(PDFDocHandle Doc, ppBool ProtectionEnabled, ppInt32 Permission,
TPDFProtectionType KeyLength, char * UserPassword, char * OwnerPassword);
```

File

VSDocA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`ppBool ProtectionEnabled`	[in] If true set document security. If false remove security from PDF document.
`ppInt32 Permission`	[in] Document Permission's Flags.
`TPDFProtectionType KeyLength`	[in] 40-bits or 128-bits Security Key Length.
`char * UserPassword`	[in] User Password text string.
`char * OwnerPassword`	[in] Owner Password text string.

Returns

None.

Description

Set PDF Document Security.

See Also

PDFDocIsCrypted (↗ see page 9), PDFDocCheckPassword (↗ see page 10), TPDFProtectionType (↗ see page 234).

3.3.7 PDFDocSetEMFImagesAsJpeg Function

```
void PDFDocSetEMFImagesAsJpeg(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 images stored in the EMF file will be saved as JPEG images. If value has set to false, all images stored in the EMF file will be saved with flate compression.

Returns

None.

Description

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

3.3.8 PDFDocSetUsedDC Function

```
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.

3.3.9 PDFDocSetJpegImageQuality Function

```
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.

3.3.10 PDFDocSetLinearized Function

```
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.

3.3.11 PDFDocSetPacked Function

```
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.

3.3.12 PDFDocSetRemoveUnUsed Function

```
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.

3.3.13 PDFDocSetAutoLaunch Function

```
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).

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
PDFPageGetCosObject (↗ see page 14)
PDFPageGetBox (↗ see page 14)
PDFPageSetBox (↗ see page 15)
PDFPageGetContentCount (↗ see page 15)
PDFPageAddContent (↗ see page 16)
PDFPageInsertContent (↗ see page 16)
PDFPageRemoveContent (↗ see page 17)
PDFPageGetRotateAngle (↗ see page 17)
PDFPageSetRotateAngle (↗ see page 17)
PDFPageCreatePaintBox (↗ see page 18)
PDFPageContentCreatePaintBox (↗ see page 18)

4.1 PDFPageGetCosObject Function

```
PDFCosHandle PDFPageGetCosObject(PDFDocHandle Doc, ppInt32 Page);
```

File

VSPageA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`ppInt32 Page`	[in] Page Index in PDF Document.

Returns

Page Object Handle.

Description

Get Page Object Handle.

4.2 PDFPageGetBox Function

```
TPDFRect PDFPageGetBox(PDFDocHandle Doc, ppInt32 Page, TPDFPageBoxType Type);
```

File

VSPageA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`ppInt32 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 15)

4.3 PDFPageSetBox Function

```
void PDFPageSetBox(PDFDocHandle Doc, ppInt32 Page, TPDFPageBoxType Type, TPDFRect Rect);
```

File

VSPageA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`ppInt32 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 14)

4.4 PDFPageGetContentCount Function

```
ppInt32 PDFPageGetContentCount(PDFDocHandle Doc, ppInt32 Page);
```

File

VSPageA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`ppInt32 Page`	[in] Page Index in PDF Document.

Returns

Page Content Streams Count.

Description

Get Page Content Streams Count.

4.5 PDFPageAddContent Function

```
ppInt32 PDFPageAddContent(PDFDocHandle Doc, ppInt32 Page);
```

File

VSPageA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`ppInt32 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 15)

4.6 PDFPageInsertContent Function

```
ppInt32 PDFPageInsertContent(PDFDocHandle Doc, ppInt32 Page, ppInt32 Index);
```

File

VSPageA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`ppInt32 Page`	[in] Page Index in PDF Document.
`ppInt32 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 17)

4.7 PDFPageRemoveContent Function

```
void PDFPageRemoveContent(PDFDocHandle Doc, ppInt32 Page, ppInt32 Index);
```

File

VSPageA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`ppInt32 Page`	[in] Page Index in PDF Document.
`ppInt32 Index`	[in] Content Index for deleting.

Returns

None.

Description

Remove Content from Page Content Streams according to index.

See Also

PDFPageInsertContent (see page 16)

4.8 PDFPageGetRotateAngle Function

```
TPDFPageRotateAngle PDFPageGetRotateAngle(PDFDocHandle Doc, ppInt32 Page);
```

File

VSPageA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`ppInt32 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 17)

4.9 PDFPageSetRotateAngle Function

```
void PDFPageSetRotateAngle(PDFDocHandle Doc, ppInt32 Page, TPDFPageRotateAngle Rotate);
```

File

VSPageA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`ppInt32 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 17)

4.10 PDFPageCreatePaintBox Function

```
PBXHandle PDFPageCreatePaintBox(PDFDocHandle Doc, ppInt32 Page, ppInt32 Resolution);
```

File

VSPageA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`ppInt32 Page`	[in] Page Index in PDF Document.
`ppInt32 Resolution`	[in] Points per inch (dpi).

Returns

Paint Box Handle.

Description

Create Paint Box for Last Page Content with defined resolution.

See Also

PDFPageContentCreatePaintBox (◰ see page 18)

4.11 PDFPageContentCreatePaintBox Function

```
PBXHandle PDFPageContentCreatePaintBox(PDFDocHandle Doc, ppInt32 Page, ppInt32 Index,
ppInt32 Resolution);
```

File

VSPageA.h

Parameters

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

`ppInt32 Page`	[in] Page Index in PDF Document.
`ppInt32 Index`	[in] Content's Index in Page Content Streams.
`ppInt32 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 18)

5 PDF Page Copier Level

Selection and copping page of one document to another is used with help of the TPDFDocumentConnection (⧉ see page 226) structuse and it's functions. Copping is executed in four steps:

1. Links creating between documents

2. Pages selection from source document.

3. Pages copping from source to destination document.

4. Links removing between documents.

Functions

Function
PDFCreateDocumentConnection (⧉ see page 20)
PDFCopyPagesToDestinationDocument (⧉ see page 20)
PDFSelectPageFromSourceDocument (⧉ see page 21)
PDFFreeDocumentConnection (⧉ see page 21)

5.1 PDFCreateDocumentConnection Function

```
void PDFCreateDocumentConnection(PDFDocHandle OldDocument, PDFDocHandle NewDocument,
PPDFDocumentConnection DocumentConnection);
```

File

VSPagesA.h

Parameters

Parameters	Description
`PDFDocHandle OldDocument`	[in] Source Document where from take pages.
`PDFDocHandle NewDocument`	[in] Destination Document where to put pages.
`PPDFDocumentConnection DocumentConnection`	[out] pointer to PPDFDocumentConnection (⧉ see page 193) structure.

Returns

None.

Description

Create source and destination documents connection and initialize page numbers array and its size.

See Also

TPDFDocumentConnection (⧉ see page 226)

5.2 PDFCopyPagesToDestinationDocument Function

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

File

VSPagesA.h

Parameters

Parameters	Description
`PPDFDocumentConnection DocumentConnection`	[out] pointer to PPDFDocumentConnection (see page 193) 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 (see page 226)

5.3 PDFSelectPageFromSourceDocument Function

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

File

VSPagesA.h

Parameters

Parameters	Description
`PPDFDocumentConnection DocumentConnection`	[out] pointer to PPDFDocumentConnection (see page 193) 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

TPDFDocumentConnection (see page 226)

5.4 PDFFreeDocumentConnection Function

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

File

VSPagesA.h

Parameters

Parameters	Description
`PPDFDocumentConnection DocumentConnection`	[out] pointer to PPDFDocumentConnection (see page 193) structure.

Returns

None.

Description

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

See Also

TPDFDocumentConnection (↗ see page 226)

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
PDFDocAppentImageFromJPEGFile (↗ see page 23)
PDFDocAppentImageFromJPEGMemoryBuffer (↗ see page 23)
PDFDocAppentImageFromPNGFile (↗ see page 24)
PDFDocAppentImageFromTIFFFile (↗ see page 24)
PDFDocAppentImageFromBitmapHandle (↗ see page 25)

6.1 PDFDocAppentImageFromJPEGFile Function

```
ppInt32 PDFDocAppentImageFromJPEGFile(PDFDocHandle Doc, char * filename,
TImageCompressionType ImageCompressionType);
```

File

VSImageA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current PDF document.
`char * filename`	[in] JPEG filename.
`TImageCompressionType ImageCompressionType`	[in] Image compression type.

Returns

Pdf image index.

Description

Append image to document from JPEG file.

6.2 PDFDocAppentImageFromJPEGMemoryBuffer Function

```
ppInt32 PDFDocAppentImageFromJPEGMemoryBuffer(PDFDocHandle Doc, int size, char * buffer,
TImageCompressionType ImageCompressionType);
```

File

VSImageA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current PDF document.
`int size`	[in] Size of JPEG image buffer.
`char * buffer`	[in] Pointer to JPEG image buffer.
`TImageCompressionType ImageCompressionType`	[in] Image compression type.

Returns

Pdf image index.

Description

Append image to document from JPEG buffer.

6.3 PDFDocAppentImageFromPNGFile Function

```
ppInt32 PDFDocAppentImageFromPNGFile(PDFDocHandle Doc, char * filename,
TImageCompressionType ImageCompressionType);
```

File

VSImageA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current PDF document.
`char * filename`	[in] PNG filename.
`TImageCompressionType ImageCompressionType`	[in] Image compression type.

Returns

Pdf image index.

Description

Append image to document from PNG file.

6.4 PDFDocAppentImageFromTIFFFile Function

```
ppInt32 PDFDocAppentImageFromTIFFFile(PDFDocHandle Doc, char * filename, ppInt32 *
ImageIndex, TImageCompressionType ImageCompressionType);
```

File

VSImageA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current PDF document.
`char * filename`	[in] TIFF filename.
`ppInt32 * ImageIndex`	[in, out] Image index in TIFF file (if ImageIndex = -1 returns count of images in file)
`TImageCompressionType ImageCompressionType`	[in] Image compression type.

Returns

Pdf image index.

Description

Append image to document from TIFF file.

6.5 PDFDocAppentImageFromBitmapHandle Function

```
ppInt32 PDFDocAppentImageFromBitmapHandle(PDFDocHandle Doc, HBITMAP Bitmap,
TImageCompressionType ImageCompressionType);
```

File

VSImageA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current PDF document.
`HBITMAP Bitmap`	[in] Bitmap handle.
`TImageCompressionType ImageCompressionType`	[in] Image compression type.

Returns

Pdf image index.

Description

Append image to document 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
PDFExtGraphicStateCreate (↗ see page 26)
PDFExtGraphicStateSetAlphaFill (↗ see page 26)
PDFExtGraphicStateSetAlphaIsShape (↗ see page 27)
PDFExtGraphicStateSetAlphaStroke (↗ see page 27)
PDFExtGraphicStateSetBlendMode (↗ see page 28)
PDFExtGraphicStateSetDashPattern (↗ see page 28)
PDFExtGraphicStateSetFlatness (↗ see page 29)
PDFExtGraphicStateSetLineCap (↗ see page 29)
PDFExtGraphicStateSetLineJoin (↗ see page 29)
PDFExtGraphicStateSetLineWidth (↗ see page 30)
PDFExtGraphicStateSetMitterLimit (↗ see page 30)
PDFExtGraphicStateSetOverprintFill (↗ see page 31)
PDFExtGraphicStateSetOverprintMode (↗ see page 31)
PDFExtGraphicStateSetOverprintStroke (↗ see page 32)
PDFExtGraphicStateSetRenderingIntent (↗ see page 32)
PDFExtGraphicStateSetSmoothness (↗ see page 33)
PDFExtGraphicStateSetStrokeAdjustment (↗ see page 33)
PDFExtGraphicStateSetTextKnockout (↗ see page 34)

7.1 PDFExtGraphicStateCreate Function

```
ppInt32 PDFExtGraphicStateCreate(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

```
void PDFExtGraphicStateSetAlphaFill(PDFDocHandle Doc, ppInt32 GState, ppReal AlphaFill);
```

File

VSGStateA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in, out] Current PDF document.
`ppInt32 GState`	[in] Graphic state index.
`ppReal AlphaFill`	[in] Alpha fill.

Returns

None.

Description

Setting alpha fill for current graphic state.

7.3 PDFExtGraphicStateSetAlphaIsShape Function

```
void PDFExtGraphicStateSetAlphaIsShape(PDFDocHandle Doc, ppInt32 GState, ppBool
AlphaIsShape);
```

File

VSGStateA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in, out] Current PDF document.
`ppInt32 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

```
void PDFExtGraphicStateSetAlphaStroke(PDFDocHandle Doc, ppInt32 GState, ppReal AlphaStroke);
```

File

VSGStateA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in, out] Current PDF document.
`ppInt32 GState`	[in] Graphic state index.
`AlphaIsShape`	[in] Alpha stroke.

Returns

None.

Description

Setting alpha stroke for current graphic state.

7.5 PDFExtGraphicStateSetBlendMode Function

```
void PDFExtGraphicStateSetBlendMode(PDFDocHandle Doc, ppInt32 GState, PDFBlendMode
BlendMode);
```

File

VSGStateA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in, out] Current PDF document.
`ppInt32 GState`	[in] Graphic state index.
`PDFBlendMode BlendMode`	[in] Blending mode.

Returns

None.

Description

Setting blending mode for current graphic state.

7.6 PDFExtGraphicStateSetDashPattern Function

```
void PDFExtGraphicStateSetDashPattern(PDFDocHandle Doc, ppInt32 GState, PDFCosHandle
DashPattern);
```

File

VSGStateA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in, out] Current PDF document.
`ppInt32 GState`	[in] Graphic state index.
`PDFCosHandle DashPattern`	[in] Dash pattern.

Returns

None.

Description

Setting dash pattern for current graphic state.

7.7 PDFExtGraphicStateSetFlatness Function

```
void PDFExtGraphicStateSetFlatness(PDFDocHandle Doc, ppInt32 GState, ppReal Flatness);
```

File

VSGStateA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in, out] Current PDF document.
`ppInt32 GState`	[in] Graphic state index.
`ppReal Flatness`	[in] Graphic state flatness.

Returns

None.

Description

Setting flatness for current graphic state.

7.8 PDFExtGraphicStateSetLineCap Function

```
void PDFExtGraphicStateSetLineCap(PDFDocHandle Doc, ppInt32 GState, ppInt32 LineCap);
```

File

VSGStateA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in, out] Current PDF document.
`ppInt32 GState`	[in] Graphic state index.
`ppInt32 LineCap`	[in] Line cap.

Returns

None.

Description

Setting line cap for current graphic state.

7.9 PDFExtGraphicStateSetLineJoin Function

```
void PDFExtGraphicStateSetLineJoin(PDFDocHandle Doc, ppInt32 GState, ppInt32 LineJoin);
```

File

VSGStateA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in, out] Current PDF document.

`ppInt32 GState`	[in] Graphic state index.
`ppInt32 LineJoin`	[in] Line join.

Returns

None.

Description

Setting line cap for current graphic state.

7.10 PDFExtGraphicStateSetLineWidth Function

```
void PDFExtGraphicStateSetLineWidth(PDFDocHandle Doc, ppInt32 GState, ppReal LineWidth);
```

File

VSGStateA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in, out] Current PDF document.
`ppInt32 GState`	[in] Graphic state index.
`ppReal LineWidth`	[in] Line width.

Returns

None.

Description

Setting line width for current graphic state.

7.11 PDFExtGraphicStateSetMitterLimit Function

```
void PDFExtGraphicStateSetMitterLimit(PDFDocHandle Doc, ppInt32 GState, ppReal MitterLimit);
```

File

VSGStateA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in, out] Current PDF document.
`ppInt32 GState`	[in] Graphic state index.
`ppReal MitterLimit`	[in] Mitter limit.

Returns

None.

Description

Setting mitter limit for current graphic state.

7.12 PDFExtGraphicStateSetOverprintFill Function

```
void PDFExtGraphicStateSetOverprintFill(PDFDocHandle Doc, ppInt32 GState, ppBool
OverprintFill);
```

File

VSGStateA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in, out] Current PDF document.
`ppInt32 GState`	[in] Graphic state index.
`ppBool OverprintFill`	[in] Overprint fill.

Returns

None.

Description

Setting overprint fill for current graphic state.

7.13 PDFExtGraphicStateSetOverprintMode Function

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

File

VSGStateA.h

Parameters

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

Returns

None.

Description

Setting overprint mode for current graphic state.

7.14 PDFExtGraphicStateSetOverprintStroke Function

```
void PDFExtGraphicStateSetOverprintStroke(PDFDocHandle Doc, ppInt32 GState, ppBool
OverprintStroke);
```

File

VSGStateA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in, out] Current PDF document.
`ppInt32 GState`	[in] Graphic state index.
`ppBool OverprintStroke`	[in] Overprint stroke.

Returns

None.

Description

Setting overprint stroke for current graphic state.

7.15 PDFExtGraphicStateSetRenderingIntent Function

```
void PDFExtGraphicStateSetRenderingIntent(PDFDocHandle Doc, ppInt32 GState,
PDFRenderingIntents Intent);
```

File

VSGStateA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in, out] Current PDF document.
`ppInt32 GState`	[in] Graphic state index.
`PDFRenderingIntents Intent`	[in] Rendering intents.

Returns

None.

Description

Setting graphic state rendering intents.

7.16 PDFExtGraphicStateSetSmoothness Function

```
void PDFExtGraphicStateSetSmoothness(PDFDocHandle Doc, ppInt32 GState, ppReal Smoothness);
```

File

VSGStateA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in, out] Current PDF document.
`ppInt32 GState`	[in] Graphic state index.
`ppReal Smoothness`	[in] Smoothness value.

Returns

None.

Description

Setting smoothness for current graphic state.

7.17 PDFExtGraphicStateSetStrokeAdjustment Function

```
void PDFExtGraphicStateSetStrokeAdjustment(PDFDocHandle Doc, ppInt32 GState, ppBool
StrokeAdjustment);
```

File

VSGStateA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in, out] Current PDF document.
`ppInt32 GState`	[in] Graphic state index.
`ppBool StrokeAdjustment`	[in] Stroke adjustment.

Returns

None.

Description

Setting stroke adjustment for current graphic state.

7.18 PDFExtGraphicStateSetTextKnockout Function

```
void PDFExtGraphicStateSetTextKnockout(PDFDocHandle Doc, ppInt32 GState, ppBool
TextKnockout);
```

File

VSGStateA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in, out] Current PDF document.
`ppInt32 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
PDFFontAppend14Standard (↗ see page 35)
PDFFontAppendTrueType (↗ see page 35)
PDFFontAppendTrueTypeFromFile (↗ see page 36)
PDFFontAppendTrueTypeFromStream (↗ see page 36)
PDFFontAppendType1FromFile (↗ see page 37)
PDFFontAppendType1FromStream (↗ see page 37)

8.1 PDFFontAppend14Standard Function

```
ppInt32 PDFFontAppend14Standard(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 PDFFontAppendTrueType Function

```
ppInt32 PDFFontAppendTrueType(PDFDocHandle Doc, char * fontname, ppBool Bold, ppBool
Italic);
```

File

VSFontA.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 PDFFontAppendTrueTypeFromFile Function

```
ppInt32 PDFFontAppendTrueTypeFromFile(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 true type font to document from file.

8.4 PDFFontAppendTrueTypeFromStream Function

```
ppInt32 PDFFontAppendTrueTypeFromStream(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 true type font to document from stream.

8.5 PDFFontAppendType1FromFile Function

```
ppInt32 PDFFontAppendType1FromFile(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 PDFFontAppendType1FromStream Function

```
ppInt32 PDFFontAppendType1FromStream(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 38)
PBXSetColor (↗ see page 39)
PBXSetDash (↗ see page 39)
PBXSetFillColor (↗ see page 39)
PBXSetFlatness (↗ see page 40)
PBXSetGState (↗ see page 40)
PBXSetLineWidth (↗ see page 40)
PBXSetStrokeColor (↗ see page 41)
PBXStateRestore (↗ see page 41)
PBXStateStore (↗ see page 41)
PBXSetLineCap (↗ see page 42)
PBXSetLineJoin (↗ see page 42)
PBXSetMiterLimit (↗ see page 43)

9.1.1 PBXNoDash Function

```
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

```
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

```
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

```
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

```
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 PBXSetGState Function

```
void PBXSetGState(PBXHandle PaintBox, ppInt32 Index);
```

File

VSCanvasA.h

Parameters

Parameters	Description
`PBXHandle PaintBox`	[in] PaintBox object handle
`ppInt32 Index`	[in] Index of the created extended graphic state returned from function PDFExtGraphicStateCreate (⧉ see page 26)

Returns

None.

Description

This function sets extended graphic state which can be created with function PDFExtGraphicStateCreate (⧉ see page 26).

9.1.7 PBXSetLineWidth Function

```
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.

9.1.8 PBXSetStrokeColor Function

```
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.

9.1.9 PBXStateRestore Function

```
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.

9.1.10 PBXStateStore Function

```
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.

9.1.11 PBXSetLineCap Function

```
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 230) enumeration that specifies the line cap.

Returns

None.

Description

Specifies line cap style.

9.1.12 PBXSetLineJoin Function

```
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 231) 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.

9.1.13 PBXSetMiterLimit Function

```
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.

9.2 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 43)
PBXArc (↗ see page 44)
PBXArc2 (↗ see page 44)
PBXEllipse (↗ see page 45)
PBXCircle (↗ see page 45)
PBXPie (↗ see page 46)
PBXPie2 (↗ see page 47)
PBXCurveTo (↗ see page 47)
PBXLineTo (↗ see page 48)
PBXMoveTo (↗ see page 48)
PBXRectangle (↗ see page 48)
PBXRectRotated (↗ see page 49)
PBXRoundRect (↗ see page 49)
PBXClosePath (↗ see page 50)

9.2.1 PBXNewPath Function

```
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

```
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 45) 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

```
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

```
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 44). 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

```
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 "Rin the counter-clock-wise direction.

Notes

If you need a circle drawn in the clock-wise direction, please use PBXArc (see page 44). 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

```
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

```
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

```
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

```
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

```
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

```
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

```
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

```
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

```
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 50)
PBXDrawTextBox (↗ see page 51)
PBXEoClip (↗ see page 51)
PBXEoFill (↗ see page 52)
PBXEoFillAndStroke (↗ see page 52)
PBXFill (↗ see page 52)
PBXFillAndStroke (↗ see page 53)
PBXStroke (↗ see page 53)

9.3.1 PBXClip Function

```
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 PBXDrawTextBox Function

```
void PBXDrawTextBox(PBXHandle PaintBox, TPDFTextBox TextBox);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PBXHandle PaintBox`	[in] Created PaintBox for drawing
`TPDFTextBox TextBox`	[in] TextBox structure to create text object

Returns

None.

Description

Draw TextBox on PaintBox Content

9.3.3 PBXEoClip Function

```
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.

9.3.4 PBXEoFill Function

```
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.

9.3.5 PBXEoFillAndStroke Function

```
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.

9.3.6 PBXFill Function

```
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.

9.3.7 PBXFillAndStroke Function

```
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.

9.3.8 PBXStroke Function

```
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 54)
PBXGetUnicodeWidth (↗ see page 54)
PBXSetActiveFont (↗ see page 55)
PBXSetActiveFontWithCharset (↗ see page 55)
PBXSetCharacterSpacing (↗ see page 55)
PBXSetHorizontalScaling (↗ see page 56)
PBXSetTextRenderingMode (↗ see page 56)
PBXSetWordSpacing (↗ see page 57)
PBXTextOut (↗ see page 57)
PBXUnicodeTextOut (↗ see page 58)

9.4.1 PBXGetTextWidth Function

```
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

```
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 PBXSetActiveFont Function

```
void PBXSetActiveFont(PBXHandle PaintBox, ppInt32 Index, ppReal FontSize, ppBool UnderLine,
ppBool StrikeOut);
```

File

VSCanvasA.h

Parameters

Parameters	Description
`PBXHandle PaintBox`	[in] PaintBox object handle
`ppInt32 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.

9.4.4 PBXSetActiveFontWithCharset Function

```
void PBXSetActiveFontWithCharset(PBXHandle PaintBox, ppInt32 Index, ppReal FontSize, ppUns8
Charset, ppBool UnderLine, ppBool StrikeOut);
```

File

VSCanvasA.h

Parameters

Parameters	Description
`PBXHandle PaintBox`	[in] PaintBox object handle
`ppInt32 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.

9.4.5 PBXSetCharacterSpacing Function

```
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.

9.4.6 PBXSetHorizontalScaling Function

```
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 (%).

9.4.7 PBXSetTextRenderingMode Function

```
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.

9.4.8 PBXSetWordSpacing Function

```
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.

9.4.9 PBXTextOut Function

```
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.

9.4.10 PBXUnicodeTextOut Function

```
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.

9.5 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 58)
PBXGetWidth (⧉ see page 59)
PBXPlayMetaFile (⧉ see page 59)
PBXShowImage (⧉ see page 60)
PBXClose (⧉ see page 60)

9.5.1 PBXGetHeight Function

```
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

```
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

```
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 PBXShowImage Function

```
void PBXShowImage(PBXHandle PaintBox, ppInt32 Index, ppReal X, ppReal Y, ppReal Width,
ppReal Height, ppReal Angle);
```

File

VSCanvasA.h

Parameters

Parameters	Description
`PBXHandle PaintBox`	[in] PaintBox object handle
`ppInt32 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

9.5.5 PBXClose Function

```
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.

10 PDF Acroform Level

Acroforms - are PDF documents objects such as pushbutton, checkbox, radiobuttons, editbox, combobox, listbox or digtital 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
PDFAcroCheckBoxInDocument (↗ see page 61)
PDFAcroComboBoxInDocument (↗ see page 62)
PDFAcroCosObjectGet (↗ see page 62)
PDFAcroEditBoxInDocument (↗ see page 63)
PDFAcroItemsBoxAdd (↗ see page 63)
PDFAcroListBoxInDocument (↗ see page 63)
PDFAcroObjectAddAction (↗ see page 64)
PDFAcroPushButtonInDocument (↗ see page 64)
PDFAcroRadioButtonInDocument (↗ see page 65)
PDFAcroSignatureInDocument (↗ see page 65)
PDFAcroSignDocument (↗ see page 66)
PDFAcroGetCount (↗ see page 66)
PDFAcroGetKeyByName (↗ see page 66)
PDFAcroGetKeyByValue (↗ see page 67)
PDFAcroGetNameByKey (↗ see page 67)
PDFAcroGetNameByValue (↗ see page 68)
PDFAcroGetTypeByKey (↗ see page 68)
PDFAcroGetTypeByName (↗ see page 69)
PDFAcroGetTypeByValue (↗ see page 69)
PDFAcroGetValueByKey (↗ see page 69)
PDFAcroGetValueByName (↗ see page 70)
PDFAcroSetNameByKey (↗ see page 70)
PDFAcroSetNameByName (↗ see page 71)
PDFAcroSetNameByValue (↗ see page 71)
PDFAcroSetValueByKey (↗ see page 72)
PDFAcroSetValueByName (↗ see page 72)
PDFAcroSetValueByValue (↗ see page 72)

10.1 PDFAcroCheckBoxInDocument Function

```
ppInt32 PDFAcroCheckBoxInDocument(PDFDocHandle Doc, ppInt32 PageIndex, TPDFCheckBox
CheckBox);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Destination Document.
`ppInt32 PageIndex`	[in] Page index (as from 0) where CheckBox will be stated.

`TPDFCheckBox CheckBox`	[in] CheckBox Structure which describes all needed fields (see TPDFCheckBox (⧉ see page 223)).

Returns

Index of acroform object in document.

Description

Sets CheckBox on Page Content for item selection.

10.2 PDFAcroComboBoxInDocument Function

```
ppInt32 PDFAcroComboBoxInDocument(PDFDocHandle Doc, ppInt32 PageIndex, TPDFComboBox
ComboBox);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Destination Document.
`ppInt32 PageIndex`	[in] Page index (as from 0) where ComboBox will be stated.
`TPDFComboBox ComboBox`	[in] ComboBox Structure which describes all needed fields (see TPDFComboBox (⧉ see page 225)).

Returns

Index of acroform object in document.

Description

Sets ComboBox on Page Content for item selection.

See Also

Array of text items(Opt) in ComboBox structure will be filled with function PDFAcroItemsBoxAdd (⧉ see page 63).

10.3 PDFAcroCosObjectGet Function

```
PDFCosHandle PDFAcroCosObjectGet(PDFDocHandle Doc, ppInt32 Index);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current document.
`ppInt32 Index`	[in] Index of acroform object in document.

Returns

Acro Object Handle in PDF Document by Index.

Description

Gets Acro Form Object from Document with Index as Acro Object Handle.

10.4 PDFAcroEditBoxInDocument Function

```
ppInt32 PDFAcroEditBoxInDocument(PDFDocHandle Doc, ppInt32 PageIndex, TPDFEditBox EditBox);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Destination Document.
`ppInt32 PageIndex`	[in] Page index (as from 0) where EditBox will be stated.
`TPDFEditBox EditBox`	[in] EditBox Structure which describes all needed fields (see TPDFEditBox (↗ see page 227)).

Returns

Index of acroform object in document.

Description

Sets EditBox on Page Content for text entering

10.5 PDFAcroItemsBoxAdd Function

```
void PDFAcroItemsBoxAdd(PDFDocHandle Doc, ppInt32 AcroObjectIndex, char * String,
TPDFItemsBox ItemsBox);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Destination document.
`ppInt32 AcroObjectIndex`	[in] Index of acroform object in document ComboBox or ListBox.
`char * String`	[in] Text string for addition in items array
`TPDFItemsBox ItemsBox`	[in] ComboBox or ListBox Structure which describes all needed fields (see TPDFItemsBox (↗ see page 230)).

Returns

None.

Description

Adds Text Item to items array of ComboBox or ListBox.

See Also

PDFAcroComboBoxInDocument (↗ see page 62), PDFAcroListBoxInDocument (↗ see page 63)

10.6 PDFAcroListBoxInDocument Function

```
ppInt32 PDFAcroListBoxInDocument(PDFDocHandle Doc, ppInt32 PageIndex, TPDFListBox ListBox);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Destination Document.
`ppInt32 PageIndex`	[in] Page index (as from 0) where ListBox will be stated.
`TPDFListBox ListBox`	[in] ListBox Structure which describes all needed fields (see TPDFListBox (↗ see page 231)).

Returns

Index of acroform object in document.

Description

Sets ListBox on Page Content for item selection.

See Also

Array of text items(Opt) in ListBox structure will be filled with function PDFAcroItemsBoxAdd (↗ see page 63).

10.7 PDFAcroObjectAddAction Function

```
void PDFAcroObjectAddAction(PDFDocHandle Doc, ppInt32 AcroObjectIndex, PDFActionHandle
Action, TPDFAcroEventType Type);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Destination document.
`ppInt32 AcroObjectIndex`	[in] Index of acroform object in document.
`PDFActionHandle Action`	[in] Handle on action for include to trigger.
`TPDFAcroEventType Type`	[in] Type of event on Control for Action

Returns

None.

Description

Adds Action OnEvent in Acro Form Object (Control).

See Also

TPDFAcroEventType (↗ see page 221), Actions

10.8 PDFAcroPushButtonInDocument Function

```
ppInt32 PDFAcroPushButtonInDocument(PDFDocHandle Doc, ppInt32 PageIndex, TPDFPushButton
PushButton);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Destination Document.
`ppInt32 PageIndex`	[in] Page index (as from 0) where PushButton will be stated.
`TPDFPushButton PushButton`	[in] PushButton Structure which describes all needed fields (see TPDFPushButton (⧉ see page 234)).

Returns

Index of acroform object in document.

Description

Sets PushButton on Page Content for action control.

10.9 PDFAcroRadioButtonInDocument Function

```
ppInt32 PDFAcroRadioButtonInDocument(PDFDocHandle Doc, ppInt32 PageIndex, char * GroupName,
TPDFRadioButton RadioButton);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Destination Document.
`ppInt32 PageIndex`	[in] Page index (as from 0) where RadioButton will be stated.
`char * GroupName`	[in] Name of Radio Buttons Group to which RadioButton will be linked
`TPDFRadioButton RadioButton`	[in] RadioButton Structure which describes all needed fields (see TPDFRadioButton (⧉ see page 235)).

Returns

Index of acroform object in document.

Description

Sets RadioButton on Page Content for item selection.

10.10 PDFAcroSignatureInDocument Function

```
ppInt32 PDFAcroSignatureInDocument(PDFDocHandle Doc, ppInt32 PageIndex, TPDFSignature
Signature);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Destination Document
`ppInt32 PageIndex`	[in] Page index (as from 0) where Digital Signature will be stated.
`TPDFSignature Signature`	[in] Digital Signature Structure which describes all needed fields (see TPDFSignature (⧉ see page 237)).

Returns

Index of acroform object in document.

Description

Sets Digital Signature Blank Field on Page Content for further sign (filling)

10.11 PDFAcroSignDocument Function

```
void PDFAcroSignDocument(PDFDocHandle Doc, ppInt32 PageIndex, TPDFDocumentSignature
DocumentSignature);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Destination Document
`ppInt32 PageIndex`	[in] Page index (as from 0) where Digital Signature will be linked.
`TPDFDocumentSignature DocumentSignature`	[in] Digital Signature Structure which describes owner, reason and information about PFX file (see TPDFDocumentSignature (see page 226)).

Returns

None.

Description

Sets Invisible Filled Digital Signature on Page. This signature is in PFX Digital Signature File.

10.12 PDFAcroGetCount Function

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

File

VSAcroObjects.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.

10.13 PDFAcroGetKeyByName Function

```
ppInt32 PDFAcroGetKeyByName(PDFDocHandle Doc, char * Name);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Source Document.
`char * Name`	[in] Full Name of Acro Object, where partial names are separated by point, for example 'button.2'

Returns

Integer, Index of object(as from 0), -1 when failure.

Description

Gets Key(Index) of Acro Form Object in Document according to Full Name.

See Also

PDFAcroGetNameByKey (see page 67)

10.14 PDFAcroGetKeyByValue Function

```
ppInt32 PDFAcroGetKeyByValue(PDFDocHandle Doc, char * Value);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Source Document.
`char * Value`	[in] Value of Acro Object, text string.

Returns

Integer, Index of object(as from 0), -1 when failure.

Description

Gets Key(Index) of Acro Form Object in Document according to its Value.

See Also

PDFAcroGetValueByKey (see page 69)

10.15 PDFAcroGetNameByKey Function

```
char * PDFAcroGetNameByKey(PDFDocHandle Doc, ppUns32 Key);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Source Document.
`ppUns32 Key`	[in] Key as from 0. Object number in AcroForm.

Returns

Text string (where partial names are separated by point).

Description

Gets Full name of Acro Form Object in Document according to Key(Index).

See Also

PDFAcroSetNameByKey (↗ see page 70)

10.16 PDFAcroGetNameByValue Function

```
char * PDFAcroGetNameByValue(PDFDocHandle Doc, char * Value);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Source Document.
`char * Value`	[in] Value of Acro Object, text string.

Returns

Text string (where partial names are separated by point).

Description

Gets Full name of Acro Form Object in Document according to its Value.

See Also

PDFAcroSetNameByValue (↗ see page 71)

10.17 PDFAcroGetTypeByKey Function

```
TPDFAcroType PDFAcroGetTypeByKey(PDFDocHandle Doc, ppUns32 Key);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Source Document.
`ppUns32 Key`	[in] Key as from 0. Object number in AcroForm.

Returns

Typed value, for example PushButton, ComboBox, etc.

Description

Gets Type of Acro Form Object in Document according to Key(Index).

See Also

PDFAcroType

10.18 PDFAcroGetTypeByName Function

```
TPDFAcroType PDFAcroGetTypeByName(PDFDocHandle Doc, char * Name);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Source Document.
`char * Name`	[in] Full Name of Acro Object, where partial names are separated by point, for example 'button.2'

Returns

Typed value, for example PushButton, ComboBox, etc.

Description

Gets Type of Acro Form Object in Document according to its Full Name.

See Also

PDFAcroType

10.19 PDFAcroGetTypeByValue Function

```
TPDFAcroType PDFAcroGetTypeByValue(PDFDocHandle Doc, char * Value);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Source Document.
`char * Value`	[in] Value of Acro Object, text string.

Returns

Typed value, for example PushButton, ComboBox, etc.

Description

Gets Type of Acro Form Object in Document according to its Value.

See Also

PDFAcroType

10.20 PDFAcroGetValueByKey Function

```
char * PDFAcroGetValueByKey(PDFDocHandle Doc, ppUns32 Key);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Source Document.
`ppUns32 Key`	[in] Key as from 0. Object number in AcroForm.

Returns

Text string

Description

Gets Value of Acro Form Object in Document according to Key(Index).

See Also

PDFAcroSetValueByKey (↗ see page 72)

10.21 PDFAcroGetValueByName Function

```
char * PDFAcroGetValueByName(PDFDocHandle Doc, char * Name);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Source Document.
`char * Name`	[in] Full Name of Acro Object, where partial names are separated by point, for example 'button.2'

Returns

Text string

Description

Gets Value of Acro Form Object in Document according to its Full Name.

See Also

PDFAcroSetValueByName (↗ see page 72)

10.22 PDFAcroSetNameByKey Function

```
void PDFAcroSetNameByKey(PDFDocHandle Doc, ppUns32 Key, char * Name);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Source Document.
`ppUns32 Key`	[in] Key as from 0. Object number in AcroForm.
`char * Name`	[in] Partial Name of Acro Object.

Returns

None.

Description

Sets name of Acro Form Object in Document according to Key(Index).

See Also

PDFAcroGetNameByKey (see page 67)

10.23 PDFAcroSetNameByName Function

```
void PDFAcroSetNameByName(PDFDocHandle Doc, char * OldName, char * NewName);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Source Document.
`char * OldName`	[in] Full Name of Acro Object, where partial names are separated by point, for example 'button.2'
`char * NewName`	[in] Partial Name of Acro Object.

Returns

None.

Description

Sets name of Acro Form Object in Document according to Full Name.

10.24 PDFAcroSetNameByValue Function

```
void PDFAcroSetNameByValue(PDFDocHandle Doc, char * Value, char * Name);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Source Document.
`char * Value`	[in] Value of Acro Object, text string.
`char * Name`	[in] Partial Name of Acro Object.

Returns

None.

Description

Sets name of Acro Form Object in Document according to its Value.

See Also

PDFAcroGetNameByValue (see page 68)

10.25 PDFAcroSetValueByKey Function

```
void PDFAcroSetValueByKey(PDFDocHandle Doc, ppUns32 Key, char * Value);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Source Document.
`ppUns32 Key`	[in] Key as from 0. Object number in AcroForm.
`char * Value`	[in] Value of Acro Object, text string.

Returns

None.

Description

Sets Value of Acro Form Object in Document according to Key(Index).

See Also

PDFAcroGetValueByKey (⧉ see page 69)

10.26 PDFAcroSetValueByName Function

```
void PDFAcroSetValueByName(PDFDocHandle Doc, char * Name, char * Value);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Source Document.
`char * Value`	[in] Value of Acro Object, text string.
`Key`	[in] Key as from 0. Object number in AcroForm.

Returns

None.

Description

Sets Value of Acro Form Object in Document according to its Name.

See Also

PDFAcroGetValueByName (⧉ see page 70)

10.27 PDFAcroSetValueByValue Function

```
void PDFAcroSetValueByValue(PDFDocHandle Doc, char * OldValue, char * NewValue);
```

File

VSAcroObjects.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Source Document.
`char * OldValue`	[in] Old Value of Acro Object, text string.
`char * NewValue`	[in] New Value of Acro Object, text string.

Returns

None.

Description

Sets New Value of Acro Form Object in Document according to its Old Value.

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
PDFOutlineAddChild (↗ see page 74)
PDFOutlineAddNewChild (↗ see page 75)
PDFOutlineAddNewNext (↗ see page 75)
PDFOutlineAddNewPrev (↗ see page 75)
PDFOutlineAddNewSibling (↗ see page 76)
PDFOutlineAddNext (↗ see page 76)
PDFOutlineAddPrev (↗ see page 77)
PDFOutlineAddSibling (↗ see page 77)
PDFOutlineDelete (↗ see page 77)
PDFOutlineGetAction (↗ see page 78)
PDFOutlineGetDestination (↗ see page 78)
PDFOutlineGetCount (↗ see page 79)
PDFOutlineGetExpanded (↗ see page 79)
PDFOutlineGetFlags (↗ see page 79)
PDFOutlineGetNext (↗ see page 80)
PDFOutlineGetParent (↗ see page 80)
PDFOutlineGetPrev (↗ see page 80)
PDFOutlineGetTitle (↗ see page 81)
PDFOutlineHasChildren (↗ see page 81)
PDFOutlineSetAction (↗ see page 82)
PDFOutlineSetColor (↗ see page 82)
PDFOutlineSetDestination (↗ see page 82)
PDFOutlineSetExpanded (↗ see page 83)
PDFOutlineSetFlags (↗ see page 83)
PDFOutlineSetTitle (↗ see page 84)
PDFOutlineGetColor (↗ see page 84)
PDFOutlineGetFirstChild (↗ see page 84)
PDFOutlineGetLastChild (↗ see page 85)
PDFDocGetOutlineRoot (↗ see page 85)
PDFOutlineUnLink (↗ see page 86)

11.1 PDFOutlineAddChild Function

```
void PDFOutlineAddChild(PDFDocHandle Doc, PDFOutlineHandle Outline, PDFOutlineHandle Child);
```

File

VSOutlineA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current PDF document.
`PDFOutlineHandle Outline`	[in] Current outline.
`PDFOutlineHandle Child`	[in] Child handle.

Returns

None.

Description

Sets outline as child for current outline.

11.2 PDFOutlineAddNewChild Function

```
PDFOutlineHandle PDFOutlineAddNewChild(PDFDocHandle Doc, PDFOutlineHandle Outline);
```

File

VSOutlineA.h

Parameters

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

Returns

Handle to created outline.

Description

Creates outline as child for current outline.

11.3 PDFOutlineAddNewNext Function

```
PDFOutlineHandle PDFOutlineAddNewNext(PDFDocHandle Doc, PDFOutlineHandle Outline);
```

File

VSOutlineA.h

Parameters

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

Returns

Handle to created outline.

Description

Adds new outline which will be next for current outline.

11.4 PDFOutlineAddNewPrev Function

```
PDFOutlineHandle PDFOutlineAddNewPrev(PDFDocHandle Doc, PDFOutlineHandle Outline);
```

File

VSOutlineA.h

Parameters

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

Returns

Handle to created outline.

Description

Adds new outline which will be previous for current outline.

11.5 PDFOutlineAddNewSibling Function

```
PDFOutlineHandle PDFOutlineAddNewSibling(PDFDocHandle Doc, PDFOutlineHandle Outline);
```

File

VSOutlineA.h

Parameters

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

Returns

Handle to created outline.

Description

Creates new outline which will be parallel for current outline.

11.6 PDFOutlineAddNext Function

```
void PDFOutlineAddNext(PDFDocHandle Doc, PDFOutlineHandle Outline, PDFOutlineHandle Next);
```

File

VSOutlineA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current PDF document.
`PDFOutlineHandle Outline`	[in] Current outline.
`PDFOutlineHandle Next`	[in] Handle outline to be added.

Returns

None.

Description

Adds existing outline followed by current outline.

11.7 PDFOutlineAddPrev Function

```
void PDFOutlineAddPrev(PDFDocHandle Doc, PDFOutlineHandle Outline, PDFOutlineHandle Prev);
```

File

VSOutlineA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current PDF document.
`PDFOutlineHandle Outline`	[in] Current outline.
`PDFOutlineHandle Prev`	[in] Handle outline to be added.

Returns

None.

Description

Adds existing outline before the current outline.

11.8 PDFOutlineAddSibling Function

```
void PDFOutlineAddSibling(PDFDocHandle Doc, PDFOutlineHandle Outline, PDFOutlineHandle
Sibling);
```

File

VSOutlineA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current PDF document.
`PDFOutlineHandle Outline`	[in] Current outline.
`PDFOutlineHandle Sibling`	[in] Added outline.

Returns

None.

Description

Adds existing outline which will be parallel for current outline.

11.9 PDFOutlineDelete Function

```
void PDFOutlineDelete(PDFDocHandle Doc, PDFOutlineHandle Outline);
```

File

VSOutlineA.h

Parameters

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

Returns

None.

Description

Deletes current outline.

11.10 PDFOutlineGetAction Function

```
PDFActionHandle PDFOutlineGetAction(PDFDocHandle Doc, PDFOutlineHandle Outline);
```

File

VSOutlineA.h

Parameters

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

Returns

Outline action handle.

Description

Returns action linked to outline.

11.11 PDFOutlineGetDestination Function

```
PDFDestinationHandle PDFOutlineGetDestination(PDFDocHandle Doc, PDFOutlineHandle Outline);
```

File

VSOutlineA.h

Parameters

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

Returns

Linked object handle.

Description

Returns object to linked outline .

11.12 PDFOutlineGetCount Function

```
ppInt32 PDFOutlineGetCount(PDFDocHandle Doc, PDFOutlineHandle Outline);
```

File

VSOutlineA.h

Parameters

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

Returns

Count of children.

Description

Calculates count of the children.

11.13 PDFOutlineGetExpanded Function

```
ppBool PDFOutlineGetExpanded(PDFDocHandle Doc, PDFOutlineHandle Outline);
```

File

VSOutlineA.h

Parameters

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

Returns

true - if outline expanded, false - if no.

Description

Inspects current outline on expanding.

11.14 PDFOutlineGetFlags Function

```
ppInt32 PDFOutlineGetFlags(PDFDocHandle Doc, PDFOutlineHandle Outline);
```

File

VSOutlineA.h

Parameters

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

Returns

Outline flags.

Description

Calculates outline flags.

11.15 PDFOutlineGetNext Function

```
PDFOutlineHandle PDFOutlineGetNext(PDFDocHandle Doc, PDFOutlineHandle Outline);
```

File

VSOutlineA.h

Parameters

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

Returns

Handle to following outline.

Description

Returns following outline.

11.16 PDFOutlineGetParent Function

```
PDFOutlineHandle PDFOutlineGetParent(PDFDocHandle Doc, PDFOutlineHandle Outline);
```

File

VSOutlineA.h

Parameters

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

Returns

Handle to parent outline.

Description

Returns parent outline.

11.17 PDFOutlineGetPrev Function

```
PDFOutlineHandle PDFOutlineGetPrev(PDFDocHandle Doc, PDFOutlineHandle Outline);
```

File

VSOutlineA.h

Parameters

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

Returns

Handle to previous outline.

Description

Returns previous outline.

11.18 PDFOutlineGetTitle Function

```
ppInt32 PDFOutlineGetTitle(PDFDocHandle Doc, PDFOutlineHandle Outline, char * str, ppInt32
len);
```

File

VSOutlineA.h

Parameters

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

Returns

None.

Description

Returns title of outline.

11.19 PDFOutlineHasChildren Function

```
ppBool PDFOutlineHasChildren(PDFDocHandle Doc, PDFOutlineHandle Outline);
```

File

VSOutlineA.h

Parameters

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

Returns

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

Description

Inspects current outline on children presence.

11.20 PDFOutlineSetAction Function

```
void PDFOutlineSetAction(PDFDocHandle Doc, PDFOutlineHandle Outline, PDFActionHandle
Action);
```

File

VSOutlineA.h

Parameters

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

Returns

None.

Description

Links action to outline.

11.21 PDFOutlineSetColor Function

```
void PDFOutlineSetColor(PDFDocHandle Doc, PDFOutlineHandle Outline, PPDFColor Color);
```

File

VSOutlineA.h

Parameters

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

Returns

None.

Description

Sets color of outline.

11.22 PDFOutlineSetDestination Function

```
void PDFOutlineSetDestination(PDFDocHandle Doc, PDFOutlineHandle Outline,
PDFDestinationHandle Destination);
```

File

VSOutlineA.h

Parameters

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

Returns

None.

Description

Links outline to object.

11.23 PDFOutlineSetExpanded Function

```
void PDFOutlineSetExpanded(PDFDocHandle Doc, PDFOutlineHandle Outline, ppBool Expanded);
```

File

VSOutlineA.h

Parameters

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

Returns

None.

Description

Sets current outline on expanding.

11.24 PDFOutlineSetFlags Function

```
void PDFOutlineSetFlags(PDFDocHandle Doc, PDFOutlineHandle Outline, ppInt32 Flags);
```

File

VSOutlineA.h

Parameters

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

Returns

None.

Description

Sets the outline flags.

11.25 PDFOutlineSetTitle Function

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

File

VSOutlineA.h

Parameters

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

Returns

None.

Description

Sets title of outline.

11.26 PDFOutlineGetColor Function

```
ppBool PDFOutlineGetColor(PDFDocHandle Doc, PDFOutlineHandle Outline, PPDFColor Color);
```

File

VSOutlineA.h

Parameters

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

Returns

If color correspond - true, else - no.

Description

Inspects color of outline.

11.27 PDFOutlineGetFirstChild Function

```
PDFOutlineHandle PDFOutlineGetFirstChild(PDFDocHandle Doc, PDFOutlineHandle Outline);
```

File

VSOutlineA.h

Parameters

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

Returns

Handle to the first outline child.

Description

Returns the first outline child.

11.28 PDFOutlineGetLastChild Function

```
PDFOutlineHandle PDFOutlineGetLastChild(PDFDocHandle Doc, PDFOutlineHandle Outline);
```

File

VSOutlineA.h

Parameters

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

Returns

Handle to the last outline child.

Description

Returns the last outline child.

11.29 PDFDocGetOutlineRoot Function

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

File

VSOutlineA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current PDF document.

Returns

Handle to document outlines root.

Description

Returns root of outlines.

11.30 PDFOutlineUnLink Function

```
void PDFOutlineUnLink(PDFDocHandle Doc, PDFOutlineHandle Outline);
```

File

VSOutlineA.h

Parameters

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

Returns

None.

Description

Deletes outline links.

12 Thread and Bead Level

12.1 Thread Operations

Functions

Function
PDFThreadDelete (⧉ see page 87)
PDFThreadGetFirstBead (⧉ see page 87)
PDFThreadGetInfo (⧉ see page 88)
PDFThreadNew (⧉ see page 88)
PDFThreadSetFirstBead (⧉ see page 89)
PDFThreadSetInfo (⧉ see page 89)
PDFDocGetThread (⧉ see page 90)
PDFDocGetThreadCount (⧉ see page 90)

12.1.1 PDFThreadDelete Function

```
void PDFThreadDelete(PDFDocHandle Doc, PDFThreadHandle Thread);
```

File

VSThreadA.h

Parameters

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

Returns

None.

Description

Deletes Thread from PDF Document.

See Also

PDFThreadNew (⧉ see page 88)

12.1.2 PDFThreadGetFirstBead Function

```
PDFBeadHandle PDFThreadGetFirstBead(PDFDocHandle Doc, PDFThreadHandle Thread);
```

File

VSThreadA.h

Parameters

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

Returns

Bead Handle on the First Bead in Thread.

Description

Gets Bead Handle on First Bead in Thread.

See Also

PDFThreadSetFirstBead (see page 89)

12.1.3 PDFThreadGetInfo Function

```
ppInt32 PDFThreadGetInfo(PDFDocHandle Doc, PDFThreadHandle Thread, char * InfoKey, char *
Value, ppInt32 MaxLength);
```

File

VSThreadA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`PDFThreadHandle Thread`	[in] Thread Handle in PDF Document.
`char * InfoKey`	[in] Name of requesting information.
`char * Value`	[out] Text value of requesting information.
`ppInt32 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 89)

12.1.4 PDFThreadNew Function

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

File

VSThreadA.h

Parameters

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

Returns

New Thread Handle in PDF Document.

Description

Creates New Thread in PDF Document.

See Also

PDFThreadDelete (↗ see page 87)

12.1.5 PDFThreadSetFirstBead Function

```
void PDFThreadSetFirstBead(PDFDocHandle Doc, PDFThreadHandle Thread, PDFBeadHandle Bead);
```

File

VSThreadA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`PDFThreadHandle Thread`	[in] Thread Handle in PDF Document.
`PDFBeadHandle Bead`	[in] Bead Handle of New Thread in PDF Document.

Returns

None.

Description

Initializes Bead's Thread by creation the First Bead.

See Also

PDFThreadGetFirstBead (↗ see page 87)

12.1.6 PDFThreadSetInfo Function

```
void PDFThreadSetInfo(PDFDocHandle Doc, PDFThreadHandle Thread, char * InfoKey, char *
Value, ppInt32 Length);
```

File

VSThreadA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`PDFThreadHandle Thread`	[in] Thread Handle in PDF Document.
`char * InfoKey`	[in] Name of setting information.
`char * Value`	[in] Text string information value.
`ppInt32 Length`	[in] Length of value in bytes.

Returns

None.

Description

Sets information to Thread according to property name.

See Also

PDFThreadGetInfo (↗ see page 88)

12.1.7 PDFDocGetThread Function

```
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 (⧉ see page 184)

12.1.8 PDFDocGetThreadCount Function

```
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.

12.2 Bead Operations

Functions

Function
PDFBeadDelete (⧉ see page 91)
PDFBeadGetIndex (⧉ see page 91)
PDFBeadGetNext (⧉ see page 91)
PDFBeadGetPage (⧉ see page 92)
PDFBeadGetPrev (⧉ see page 92)
PDFBeadGetRect (⧉ see page 93)
PDFBeadGetThread (⧉ see page 93)

PDFBeadInsert (see page 93)
PDFBeadNew (see page 94)
PDFBeadSetRect (see page 94)

12.2.1 PDFBeadDelete Function

```
void PDFBeadDelete(PDFDocHandle Doc, PDFBeadHandle Bead);
```

File

VSThreadA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`PDFBeadHandle Bead`	[in] Handle of deleting bead.

Returns

None.

Description

Deletes bead.

See Also

PDFBeadNew (see page 94)

12.2.2 PDFBeadGetIndex Function

```
ppInt32 PDFBeadGetIndex(PDFDocHandle Doc, PDFBeadHandle Bead);
```

File

VSThreadA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`PDFBeadHandle Bead`	[in] Bead for which index we are looking for.

Returns

Index of Bead in Owner's Thread.

Description

Gets Index of Bead in Owner's Thread.

12.2.3 PDFBeadGetNext Function

```
PDFBeadHandle PDFBeadGetNext(PDFDocHandle Doc, PDFBeadHandle Bead);
```

File

VSThreadA.h

Parameters

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

Returns

Handle of next bead.

Description

Navigates to the next bead item.

See Also

PDFBeadGetPrev (◪ see page 92)

12.2.4 PDFBeadGetPage Function

```
ppInt32 PDFBeadGetPage(PDFDocHandle Doc, PDFBeadHandle Bead);
```

File

VSThreadA.h

Parameters

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

Returns

Index of Page on which this bead appears.

Description

Gets an Index of Page on which this bead appears.

12.2.5 PDFBeadGetPrev Function

```
PDFBeadHandle PDFBeadGetPrev(PDFDocHandle Doc, PDFBeadHandle Bead);
```

File

VSThreadA.h

Parameters

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

Returns

Handle of previous bead.

Description

Navigates to previous bead item.

See Also

PDFBeadGetNext (◪ see page 91)

12.2.6 PDFBeadGetRect Function

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

File

VSThreadA.h

Parameters

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

Returns

None.

Description

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

See Also

PDFBeadSetRect (see page 94)

12.2.7 PDFBeadGetThread Function

```
PDFThreadHandle PDFBeadGetThread(PDFDocHandle Doc, PDFBeadHandle Bead);
```

File

VSThreadA.h

Parameters

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

Returns

Handle of Bead's Thread.

Description

Gets a Handle of Bead's Thread.

12.2.8 PDFBeadInsert Function

```
void PDFBeadInsert(PDFDocHandle Doc, PDFBeadHandle Bead, PDFBeadHandle NewBead);
```

File

VSThreadA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`PDFBeadHandle Bead`	[in] Current Bead Handle of Thread in PDF Document.

`PDFBeadHandle NewBead`	[in] New Bead Handle for inserting in Thread.

Returns

None.

Description

Inserts New Bead after current Bead.

12.2.9 PDFBeadNew Function

```
PDFBeadHandle PDFBeadNew(PDFDocHandle Doc, ppInt32 Page, TPDFRect Rect);
```

File

VSThreadA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document Handle.
`ppInt32 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.

See Also

PDFBeadDelete (↗ see page 91)

12.2.10 PDFBeadSetRect Function

```
void PDFBeadSetRect(PDFDocHandle Doc, PDFBeadHandle Bead, TPDFRect Rect);
```

File

VSThreadA.h

Parameters

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

Returns

None.

Description

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

See Also

PDFBeadGetRect (↗ see page 93)

13 Annotation Level

Àííîòàöèè - 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
PDFPageAddFileAttachAnnotation (↗ see page 95)
PDFPageAddFreeAnnotation (↗ see page 95)
PDFPageAddLineAnnotation (↗ see page 96)
PDFPageAddLinkAnnotation (↗ see page 96)
PDFPageAddMovieAnnotation (↗ see page 97)
PDFPageAddPolyAnnotation (↗ see page 97)
PDFPageAddPopupAnnotation (↗ see page 97)
PDFPageAddRubberStampAnnotation (↗ see page 98)
PDFPageAddSCAnnotation (↗ see page 98)
PDFPageAddSoundAnnotationFromFile (↗ see page 99)
PDFPageAddTextAnnotation (↗ see page 99)
PDFPageAddCaretAnnotation (↗ see page 100)

13.1 PDFPageAddFileAttachAnnotation Function

```
PDFAnnotationHandle PDFPageAddFileAttachAnnotation(PDFDocHandle Doc, ppInt32 Page,
TFileAttachAnnotDict Annot);
```

File

VSAnnotA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current document.
`ppInt32 Page`	[in] Number of page.
`AnnotDescription`	[in] Pointer to attach file annotation.

Returns

Annotation handle.

Description

Adds attached file annotation to document.

13.2 PDFPageAddFreeAnnotation Function

```
PDFAnnotationHandle PDFPageAddFreeAnnotation(PDFDocHandle Doc, ppInt32 Page, TFreeAnnotDict
Annot);
```

File

VSAnnotA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current document.
`ppInt32 Page`	[in] Number of page.
`AnnotDescription`	[in] Pointer to free annotation.

Returns

Annotation handle.

Description

Adds free annotation to document.

13.3 PDFPageAddLineAnnotation Function

```
PDFAnnotationHandle PDFPageAddLineAnnotation(PDFDocHandle Doc, ppInt32 Page, TLineAnnotDict
Annot);
```

File

VSAnnotA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current document.
`ppInt32 Page`	[in] Number of page.
`AnnotDescription`	[in] Pointer to line annotation.

Returns

Annotation handle.

Description

Adds line annotation to document.

13.4 PDFPageAddLinkAnnotation Function

```
PDFAnnotationHandle PDFPageAddLinkAnnotation(PDFDocHandle Doc, ppInt32 Page, TLinkAnnotDict
Annot);
```

File

VSAnnotA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current document.
`ppInt32 Page`	[in] Number of page.
`AnnotDescription`	[in] Pointer to text annotation.

Returns

Annotation handle.

Description

Adds Link annotation to document.

13.5 PDFPageAddMovieAnnotation Function

```
PDFAnnotationHandle PDFPageAddMovieAnnotation(PDFDocHandle Doc, ppInt32 Page,
TMovieAnnotDict Annot);
```

File

VSAnnotA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current document.
`ppInt32 Page`	[in] Number of page.
`AnnotDescription`	[in] Pointer to movie annotation.

Returns

Annotation handle.

Description

Adds movie annotation to document.

13.6 PDFPageAddPolyAnnotation Function

```
PDFAnnotationHandle PDFPageAddPolyAnnotation(PDFDocHandle Doc, ppInt32 Page, TPolyAnnotDict
Annot);
```

File

VSAnnotA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current document.
`ppInt32 Page`	[in] Number of page.
`AnnotDescription`	[in] Pointer to polygon annotation.

Returns

Annotation handle.

Description

Adds polygon annotation to document.

13.7 PDFPageAddPopupAnnotation Function

```
PDFAnnotationHandle PDFPageAddPopupAnnotation(PDFDocHandle Doc, ppInt32 Page,
TPopupAnnotDict Annot);
```

File

VSAnnotA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current document.
`ppInt32 Page`	[in] Number of page.
`AnnotDescription`	[in] Pointer to text annotation.

Returns

Annotation handle.

Description

Adds popup annotation to document.

13.8 PDFPageAddRubberStampAnnotation Function

```
PDFAnnotationHandle PDFPageAddRubberStampAnnotation(PDFDocHandle Doc, ppInt32 Page,
TRubberStampAnnotDict Annot);
```

File

VSAnnotA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current document.
`ppInt32 Page`	[in] Number of page.
`AnnotDescription`	[in] Pointer to rubber stamp annotation.

Returns

Annotation handle.

Description

Adds rubber stamp annotation to document.

13.9 PDFPageAddSCAnnotation Function

```
PDFAnnotationHandle PDFPageAddSCAnnotation(PDFDocHandle Doc, ppInt32 Page, TSCAnnotDict
Annot, TSCType Type);
```

File

VSAnnotA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current document.
`ppInt32 Page`	[in] Number of page.
`AnnotDescription`	[in] Pointer to square and circle annotation.

Returns

Annotation handle.

Description

Adds square and circle annotation to document.

13.10 PDFPageAddSoundAnnotationFromFile Function

```
PDFAnnotationHandle PDFPageAddSoundAnnotationFromFile(PDFDocHandle Doc, ppInt32 Page,
TSoundAnnotDict Annot, int SamplingRate, int Channels, int BitsPerSample, TSEFormat
EncFormat);
```

File

VSAnnotA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current document.
`ppInt32 Page`	[in] Number of page.
`int SamplingRate`	[in] The sampling rate, in samples per second.
`int Channels`	[in] The number of sound channels.
`int BitsPerSample`	[in] The number of bits per sample value per channel.
`TSEFormat EncFormat`	[in] Sound encoding format.
`AnnotDescription`	[in] Pointer to text annotation.
`FileName`	[in] Sound filename.

Returns

Annotation handle.

Description

Adds sound annotation to document from file.

13.11 PDFPageAddTextAnnotation Function

```
PDFAnnotationHandle PDFPageAddTextAnnotation(PDFDocHandle Doc, ppInt32 Page, TTextAnnotDict
Annot);
```

File

VSAnnotA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current document.
`ppInt32 Page`	[in] Number of page.
`AnnotDescription`	[in] Pointer to text annotation.

Returns

Annotation handle.

Description

Adds text annotation to document.

13.12 PDFPageAddCaretAnnotation Function

```
PDFAnnotationHandle PDFPageAddCaretAnnotation(PDFDocHandle Doc, ppInt32 Page,
TCaretAnnotDict Annot);
```

File

VSAnnotA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] Current document.
`ppInt32 Page`	[in] Number of page.
`AnnotDescription`	[in] Pointer to caret annotation.

Returns

Annotation handle.

Description

Adds caret annotation to document.

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.

14.1 Common Action Level

Functions

Function
PDFDestinationGetInfo (↗ see page 101)
PDFDestinationNameNew (↗ see page 101)
PDFActionSetNext (↗ see page 102)
PDFActionGetNextItem (↗ see page 102)
PDFActionGetNextItemCount (↗ see page 103)
PDFActionGetType (↗ see page 103)

14.1.1 PDFDestinationGetInfo Function

```
ppBool PDFDestinationGetInfo(PDFDocHandle Doc, PDFDestinationHandle DestH, PDFExplicitDest
* Destination);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`PDFDestinationHandle DestH`	[in] Destination handle which is needed to be converted to structure.
`PDFExplicitDest * Destination`	[out] Pointer to PDFExplicitDest (↗ see page 181) structure

Returns

If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

Description

Converts PDFDestinationHandle (↗ see page 180) (received from PDFActionGetGoToDestination (↗ see page 104) and PDFActionGetGoToRemoteDestination (↗ see page 106) functions) to PDFExplicitDest (↗ see page 181) structure.

14.1.2 PDFDestinationNameNew Function

```
void PDFDestinationNameNew(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

14.1.3 PDFActionSetNext Function

```
void PDFActionSetNext(PDFDocHandle Doc, PDFActionHandle Action, PDFActionHandle Next);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`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.

14.1.4 PDFActionGetNextItem Function

```
PDFActionHandle PDFActionGetNextItem(PDFDocHandle Doc, PDFActionHandle Action, ppInt32
Index);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`PDFActionHandle Action`	[in] Handle of the action object.
`ppInt32 Index`	[in] Index of the action which needs to be received.

Returns

The return value is a handle to the specified object.

Description

This function retrieves the actions which will be executed after this action.

14.1.5 PDFActionGetNextItemCount Function

```
ppInt32 PDFActionGetNextItemCount(PDFDocHandle Doc, PDFActionHandle Action);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`PDFActionHandle Action`	[in] Handle of the action object.

Returns

The number of actions.

Description

This function retrieves the number of actions which will be executed after this action.

14.1.6 PDFActionGetType Function

```
PDFActionType PDFActionGetType(PDFDocHandle Doc, PDFActionHandle Action);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`PDFActionHandle Action`	[in] Handle of the action object.

Returns

If the function succeeds, the return value will identify the object.

Description

This function returns the type of the specified object.

14.2 Goto Action

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

Functions

Function
PDFActionGetGoToDestination (⧉ see page 104)
PDFActionNewGoToName (⧉ see page 104)
PDFActionNewGoToDestination (⧉ see page 104)

14.2.1 PDFActionGetGoToDestination Function

```
PDFDestinationHandle PDFActionGetGoToDestination(PDFDocHandle Doc, PDFActionHandle Action);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`PDFActionHandle Action`	[in] Handle of the PDF action.

Returns

Destination handle which can be processed with PDFDestinationGetInfo (↗ see page 101) function.

Description

Returns destination handle for this action.

14.2.2 PDFActionNewGoToName Function

```
PDFActionHandle PDFActionNewGoToName(PDFDocHandle Doc, char * String, ppInt32 Length);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`char * String`	[in] Name destination.
`ppInt32 Length`	[in] Length of the name destination.

Returns

The return value is a handle to a PDF action.

Description

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

14.2.3 PDFActionNewGoToDestination Function

```
PDFActionHandle PDFActionNewGoToDestination(PDFDocHandle Doc, PDFExplicitDest Destination);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`PDFExplicitDest Destination`	[in] Explicit destination.

Returns

The return value is a handle to a PDF action.

Description

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

14.3 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
PDFActionNewGoToRemoteDestination (↗ see page 105)
PDFActionNewGoToRemoteName (↗ see page 105)
PDFActionGetGoToRemoteDestination (↗ see page 106)
PDFActionGetGoToRemoteInNewWindow (↗ see page 106)

14.3.1 PDFActionNewGoToRemoteDestination Function

```
PDFActionHandle PDFActionNewGoToRemoteDestination(PDFDocHandle Doc, char * FileName,
ppInt32 FileNameLength, PDFExplicitDest Dest, 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.
`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.
`Destination`	[in] Explicit destination.

Returns

The return value is a handle to a PDF action.

Description

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

14.3.2 PDFActionNewGoToRemoteName Function

```
PDFActionHandle PDFActionNewGoToRemoteName(PDFDocHandle Doc, char * FileName, ppInt32
FileNameLength, char * String, ppInt32 Length, 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.
`char * String`	[in] Name destination.
`ppInt32 Length`	[in] Length of the name destination.
`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.

14.3.3 PDFActionGetGoToRemoteDestination Function

```
PDFDestinationHandle PDFActionGetGoToRemoteDestination(PDFDocHandle Doc, PDFActionHandle
Action, PDFString * FileName);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`PDFActionHandle Action`	[in] Handle of the PDF action.
`PDFString * FileName`	[out] Name of the PDF document which will be opened after execution of this action.

Returns

Destination handle which can be processed with PDFDestinationGetInfo (see page 101) function.

Description

Returns destination handle for this action.

14.3.4 PDFActionGetGoToRemoteInNewWindow Function

```
ppBool PDFActionGetGoToRemoteInNewWindow(PDFDocHandle Doc, PDFActionHandle Action);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`PDFActionHandle Action`	[in] Handle of the PDF action.

Returns

If the new document is opened in new window, the return value will be nonzero. If the new document is opened in current window, the return value will be zero.

Description

Returns stored JavaScript in CosStream or in string.

14.4 Launch Action

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

Functions

Function
PDFActionNewLaunch (⧉ see page 107)
PDFActionGetLaunch (⧉ see page 107)

14.4.1 PDFActionNewLaunch Function

```
PDFActionHandle PDFActionNewLaunch(PDFDocHandle Doc, PDFLaunch Launch, ppBool InNewWindow);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`PDFLaunch Launch`	[in] Point to a **PDFLaunch (⧉ see page 172)** structure that defines the characteristics of the launch action.
`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 "Launch" action.

14.4.2 PDFActionGetLaunch Function

```
void PDFActionGetLaunch(PDFDocHandle Doc, PDFActionHandle Action, PDFLaunchP Launch, ppBool
* InNewWindow, ppBool * IsWinFormat);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`PDFActionHandle Action`	[in] Handle of the PDF action.
`PDFLaunchP Launch`	[out] Point to a **PDFLaunch (⧉ see page 172)** structure where the characteristics of the launch action will be stored.
`ppBool * IsWinFormat`	[out] If information is stored for windows platform the value will be true.
`IsNewWindow`	[out] If the launched application is opened in new window, the value will be true.

Returns

None.

Description

Returns information that defines the characteristics of the launch action.

Notes

Adobe Acrobat (r) supports only windows platform information.

14.5 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 108)
PDFActionGetHideItem (↗ see page 108)
PDFActionGetHideCount (↗ see page 109)
PDFActionGetHideIsHide (↗ see page 109)
PDFActionHideAddAnnotation (↗ see page 110)
PDFActionHideAddAnnotationName (↗ see page 110)

14.5.1 PDFActionNewHide Function

```
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.

14.5.2 PDFActionGetHideItem Function

```
void PDFActionGetHideItem(PDFDocHandle Doc, PDFActionHandle Action, ppInt32 Index,
PDFAnnatationIdentifyP Annotation);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle.
`PDFActionHandle Action`	[in] Handle of the PDF action.
`ppInt32 Index`	[in] Index of the action in the list, referenced by a 0-based index.
`PDFAnnatationIdentifyP Annotation`	[in] Information about annotation

Returns

None.

Description

Returns information about annotation which will be used by this action.

14.5.3 PDFActionGetHideCount Function

```
ppInt32 PDFActionGetHideCount(PDFDocHandle Doc, PDFActionHandle Action);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle.
`PDFActionHandle Action`	[in] Handle of the PDF action.

Returns

Count of the used annotations.

Description

Returns count of the annotation which will be used by this action.

Notes

If result is zero all annotations will be used.

14.5.4 PDFActionGetHideIsHide Function

```
ppBool PDFActionGetHideIsHide(PDFDocHandle Doc, PDFActionHandle Action);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle.
`PDFActionHandle Action`	[in] Handle of the PDF action.

Returns

Result indicating whether to hide the annotation (true) or show it (false).

Description

Returns operation which will be executed by this action.

14.5.5 PDFActionHideAddAnnotation Function

```
void PDFActionHideAddAnnotation(PDFDocHandle Doc, PDFActionHandle Action,
PDFAnnotationHandle Annotation);
```

File

VSActionA.h

Parameters

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

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.

14.5.6 PDFActionHideAddAnnotationName Function

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

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle.
`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.

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.

14.6 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)
PDFActionGetURI (↗ see page 111)

14.6.1 PDFActionNewURI Function

```
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.

14.6.2 PDFActionGetURI Function

```
void PDFActionGetURI(PDFDocHandle Doc, PDFActionHandle Action, PDFString * URI, ppBool *
IsMap);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`PDFActionHandle Action`	[in] Handle of the PDF action.
`PDFString * URI`	[out] The uniform resource identifier to resolve.
`ppBool * IsMap`	[out] If tracking the mouse position, the value will be true.

Returns

None.

Description

Returns information that defines the characteristics of the "URI" action.

14.7 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)
PDFActionGetThread (↗ see page 112)

14.7.1 PDFActionNewThread Function

```
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 **PDFThreadActionParam (↗ see page 173)** 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 (↗ see page 179) or PDFThreadHandle (↗ see page 184) impossible to use in **PDFThreadActionParam (↗ see page 173)** structure.

14.7.2 PDFActionGetThread Function

```
void PDFActionGetThread(PDFDocHandle Doc, PDFActionHandle Action, PDFString * FileName,
PDFThreadActionParamP Thread);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`PDFActionHandle Action`	[in] Handle of the PDF action.
`PDFString * FileName`	[out] Name of the PDF document where destination thread must be desired.
`PDFThreadActionParamP Thread`	[out] Point to a **PDFThreadActionParam (↗ see page 173)** structure where the characteristics of the thread action will be stored.

Returns

None.

Description

Returns information that defines the characteristics of the thread action.

14.8 Named Action

Viewer applications supports several named actions.

Functions

Function
PDFActionNewNamed (⧉ see page 113)
PDFActionGetNamed (⧉ see page 113)

14.8.1 PDFActionNewNamed Function

```
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.

14.8.2 PDFActionGetNamed Function

```
void PDFActionGetNamed(PDFDocHandle Doc, PDFActionHandle Action, PDFNamedActionType *
NamedType);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle.
`PDFActionHandle Action`	[in] Handle of the PDF action.
`PDFNamedActionType * NamedType`	[out] Operation for the named action.

Returns

None.

Description

Returns operation which will be executed by this action.

14.9 Sound Action

A sound action plays a sound through the computer’s speakers.

Functions

Function
PDFActionNewSound (⧉ see page 114)
PDFActionGetSound (⧉ see page 114)

14.9.1 PDFActionNewSound Function

```
PDFActionHandle PDFActionNewSound(PDFDocHandle Doc, PDFSoundHandle Sound, ppReal Volume,
ppBool Synch, ppBool Repeating, ppBool Mix);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle.
`PDFSoundHandle Sound`	[in] PDF Annotation with sound information
`ppReal Volume`	[in] The volume at which to play the sound, in the range from -1.0 to 1.0. Higher values denote greater volume; negative values mute the sound.
`ppBool Synch`	[in] A flag specifying whether to play the sound synchronously or asynchronously.
`ppBool Repeating`	[in] A flag specifying whether to repeat the sound indefinitely.
`ppBool Mix`	[in] A flag specifying whether to mix this sound with any other sound already playing.

Returns

The return value is a handle to a PDF action.

Description

Creates new "Sound" action.

14.9.2 PDFActionGetSound Function

```
PDFSoundHandle PDFActionGetSound(PDFDocHandle Doc, PDFActionHandle Action, ppReal * Volume,
ppBool * Synch, ppBool * Repeating, ppBool * Mix);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`PDFActionHandle Action`	[in] Handle of the PDF action.

`ppReal * Volume`	[out] The volume at which to play the sound, in the range from -1.0 to 1.0. Higher values denote greater volume; negative values mute the sound.
`ppBool * Synch`	[out] A flag specifying whether to play the sound synchronously or asynchronously.
`ppBool * Repeating`	[out] A flag specifying whether to repeat the sound indefinitely.
`ppBool * Mix`	[our] A flag specifying whether to mix this sound with any other sound already playing.

Returns

The return value is sound annotation handle.

Description

Returns information that defines the characteristics of the sound action.

14.10 Movie Action

A movie action can be used to play a movie in a floating window or within the annotation rectangle of a movie annotation.

Functions

Function
PDFActionNewMovie (↗ see page 115)
PDFActionNewMovieName (↗ see page 115)
PDFActionGetMovie (↗ see page 116)

14.10.1 PDFActionNewMovie Function

```
PDFActionHandle PDFActionNewMovie(PDFDocHandle Doc, PDFAnnotationHandle Movie,
PDFMovieActionOperation Operation);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle.
`PDFAnnotationHandle Movie`	[in] PDF Annotation with movie information
`PDFMovieActionOperation Operation`	[in] The operation to be performed on the movie.

Returns

The return value is a handle to a PDF action.

Description

Creates new "Movie" action.

14.10.2 PDFActionNewMovieName Function

```
PDFActionHandle PDFActionNewMovieName(PDFDocHandle Doc, char * String, ppInt32 Length,
PDFMovieActionOperation Operation);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle.
`char * String`	[in] PDF Annotation name with movie information
`ppInt32 Length`	[in] Annotation name length.
`PDFMovieActionOperation Operation`	[in] The operation to be performed on the movie.

Returns

The return value is a handle to a PDF action.

Description

Creates new "Movie" action.

14.10.3 PDFActionGetMovie Function

```
void PDFActionGetMovie(PDFDocHandle Doc, PDFActionHandle Action, PDFAnnatationIdentifyP
Param, PDFMovieActionOperation * Operation);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`PDFActionHandle Action`	[in] Handle of the PDF action.
`PDFAnnatationIdentifyP Param`	[out] Pointer to description of the movie annotation
`PDFMovieActionOperation * Operation`	[out] Returns type of the performed operation on the movie.

Returns

None.

Description

Returns information that defines the characteristics of the movie action.

14.11 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 116)
PDFActionGetImportData (see page 117)

14.11.1 PDFActionNewImportData Function

```
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.

14.11.2 PDFActionGetImportData Function

```
void PDFActionGetImportData(PDFDocHandle Doc, PDFActionHandle Action, PDFString * FileName);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle.
`PDFActionHandle Action`	[in] Handle of the PDF action.
`PDFString * FileName`	[out] The FDF filename from which data will be imported.

Returns

None.

Description

Returns information about filename from which will be imported the data.

Notes

14.12 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
PDFActionSubmitFormAddAnnotation (see page 118)
PDFActionSubmitFormAddAnnotationName (see page 118)
PDFActionNewSubmitForm (see page 119)
PDFActionGetSubmitForm (see page 120)
PDFActionGetSubmitFormCount (see page 120)
PDFActionGetSubmitFormItem (see page 120)

14.12.1 PDFActionSubmitFormAddAnnotation Function

```
void PDFActionSubmitFormAddAnnotation(PDFDocHandle Doc, PDFActionHandle Action,
PDFAnnotationHandle Annotation);
```

File

VSActionA.h

Parameters

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

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.

14.12.2 PDFActionSubmitFormAddAnnotationName Function

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

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle.
`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.

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.

14.12.3 PDFActionNewSubmitForm Function

```
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 locater 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.

14.12.4 PDFActionGetSubmitForm Function

```
void PDFActionGetSubmitForm(PDFDocHandle Doc, PDFActionHandle Action, PDFString * URI,
ppInt32 * Flags);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle.
`PDFActionHandle Action`	[in] Handle of the PDF action.
`PDFString * URI`	[out] A URL file specification giving the uniform resource locater of the script at the Web server that will process the submission
`Flag`	[out] A set of flags specifying various characteristics of the action.

Returns

None.

Description

Returns information about submitform action.

14.12.5 PDFActionGetSubmitFormCount Function

```
ppInt32 PDFActionGetSubmitFormCount(PDFDocHandle Doc, PDFActionHandle Action);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle.
`PDFActionHandle Action`	[in] Handle of the PDF action.

Returns

Count of the used annotations.

Description

Returns count of the annotation which will be used by this action.

Notes

If result is zero, operation will be performed for all acroform object in the PDF document, flag PDF_SUBMIT_FORM_FLAG_EXCLUDE is not used.

14.12.6 PDFActionGetSubmitFormItem Function

```
void PDFActionGetSubmitFormItem(PDFDocHandle Doc, PDFActionHandle Action, ppInt32 Index,
PDFAnnatationIdentifyP Annotation);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle.
`PDFActionHandle Action`	[in] Handle of the PDF action.
`ppInt32 Index`	[in] Index of the action in the list, referenced by a 0-based index.
`PDFAnnatationIdentifyP Annotation`	[in] Information about annotation

Returns

None.

Description

Returns information about annotation which will be used by this action.

14.13 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
PDFActionResetFormAddAnnotation (see page 121)
PDFActionResetFormAddAnnotationName (see page 122)
PDFActionNewResetForm (see page 122)
PDFActionGetResetForm (see page 122)
PDFActionGetResetFormCount (see page 123)
PDFActionGetResetFormItem (see page 123)

14.13.1 PDFActionResetFormAddAnnotation Function

```
void PDFActionResetFormAddAnnotation(PDFDocHandle Doc, PDFActionHandle Action,
PDFAnnotationHandle Annotation);
```

File

VSActionA.h

Parameters

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

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.

14.13.2 PDFActionResetFormAddAnnotationName Function

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

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle.
`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.

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.

14.13.3 PDFActionNewResetForm Function

```
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.

14.13.4 PDFActionGetResetForm Function

```
void PDFActionGetResetForm(PDFDocHandle Doc, PDFActionHandle Action, ppBool * Exclude);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle.
`PDFActionHandle Action`	[in] Handle of the PDF action.
`ppBool * Exclude`	[out] If false, the list specifies which fields to be 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

None.

Description

Returns information which operation will be performed with presenting acroform object in the list.

Notes

If list is empty, all acroform objects will be reset.

14.13.5 PDFActionGetResetFormCount Function

```
ppInt32 PDFActionGetResetFormCount(PDFDocHandle Doc, PDFActionHandle Action);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle.
`PDFActionHandle Action`	[in] Handle of the PDF action.

Returns

Count of the used annotations.

Description

Returns count of the annotation which will be used by this action.

Notes

If result is zero all acroform objects will be reset.

14.13.6 PDFActionGetResetFormItem Function

```
void PDFActionGetResetFormItem(PDFDocHandle Doc, PDFAnnotationHandle Action, ppInt32 Index,
PDFAnnatationIdentifyP Annotation);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle.
`PDFAnnotationHandle Action`	[in] Handle of the PDF action.
`ppInt32 Index`	[in] Index of the action in the list, referenced by a 0-based index.
`PDFAnnatationIdentifyP Annotation`	[in] Information about annotation

Returns

None.

Description

Returns information about annotation which will be used by this action.

14.14 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
PDFActionGetJavaScriptHandle (↗ see page 124)
PDFActionGetJavaScriptIsHandle (↗ see page 124)
PDFActionGetJavaScriptString (↗ see page 125)
PDFActionNewJavaScriptStream (↗ see page 125)
PDFActionNewJavaScript (↗ see page 125)

14.14.1 PDFActionGetJavaScriptHandle Function

```
PDFCosHandle PDFActionGetJavaScriptHandle(PDFDocHandle Doc, PDFActionHandle Action);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`PDFActionHandle Action`	[in] Handle of the PDF action.

Returns

CosStream handle with javascript

Description

Returns CosStream where javascript is stored for this action.

14.14.2 PDFActionGetJavaScriptIsHandle Function

```
ppBool PDFActionGetJavaScriptIsHandle(PDFDocHandle Doc, PDFActionHandle Action);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`PDFActionHandle Action`	[in] Handle of the PDF action.

Returns

If the javascript is stored in CosStream, the return value will be nonzero. If the javascript is stored in string, the return value will be zero.

Description

Returns JavaScript storage either in CosStream or in string.

14.14.3 PDFActionGetJavaScriptString Function

```
char * PDFActionGetJavaScriptString(PDFDocHandle Doc, PDFActionHandle Action, ppInt32 *
Length);
```

File

VSActionA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] PDF Document handle
`PDFActionHandle Action`	[in] Handle of the PDF action.
`ppInt32 * Length`	[out] Pointer to integer where will be stored size of the javascript string.

Returns

Pointer to string with javascript.

Description

Returns string where stored javascript for this action.

14.14.4 PDFActionNewJavaScriptStream Function

```
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.

14.14.5 PDFActionNewJavaScript Function

```
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 127)
CosCopyObj (⧉ see page 127)
CosGetNumberValue (⧉ see page 128)
CosGetFromDoc (⧉ see page 128)
CosFreeObj (⧉ see page 128)
CosObjGetGeneration (⧉ see page 129)
CosObjGetID (⧉ see page 129)
CosObjIsIndirect (⧉ see page 130)

15.1.1 CosGetType Function

```
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 CosCopyObj Function

```
PDFCosHandle CosCopyObj(PDFDocHandle Doc, PDFCosHandle CosObject);
```

File

VSCosA.h

Parameters

Parameters	Description
`PDFCosHandle CosObject`	[in] The Cos object from which information will be received for copy.

Returns

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

Description

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

15.1.3 CosGetNumberValue Function

```
ppReal CosGetNumberValue(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.

15.1.4 CosGetFromDoc Function

```
PDFCosHandle CosGetFromDoc(PDFDocHandle Doc, ppInt32 ID);
```

File

VSCosA.h

Parameters

Parameters	Description
`PDFDocHandle Doc`	[in] The document from which Cos object will be loaded.
`ppInt32 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 CosFreeObj Function

```
void CosFreeObj(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

15.1.6 CosObjGetGeneration Function

```
ppUns16 CosObjGetGeneration(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 CosObjIsIndirect (⧉ see page 130) function.

Returns

The generation number of CosObj.

Description

Gets the generation number of an indirect Cos object.

See Also

CosObjIsIndirect (⧉ see page 130) CosObjGetID (⧉ see page 129)

15.1.7 CosObjGetID Function

```
ppInt32 CosObjGetID(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 CosObjIsIndirect (⧉ see page 130) function.

Returns

The ID of CosObj.

Description

Gets the index for an indirect object.

See Also

CosObjIsIndirect (⧉ see page 130) CosObjGetGeneration (⧉ see page 129)

15.1.8 CosObjIsIndirect Function

```
ppBool CosObjIsIndirect(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

CosObjGetID (⧉ see page 129) CosObjGetGeneration (⧉ see page 129)

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
CosNewNull (⧉ see page 130)

15.2.1 CosNewNull Function

```
PDFCosHandle CosNewNull(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

CosFreeObj (⧉ see page 128) CosGetType (⧉ see page 127)

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
CosNewBool (⧉ see page 131)
CosGetBoolValue (⧉ see page 131)
CosSetBoolValue (⧉ see page 132)

15.3.1 CosNewBool Function

```
PDFCosHandle CosNewBool(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

CosFreeObj (⧉ see page 128) CosGetType (⧉ see page 127) CosGetBoolValue (⧉ see page 131) CosSetBoolValue (⧉ see page 132)

15.3.2 CosGetBoolValue Function

```
ppBool CosGetBoolValue(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

CosNewBool (⧉ see page 131) CosSetBoolValue (⧉ see page 132)

15.3.3 CosSetBoolValue Function

```
void CosSetBoolValue(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

CosNewBool (⧉ see page 131) CosGetBoolValue (⧉ see page 131)

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.

15.4.1 Cos Real Object

Functions

Function
CosNewReal (⧉ see page 132)
CosGetRealValue (⧉ see page 133)
CosSetRealValue (⧉ see page 133)

15.4.1.1 CosNewReal Function

```
PDFCosHandle CosNewReal(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

CosFreeObj (↗ see page 128) CosGetType (↗ see page 127) CosGetRealValue (↗ see page 133) CosSetRealValue (↗ see page 133)

15.4.1.2 CosGetRealValue Function

```
ppReal CosGetRealValue(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

CosNewReal (↗ see page 132) CosSetRealValue (↗ see page 133)

15.4.1.3 CosSetRealValue Function

```
void CosSetRealValue(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

CosNewReal (↗ see page 132) CosGetRealValue (↗ see page 133)

15.4.2 Cos Integer Object

Functions

Function
CosNewInt (↗ see page 134)
CosGetIntValue (↗ see page 134)
CosSetIntValue (↗ see page 135)

15.4.2.1 CosNewInt Function

```
PDFCosHandle CosNewInt(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

CosFreeObj (↗ see page 128) CosGetType (↗ see page 127) CosGetIntValue (↗ see page 134) CosSetIntValue (↗ see page 135)

15.4.2.2 CosGetIntValue Function

```
ppInt32 CosGetIntValue(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

CosNewInt (↗ see page 134) CosSetIntValue (↗ see page 135)

15.4.2.3 CosSetIntValue Function

```
void CosSetIntValue(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

CosNewInt (see page 134) CosGetIntValue (see page 134)

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
CosNewName (see page 135)
CosGetNameValue (see page 136)
CosSetNameValue (see page 136)

15.5.1 CosNewName Function

```
PDFCosHandle CosNewName(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

CosFreeObj (see page 128) CosGetType (see page 127) CosGetNameValue (see page 136) CosSetNameValue (see page 136)

15.5.2 CosGetNameValue Function

```
ppAtom CosGetNameValue(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

CosNewName (see page 135) CosSetNameValue (see page 136)

15.5.3 CosSetNameValue Function

```
void CosSetNameValue(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

CosNewName (see page 135) CosGetNameValue (see page 136)

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
CosNewString (⧉ see page 137)
CosGetStringValue (⧉ see page 137)
CosSetStringValue (⧉ see page 138)

15.6.1 CosNewString Function

```
PDFCosHandle CosNewString(PDFDocHandle Doc, ppBool IsIndirect, char * String, ppInt32
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.
`ppInt32 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

CosFreeObj (⧉ see page 128) CosGetType (⧉ see page 127) CosGetStringValue (⧉ see page 137) CosSetStringValue (⧉ see page 138)

15.6.2 CosGetStringValue Function

```
char * CosGetStringValue(PDFCosHandle CosObject, ppInt32 * Length);
```

File

VSCosA.h

Parameters

Parameters	Description
`PDFCosHandle CosObject`	[in] The string Cos object whose value is obtained.
`ppInt32 * 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

CosNewString (⧉ see page 137) CosSetStringValue (⧉ see page 138)

15.6.3 CosSetStringValue Function

```
void CosSetStringValue(PDFCosHandle CosObject, char * String, ppInt32 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.
`ppInt32 Length`	[in] The new length of String.

Returns

None.

Description

Sets the new value for string Cos object.

See Also

CosNewString (↗ see page 137) CosGetStringValue (↗ see page 137)

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
CosNewArray (↗ see page 138)
CosArrayAppend (↗ see page 139)
CosArrayClear (↗ see page 139)
CosArrayCount (↗ see page 140)
CosArrayInsert (↗ see page 140)
CosArrayItem (↗ see page 141)
CosArrayRemove (↗ see page 141)

15.7.1 CosNewArray Function

```
PDFCosHandle CosNewArray(PDFDocHandle Doc, ppBool IsIndirect, ppInt32 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.
`ppInt32 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

CosFreeObj (↗ see page 128) CosGetType (↗ see page 127) CosArrayCount (↗ see page 140) CosArrayInsert (↗ see page 140) CosArrayAppend (↗ see page 139) CosArrayRemove (↗ see page 141) CosArrayClear (↗ see page 139)

15.7.2 CosArrayAppend Function

```
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

CosNewArray (↗ see page 138) CosArrayCount (↗ see page 140) CosArrayInsert (↗ see page 140) CosArrayAppend CosArrayRemove (↗ see page 141) CosArrayClear (↗ see page 139)

15.7.3 CosArrayClear Function

```
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

CosNewArray (↗ see page 138) CosArrayCount (↗ see page 140) CosArrayInsert (↗ see page 140) CosArrayAppend (↗ see page 139) CosArrayRemove (↗ see page 141) CosArrayClear

15.7.4 CosArrayCount Function

```
ppInt32 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

CosNewArray (↗ see page 138) CosArrayInsert (↗ see page 140) CosArrayAppend (↗ see page 139) CosArrayRemove (↗ see page 141) CosArrayClear (↗ see page 139)

15.7.5 CosArrayInsert Function

```
ppInt32 CosArrayInsert(PDFCosHandle CosObject, PDFCosHandle NewCosObject, ppInt32 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 140) (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

CosNewArray (↗ see page 138) CosArrayCount (↗ see page 140) CosArrayInsert CosArrayAppend (↗ see page 139) CosArrayRemove (↗ see page 141) CosArrayClear (↗ see page 139)

15.7.6 CosArrayItem Function

```
PDFCosHandle CosArrayItem(PDFCosHandle CosObject, ppInt32 Index);
```

File

VSCosA.h

Parameters

Parameters	Description
`PDFCosHandle CosObject`	[in] The array from which an element is obtained.
`ppInt32 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

CosNewArray (⧉ see page 138) CosArrayCount (⧉ see page 140) CosArrayInsert (⧉ see page 140) CosArrayAppend (⧉ see page 139) CosArrayRemove (⧉ see page 141) CosArrayClear (⧉ see page 139)

15.7.7 CosArrayRemove Function

Removes element from array.

```
void CosArrayRemove(PDFCosHandle CosObject, ppInt32 Index);
```

File

VSCosA.h

Parameters

Parameters	Description
`PDFCosHandle CosObject`	[in] The array Cos object to remove the member from it.
`ppInt32 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

CosNewArray (⧉ see page 138) CosArrayCount (⧉ see page 140) CosArrayInsert (⧉ see page 140) CosArrayAppend (⧉ see page 139) CosArrayRemove CosArrayClear (⧉ see page 139)

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
CosNewDict (↗ see page 142)
CosDictAppend (↗ see page 142)
CosDictClear (↗ see page 143)
CosDictCount (↗ see page 143)
CosDictRemoveKey (↗ see page 144)
CosDictGetPair (↗ see page 144)
CosDictValueByName (↗ see page 145)

15.8.1 CosNewDict Function

```
PDFCosHandle CosNewDict(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] If true, creates the dictionary object as an indirect object.
`ppInt32 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 127) CosFreeObj (↗ see page 128) CosDictCount (↗ see page 143) CosDictGetPair (↗ see page 144) CosDictAppend (↗ see page 142) CosDictRemoveKey (↗ see page 144) CosDictValueByName (↗ see page 145) CosDictClear (↗ see page 143)

15.8.2 CosDictAppend Function

```
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

CosNewDict (↗ see page 142) CosDictCount (↗ see page 143) CosDictGetPair (↗ see page 144) CosDictRemoveKey (↗ see page 144) CosDictValueByName (↗ see page 145) CosDictClear (↗ see page 143)

15.8.3 CosDictClear Function

```
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

CosNewDict (↗ see page 142) CosDictCount (↗ see page 143) CosDictGetPair (↗ see page 144) CosDictAppend (↗ see page 142) CosDictRemoveKey (↗ see page 144) CosDictValueByName (↗ see page 145)

15.8.4 CosDictCount Function

```
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

CosNewDict (↗ see page 142) CosDictGetPair (↗ see page 144) CosDictAppend (↗ see page 142) CosDictRemoveKey (↗ see page 144) CosDictValueByName (↗ see page 145) CosDictClear (↗ see page 143)

15.8.5 CosDictRemoveKey Function

```
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

CosNewDict (↗ see page 142) CosDictCount (↗ see page 143) CosDictGetPair (↗ see page 144) CosDictAppend (↗ see page 142) CosDictValueByName (↗ see page 145) CosDictClear (↗ see page 143)

15.8.6 CosDictGetPair Function

```
void CosDictGetPair(PDFCosHandle CosObject, ppInt32 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.
`ppInt32 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.

Returns

The number of key-value pair in the dictionary.

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

CosNewDict (see page 142) CosDictCount (see page 143) CosDictAppend (see page 142) CosDictRemoveKey (see page 144) CosDictValueByName (see page 145) CosDictClear (see page 143)

15.8.7 CosDictValueByName Function

```
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

CosNewDict (see page 142) CosDictCount (see page 143) CosDictGetPair (see page 144) CosDictAppend (see page 142) CosDictRemoveKey (see page 144) CosDictClear (see page 143)

Example

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

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
CosNewStream (see page 145)
CosStreamGetValue (see page 146)
CosStreamGetAttr (see page 146)

15.9.1 CosNewStream Function

```
PDFCosHandle CosNewStream(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

CosFreeObj (↗ see page 128) CosGetType (↗ see page 127) CosSteamGetAttr CosStreamGetValue (↗ see page 146)

15.9.2 CosStreamGetValue Function

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

File

VSCosA.h

Parameters

Parameters	Description
`PDFCosHandle CosObject`	[in] The stream whose attributes value is obtained.
`Lib`	[in] PDF Library Object.

Returns

The value of the stream Cos object.

Description

Gets a stream's value.

See Also

CosNewStream (↗ see page 145) CosStreamGetAttr (↗ see page 146)

15.9.3 CosStreamGetAttr Function

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

File

VSCosA.h

Parameters

Parameters	Description
`PDFCosHandle CosObject`	[in] The stream whose attributes dictionary is obtained.
`Lib`	[in] PDF Library Object.

Returns

The stream's attributes dictionary Cos object.

Description

Gets a stream's attributes dictionary.

See Also

CosNewStream (↗ see page 145) CosStreamGetValue (↗ see page 146)

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 148)
ULGrayToColor (↗ see page 148)
ULRGBToColor (↗ see page 149)

16.1.1 ULCMYKToColor Function

```
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 225) 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

```
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 225) 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

```
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 225) 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 150)
ULClearAtoms (⧉ see page 150)
ULExistsAtomForString (⧉ see page 150)
ULGetAtomCount (⧉ see page 151)
ULStringToAtom (⧉ see page 151)

16.2.1 ULAtomToString Function

```
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 151)

16.2.2 ULClearAtoms Function

```
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 ULExistsAtomForString Function

```
ppBool ULExistsAtomForString(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 151)

16.2.4 ULGetAtomCount Function

```
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

```
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 150)

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
ULWriteFile (⧉ see page 152)
ULSetFilePosition (⧉ see page 152)
ULReadFile (⧉ see page 153)
ULOpenFile (⧉ see page 153)
ULCloseFile (⧉ see page 154)
ULLookFileChar (⧉ see page 154)
ULGetFileSize (⧉ see page 154)
ULGetFileChar (⧉ see page 155)
ULGetFilePosition (⧉ see page 155)

16.3.1 ULWriteFile Function

```
ppInt32 ULWriteFile(PDFFileHandle FileHandle, void * Buffer, ppInt32 Length);
```

File

VSBaseA.h

Parameters

Parameters	Description
`PDFFileHandle FileHandle`	[in] PDF File Handle.
`void * Buffer`	[in] Source data buffer in memory.
`ppInt32 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

ULReadFile (⧉ see page 153)

16.3.2 ULSetFilePosition Function

```
ppInt32 ULSetFilePosition(PDFFileHandle FileHandle, ppInt32 Position);
```

File

VSBaseA.h

Parameters

Parameters	Description
`PDFFileHandle FileHandle`	[in] PDF File Handle.
`ppInt32 Position`	[in] File offset in bytes.

Returns

Position which is set.

Description

Sets file cursor to position (byte offset).

See Also

ULGetFilePosition (↗ see page 155)

16.3.3 ULReadFile Function

```
ppInt32 ULReadFile(PDFFileHandle FileHandle, void * Buffer, ppInt32 Length);
```

File

VSBaseA.h

Parameters

Parameters	Description
`PDFFileHandle FileHandle`	[in] PDF File Handle.
`void * Buffer`	[in] Destination buffer in memory for data.
`ppInt32 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

ULWriteFile (↗ see page 152)

16.3.4 ULOpenFile Function

```
PDFFileHandle ULOpenFile(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

ULCloseFile (↗ see page 154), ppFileOpenMode (↗ see page 199)

16.3.5 ULCloseFile Function

```
void ULCloseFile(PDFFileHandle FileHandle);
```

File

VSBaseA.h

Parameters

Parameters	Description
`PDFFileHandle FileHandle`	[in] PDF File Handle.

Returns

None.

Description

Closes PDF File.

See Also

ULOpenFile (see page 153)

16.3.6 ULLookFileChar Function

```
ppInt32 ULLookFileChar(PDFFileHandle FileHandle);
```

File

VSBaseA.h

Parameters

Parameters	Description
`PDFFileHandle FileHandle`	[in] PDF File Handle.

Returns

One character form file. If it returns -1 than it is EOF (end of file).

Description

Gets one character from file. Same as ULGetFileChar (see page 155), only file cursor stays on that place.

16.3.7 ULGetFileSize Function

```
ppInt32 ULGetFileSize(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.

16.3.8 ULGetFileChar Function

```
ppInt32 ULGetFileChar(PDFFileHandle FileHandle);
```

File

VSBaseA.h

Parameters

Parameters	Description
`PDFFileHandle FileHandle`	[in] PDF File Handle.

Returns

One character form file. If it returns -1 than it is EOF (end of file).

Description

Gets one character from file.

16.3.9 ULGetFilePosition Function

```
ppInt32 ULGetFilePosition(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

ULSetFilePosition (↗ see page 152)

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
ULStrmClear (↗ see page 156)
ULStrmClose (↗ see page 156)

ULStrmCopyToStrm (⧉ see page 157)
ULStrmGetPosition (⧉ see page 157)
ULStrmGetSize (⧉ see page 157)
ULStrmLookChar (⧉ see page 158)
ULStrmReadBuffer (⧉ see page 158)
ULStrmReadChar (⧉ see page 159)
ULStrmReadLine (⧉ see page 159)
ULStrmSetPosition (⧉ see page 159)
ULStrmSetSize (⧉ see page 160)
ULStrmWriteBuffer (⧉ see page 160)
ULStrmWriteChar (⧉ see page 161)
ULMemStrmRDOpen (⧉ see page 161)
ULMemStrmNew (⧉ see page 161)
ULFileStrmNew (⧉ see page 162)
ULFileHandleStrmNew (⧉ see page 162)

16.4.1 ULStrmClear Function

```
void ULStrmClear(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 ULStrmClose Function

```
void ULStrmClose(PDFStreamHandle Stream);
```

File

VSBaseA.h

Parameters

Parameters	Description
`PDFStreamHandle Stream`	[in] PDF Stream Handle.

Returns

None.

Description

Closes PDF Stream.

See Also

ULMemStrmNew (⧉ see page 161), ULFileStrmNew (⧉ see page 162)

16.4.3 ULStrmCopyToStrm Function

```
ppInt32 ULStrmCopyToStrm(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 ULStrmGetPosition Function

```
ppInt32 ULStrmGetPosition(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

ULStrmSetPosition (↗ see page 159)

16.4.5 ULStrmGetSize Function

```
ppInt32 ULStrmGetSize(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

ULStrmSetSize (↗ see page 160)

16.4.6 ULStrmLookChar Function

```
ppInt32 ULStrmLookChar(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 ULStrmReadChar (↗ see page 159), only stream position stays on that place.

See Also

ULStrmWriteChar (↗ see page 161)

16.4.7 ULStrmReadBuffer Function

```
ppInt32 ULStrmReadBuffer(PDFStreamHandle Stream, void * Buffer, ppInt32 Count);
```

File

VSBaseA.h

Parameters

Parameters	Description
`PDFStreamHandle Stream`	[in] PDF Stream Handle.
`void * Buffer`	[out] Memory buffer for data.
`ppInt32 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

ULStrmWriteBuffer (↗ see page 160)

16.4.8 ULStrmReadChar Function

```
ppInt32 ULStrmReadChar(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

ULStrmWriteChar (↗ see page 161)

16.4.9 ULStrmReadLine Function

```
char * ULStrmReadLine(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)

16.4.10 ULStrmSetPosition Function

```
ppInt32 ULStrmSetPosition(PDFStreamHandle Stream, ppInt32 NewPosition);
```

File

VSBaseA.h

Parameters

Parameters	Description
`PDFStreamHandle Stream`	[in] PDF Stream Handle.
`ppInt32 NewPosition`	[in] New position in bytes (offset from beginning).

Returns

Stream position after setting.

Description

Sets new Stream position.

See Also

ULStrmGetPosition (↗ see page 157)

16.4.11 ULStrmSetSize Function

```
ppInt32 ULStrmSetSize(PDFStreamHandle Stream, ppInt32 Size);
```

File

VSBaseA.h

Parameters

Parameters	Description
`PDFStreamHandle Stream`	[in] PDF Stream Handle.
`ppInt32 Size`	[in] New size of PDF Stream.

Returns

Stream size in bytes.

Description

Sets new Stream size in bytes. Enlarges stream capacity.

See Also

ULStrmGetSize (↗ see page 157)

16.4.12 ULStrmWriteBuffer Function

```
ppInt32 ULStrmWriteBuffer(PDFStreamHandle Stream, void * Buffer, ppInt32 Count);
```

File

VSBaseA.h

Parameters

Parameters	Description
`PDFStreamHandle Stream`	[out] PDF Stream Handle.
`void * Buffer`	[in] Memory buffer with data.
`ppInt32 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

ULStrmReadBuffer (↗ see page 158)

16.4.13 ULStrmWriteChar Function

```
ppInt32 ULStrmWriteChar(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

ULStrmReadChar (see page 159)

16.4.14 ULMemStrmRDOpen Function

```
PDFStreamHandle ULMemStrmRDOpen(PDFLibHandle Lib, void * Buffer, ppInt32 Length);
```

File

VSBaseA.h

Parameters

Parameters	Description
`PDFLibHandle Lib`	[in] PDF Library Handle.
`void * Buffer`	[in] Existed memory buffer.
`ppInt32 Length`	[in] Size of buffer in bytes.

Returns

Memory PDF Stream Handle.

Description

Converts memory buffer to PDF Stream.

See Also

ULStrmClose (see page 156)

16.4.15 ULMemStrmNew Function

```
PDFStreamHandle ULMemStrmNew(PDFLibHandle Lib, ppInt32 Size);
```

File

VSBaseA.h

Parameters

Parameters	Description
`PDFLibHandle Lib`	[in] PDF Library Handle.
`ppInt32 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

ULStrmClose (see page 156)

16.4.16 ULFileStrmNew Function

```
PDFStreamHandle ULFileStrmNew(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

ULStrmClose (see page 156)

16.4.17 ULFileHandleStrmNew Function

```
PDFStreamHandle ULFileHandleStrmNew(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

ULStrmClose (see page 156)

17 Structs, Records, Enums

17.1 _t_PDFCosHandle Struct

```
struct _t_PDFCosHandle {
  void * a;
  ppInt32 b;
};
```

File

VSTypes.h

Members

Members	Description
`void * a;`	pointer to Data Object
`ppInt32 b;`	Cos Object Identifier

Description

Primary Data Object Type

17.2 _t_TImageCompressionType Enumeration

```
enum _t_TImageCompressionType {
  itcJBIG2,
  itcFlate,
  itcJPEG,
  itcCCITT42D
};
```

File

VSImageA.h

Members

Members	Description
`itcJBIG2`	JBIG2 compression
`itcFlate`	FLATE compression
`itcJPEG`	JPEG compression
`itcCCITT42D`	CCITT 4 compression

Description

Available image compression types

17.3 _t_TKeyValidType Enumeration

```
enum _t_TKeyValidType {
  kvtInvalid = 0,
  kvtUser = 1,
  kvtOwner = 2
};
```

File

VSDocA.h

Members

Members	Description
`kvtInvalid = 0`	Invalid Password
`kvtUser = 1`	User Password
`kvtOwner = 2`	Owner Password

Description

Password Type of Crypted PDF Document. Password Validity.

17.4 _t_TPDFDocumentConnection Struct

```
struct _t_TPDFDocumentConnection {
  PDFDocHandle OldDocument;
  PDFDocHandle NewDocument;
  ppUns32 Size;
  PppUns32 Pages;
};
```

File

VSPagesA.h

Members

Members	Description
`PDFDocHandle OldDocument;`	Source Document where pages are taken from
`PDFDocHandle NewDocument;`	Destination Document where to put pages
`ppUns32 Size;`	Size of Queue of Page Numbers
`PppUns32 Pages;`	Queue Page Numbers, with repeated numbers possibility

Description

Document Connection Structure, Page Objects Container

17.5 _t_TPDFEncodingType Enumeration

```
enum _t_TPDFEncodingType {
  etPDFDocEncoding,
  etWinAnsiEncoding,
  etMacRomanEncoding,
  etStandardEncoding
};
```

File

VSFontA.h

Members

Members	Description
`etPDFDocEncoding`	PDF Document encoding
`etWinAnsiEncoding`	ANSI windows encoding

`etMacRomanEncoding`	Apple encoding
`etStandardEncoding`	Standard encoding

Description

Font encoding

17.6 _t_TPDFImageCompression Enumeration

```
enum _t_TPDFImageCompression {
  pdfiCCITT,
  pdfiJbig2,
  pdfiFlate
};
```

File

VSImageA.h

Members

Members	Description
`pdfiCCITT`	CCITT compression
`pdfiJbig2`	JBIG2 compression
`pdfiFlate`	FLATE compression

Description

Available black and white compressions

17.7 _t_TPDFInformation Enumeration

```
enum _t_TPDFInformation {
  piCreator = 0,
  piAuthor,
  piDate,
  piProducer,
  piTitle,
  piSubject,
  piKeyWords,
  piModificationData
};
```

File

VSDocA.h

Members

Members	Description
`piCreator = 0`	Information about creator of the PDF Document
`piAuthor`	Information about author of the PDF Document
`piDate`	Information about date of the creation PDF Document
`piProducer`	Information about producer of the PDF Document
`piTitle`	Information about title of the PDF Document
`piSubject`	Information about subject of the PDF Document
`piKeyWords`	Information about keywords
`piModificationData`	Information about modification data

Description

Document Description's Item Type

17.8 _t_TPDFPageOrientation Enumeration

```
enum _t_TPDFPageOrientation {
  poPagePortrait = 0,
  poPageLandScape
};
```

File

VSDocA.h

Members

Members	Description
`poPagePortrait = 0`	Orientation of Page is Portrait
`poPageLandScape`	Orientation of Page is Landscape

Description

Page Orientation Type

17.9 _t_TPDFPageSize Enumeration

```
enum _t_TPDFPageSize {
  psLetter = 0,
  psA4,
  psA3,
  psLegal,
  psB5,
  psC5,
  ps8x11,
  psB4,
  psA5,
  psFolio,
  psExecutive,
  psEnvB4,
  psEnvB5,
  psEnvC6,
  psEnvDL,
  psEnvMonarch,
  psEnv9,
  psEnv10,
  psEnv11
};
```

File

VSDocA.h

Members

Members	Description
`psLetter = 0`	Document's Page Size is 792 x 612
`psA4`	Document's Page Size is 842 x 595
`psA3`	Document's Page Size is 1190 x 842
`psLegal`	Document's Page Size is 1008 x 612
`psB5`	Document's Page Size is 728 x 516

`psC5`	Document's Page Size is 649 x 459
`ps8x11`	Document's Page Size is 792 x 595
`psB4`	Document's Page Size is 1031 x 728
`psA5`	Document's Page Size is 595 x 419
`psFolio`	Document's Page Size is 936 x 612
`psExecutive`	Document's Page Size is 756 x 522
`psEnvB4`	Document's Page Size is 1031 x 728
`psEnvB5`	Document's Page Size is 708 x 499
`psEnvC6`	Document's Page Size is 459 x 323
`psEnvDL`	Document's Page Size is 623 x 312
`psEnvMonarch`	Document's Page Size is 540 x 279
`psEnv9`	Document's Page Size is 639 x 279
`psEnv10`	Document's Page Size is 684 x 297
`psEnv11`	Document's Page Size is 747 x 324

Description

Type of usual PDF Document's Page Sizes

17.10 _t_TPDFProtectionType Enumeration

```
enum _t_TPDFProtectionType {
  pt40BitProtection = 0,
  pt128BitProtection = 1
};
```

File

VSDocA.h

Members

Members	Description
`pt40BitProtection = 0`	40 Bit protection key length
`pt128BitProtection = 1`	128 Bit protection key length

Description

Protection Key-Length Type of Crypted PDF Document

17.11 _t_TPDFStdandardFont Enumeration

```
enum _t_TPDFStdandardFont {
  stdfHelvetica,
  stdfHelveticaBold,
  stdfHelveticaOblique,
  stdfHelveticaBoldOblique,
  stdfTimesRoman,
  stdfTimesBold,
  stdfTimesItalic,
  stdfTimesBoldItalic,
  stdfCourier,
  stdfCourierBold,
  stdfCourierOblique,
  stdfCourierBoldOblique,
  stdfSymbol,
  stdfZapfDingbats
};
```

File

VSFontA.h

Members

Members	Description
`stdfHelvetica`	Helvetica font
`stdfHelveticaBold`	Helvetica Bold font
`stdfHelveticaOblique`	Helvetica Oblique font
`stdfHelveticaBoldOblique`	Helvetica Bold Obliquefont
`stdfTimesRoman`	Times Roman font
`stdfTimesBold`	Times Bold font
`stdfTimesItalic`	Times Italic font
`stdfTimesBoldItalic`	Times Bold Italicfont
`stdfCourier`	Courier font
`stdfCourierBold`	Courier Bold font
`stdfCourierOblique`	Courier Oblique font
`stdfCourierBoldOblique`	Courier BoldOblique font
`stdfSymbol`	Symbol font
`stdfZapfDingbats`	Zapf Dingbats font

Description

Standard 14 fonts enum

17.12 PDFActionType Enumeration

```
enum PDFActionType {
  pdfActionGoTo = 0,
  pdfActionGoToR,
  pdfActionLaunch,
  pdfActionThread,
  pdfActionURI,
  pdfActionSound,
  pdfActionMovie,
  pdfActionHide,
  pdfActionNamed,
  pdfActionSubmitForm,
  pdfActionResetForm,
  pdfActionImportData,
  pdfActionJavaScript,
  pdfActionUnknow
};
```

File

VSActionA.h

Members

Members	Description
`pdfActionGoTo = 0`	Go to a destination in the current document.
`pdfActionGoToR`	("Go-to remote") Go to a destination in another document.
`pdfActionLaunch`	Launch an application, usually to open a file.
`pdfActionThread`	Begin reading an article thread.
`pdfActionURI`	Resolve a uniform resource identifier.
`pdfActionSound`	Play a sound.
`pdfActionMovie`	Play a movie.
`pdfActionHide`	Set an annotation's Hidden flag.
`pdfActionNamed`	Execute an action predefined by the viewer application.
`pdfActionSubmitForm`	Send data to a uniform resource locater.

`pdfActionResetForm`	Set fields to their default values.
`pdfActionImportData`	Import field values from a file.
`pdfActionJavaScript`	Execute a JavaScript script.
`pdfActionUnknow`	Other unknown type.

Description

Available types of the PDF actions

17.13 PDFAnnatationIdentify Struct

```
struct PDFAnnatationIdentify {
  PDFAnnotationIdentifyType Type;
  union {
    PDFString AnnotationName;
    PDFAnnotationHandle AnnotationHandle;
  } Annotation;
};
```

File

VSActionA.h

Members

Members	Description
`PDFAnnotationIdentifyType Type;`	Type of the information

`union {`
`PDFString AnnotationName;`
`PDFAnnotationHandle AnnotationHandle;`
`} Annotation;`
	Annotation identifier

Description

Structure used to store information about annotation if PDF action

17.14 PDFBeadActionType Enumeration

```
enum PDFBeadActionType {
  taBeadHandle,
  taBeadIndex
};
```

File

VSActionA.h

Members

Members	Description
`taBeadHandle`	Destination stored in PDFBeadHandle (see page 179)
`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.

17.15 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

17.16 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 (*left*, *top*) positioned at the top-left corner of the window and the contents of the page magnified by the factor *zoom*. A null value for any of the parameters *left*, *top*, or *zoom* 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 top positioned at the *top* 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 left positioned at the *left* 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 *left*, *bottom*, *right*, and *top* 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 top positioned at the *top* 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 left positioned at the *left* 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

17.17 PDFLaunch Struct

```
struct PDFLaunch {
  PDFString FileName;
  PDFString DefaultDir;
  PDFString Operation;
  PDFString Params;
};
```

File

VSActionA.h

Members

Members	Description
`PDFString FileName;`	The application to be launched or the document to be opened or printed. If this entry is absent and the viewer application does not understand any of the alternative entries, it should do nothing.
`PDFString DefaultDir;`	A string specifying the default directory in standard DOS syntax.
`PDFString Operation;`	A string specifying the operation to perform:
`PDFString Params;`	A parameter string to be passed to the application.

Description

This structure used by PDFActionNewLaunch (see page 107) to create new Launch action

17.18 PDFMovieActionOperation Enumeration

```
enum PDFMovieActionOperation {
  moPlay,
  moStop,
  moPause,
  moResume
};
```

File

VSActionA.h

Members

Members	Description
`moPlay`	Start playing the movie.
`moStop`	Stop playing the movie.
`moPause`	Pause a playing movie.
`moResume`	Resume a paused movie.

Description

This enumeration defined types of the action for movie

17.19 PDFThreadActionParam Struct

```
struct PDFThreadActionParam {
  PDFThreadActionType DestThreadType;
  union {
    PDFThreadHandle ThreadHandle;
    ppInt32 ThreadIndex;
    PDFString 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;`
`PDFString 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

17.20 PDFThreadActionType Enumeration

```
enum PDFThreadActionType {
  taThreadHandle,
  taThreadIndex,
  taThreadTitle
};
```

File

VSActionA.h

Members

Members	Description
`taThreadHandle`	Destination is stored in PDFThreadHandle (see page 184)
`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

17.21 PDFVersion Enumeration

```
enum PDFVersion {
  pdfver10 = 0,
  pdfver11 = 1,
  pdfver12 = 2,
  pdfver13 = 3,
  pdfver14 = 4,
  pdfver15 = 5,
  pdfver16 = 6
};
```

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

Description

Type of supported PDF Document Versions

17.22 CosType Type

```
typedef enum {
  CosNull = 2,
  CosInteger = 4,
  CosReal = 8,
  CosBoolean = 16,
  CosName = 32,
  CosString = 64,
  CosDict = 128,
  CosArray = 256,
  CosStream = 512
} CosType;
```

File

VSCosA.h

Members

Members	Description
`CosNull = 2`	NULL Cos object
`CosInteger = 4`	Integer Cos Object
`CosReal = 8`	Real Cos Object
`CosBoolean = 16`	Boolean Cos Object
`CosName = 32`	Name Cos Object
`CosString = 64`	String Cos Object
`CosDict = 128`	Dictionary Cos Object
`CosArray = 256`	Array Cos Object

`CosStream = 512`	Stream Cos Object

Description

Available Cos objects

17.23 PBoolStream Type

```
typedef struct {
  ppBool isBool;
  ppBool BoolVal;
  PDFCosHandle DictBool;
} * PBoolStream, TBoolStream;
```

File

VSAnnotA.h

Members

Members	Description
`ppBool isBool;`	Stream is boolean
`ppBool BoolVal;`	Stream value
`PDFCosHandle DictBool;`	Stream body

Description

Boolean stream structure

17.24 PBSDict Type

```
typedef struct {
  int width;
  TBStyleName name;
  int * array;
  int arrLength;
  TEffectName BordEffect;
  int intensity;
} * PBSDict, TBSDict;
```

File

VSAnnotA.h

Members

Members	Description
`int width;`	(Optional) The border width in points. If this value is 0, no border is drawn.
`TBStyleName name;`	(Optional) The border style: bsnSolid (Solid) A solid rectangle surrounding the annotation. bsnDashed (Dashed) A dashed rectangle surrounding the annotation. The dash pattern is specified by the D entry (see below). bsnBeveled (Beveled) A simulated embossed rectangle that appears to be raised above the surface of the page. bsnInset (Inset) A simulated engraved rectangle that appears to be recessed below the surface of the page. bsnUnderline (Underline) A single line along the bottom of the annotation rectangle.
`int * array;`	A dash array defining a pattern of dashes and gaps to be used in drawing dashed border
`int arrLength;`	Dash array length
`TEffectName BordEffect;`	(Optional) A name representing the border effect to apply. Possible values are: enDefault: No effect: the border is as described by the annotation dictionary's BS entry. enCloudy: The border should appear “cloudy”. The width and dash array specified by BS are honored.

`int intensity;`	A number describing the intensity of the effect. Suggested values range from 0 to 2.

Description

Border style dictionary

17.25 PCaretAnnotDict Type

```
typedef struct {
  PDFCosHandle Popup;
  PDFActionHandle Action;
  PDFActionHandle AdditAction;
  annFlag AnFlags;
  PBSDict BSDict;
  TDeviceRGB Color;
  char * Contents;
  ppInt32 ContLength;
  char * DateTime;
  TAnotName IconName;
  ppBool Open;
  TPageRect Rectangle;
  char * TitleText;
  ppInt32 TTLength;
  float Transparency;
  int * RectBound;
  TCaretSymbol CaretSymbol;
} * PCaretAnnotDict, TCaretAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`PDFCosHandle Popup;`	(Optional) An indirect reference to a pop-up annotation for entering or editing the text is associated with this annotation.
`PDFActionHandle Action;`	(Optional) An action to be performed when the annotation is activated
`PDFActionHandle AdditAction;`	(Optional) An additional-actions dictionary defining the annotation’s behavior in response to various trigger events
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`PBSDict BSDict;`	(Optional) A border style dictionary specifying the characteristics of the annotation’s border
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation’s icon when its closed • The title bar of the annotation’s pop-up window • The border of a link annotation
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation’s contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be opened.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`char * TitleText;`	(Optional) The text label to be displayed in the title bar of the annotation’s pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
`ppInt32 TTLength;`	Length of title

`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.
`int * RectBound;`	(Optional) A set of 4 numbers describing the numerical differences between two rectangles: the Rect entry of the annotation and the actual boundaries of the underlying caret. Such a difference can occur, for example, when a paragraph symbol specified by Sy is displayed along with the caret. The 4 numbers correspond to the differences in default user space between the left, top, right and bottom coordinates of Rect and those of the caret, respectively. Each value must be greater than or equal to 0. The sum of the top and bottom differences must be less than the height of Rect, and the sum of the left and right differences must be less than the width of Rect.
`TCaretSymbol CaretSymbol;`	(Optional) A name specifying a symbol to be associated with the caret: csNewPar - A new paragraph symbol (“¶”) should be associated with the caret. csNone - No symbol should be associated with the caret.

Description

Caret annotation structure

17.26 PDeviceRGB Type

```
typedef struct {
  float red;
  float green;
  float blue;
} * PDeviceRGB, TDeviceRGB;
```

File

VSAnnotA.h

Members

Members	Description
`float red;`	(0.0 - 1.0)
`float green;`	(0.0 - 1.0)
`float blue;`	(0.0 - 1.0)

Description

Color structure

17.27 PDFActionHandle Type

```
typedef struct _t_PDFCosHandle {
  void * a;
  ppInt32 b;
} PDFCosHandle, PDFActionHandle, PDFAnnotationHandle, PDFOutlineHandle,
PDFDestinationHandle, PDFThreadHandle, PDFBeadHandle, PDFSoundHandle;
```

File

VSTypes.h

Members

Members	Description
`void * a;`	pointer to Data Object
`ppInt32 b;`	Cos Object Identifier

Description

Primary Data Object Type

17.28 PDFAnnatationIdentifyP Type

```
typedef struct PDFAnnatationIdentify {
  PDFAnnotationIdentifyType Type;
  union {
    PDFString AnnotationName;
    PDFAnnotationHandle AnnotationHandle;
  } Annotation;
} * PDFAnnatationIdentifyP;
```

File

VSActionA.h

Members

Members	Description
`PDFAnnotationIdentifyType Type;`	Type of the information

`union {`
`PDFString AnnotationName;`
`PDFAnnotationHandle AnnotationHandle;`
`} Annotation;`
	Annotation identifier

Description

Structure used to store information about annotation if PDF action

17.29 PDFAnnotationHandle Type

```
typedef struct _t_PDFCosHandle {
  void * a;
  ppInt32 b;
} PDFCosHandle, PDFActionHandle, PDFAnnotationHandle, PDFOutlineHandle,
PDFDestinationHandle, PDFThreadHandle, PDFBeadHandle, PDFSoundHandle;
```

File

VSTypes.h

Members

Members	Description
`void * a;`	pointer to Data Object
`ppInt32 b;`	Cos Object Identifier

Description

Primary Data Object Type

17.30 PDFAnnotationIdentifyType Type

```
typedef enum {
  acAnnotationHandle,
  acAnnotationName
} PDFAnnotationIdentifyType;
```

File

VSActionA.h

Members

Members	Description
`acAnnotationHandle`	Store annotation handle in action.
`acAnnotationName`	Store annotation name in action.

Description

This enumeration defined type of the store annotation info in action

17.31 PDFBeadHandle Type

```
typedef struct _t_PDFCosHandle {
  void * a;
  ppInt32 b;
} PDFCosHandle, PDFActionHandle, PDFAnnotationHandle, PDFOutlineHandle,
PDFDestinationHandle, PDFThreadHandle, PDFBeadHandle, PDFSoundHandle;
```

File

VSTypes.h

Members

Members	Description
`void * a;`	pointer to Data Object
`ppInt32 b;`	Cos Object Identifier

Description

Primary Data Object Type

17.32 PDFBlendMode Type

```
typedef enum {
  blmoNormal,
  blmoMultiply,
  blmoScreen,
  blmoOverlay,
  blmoDarken,
  blmoLighten,
  blmoColorDodge,
  blmoColorBurn,
  blmoHardLight,
  blmoSoftLight,
  blmoDifference,
  blmoExclusion
} PDFBlendMode;
```

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

17.33 PDFCosHandle Type

```
typedef struct _t_PDFCosHandle {
  void * a;
  ppInt32 b;
} PDFCosHandle, PDFActionHandle, PDFAnnotationHandle, PDFOutlineHandle,
PDFDestinationHandle, PDFThreadHandle, PDFBeadHandle, PDFSoundHandle;
```

File

VSTypes.h

Members

Members	Description
`void * a;`	pointer to Data Object
`ppInt32 b;`	Cos Object Identifier

Description

Primary Data Object Type

17.34 PDFDestinationHandle Type

```
typedef struct _t_PDFCosHandle {
  void * a;
  ppInt32 b;
} PDFCosHandle, PDFActionHandle, PDFAnnotationHandle, PDFOutlineHandle,
PDFDestinationHandle, PDFThreadHandle, PDFBeadHandle, PDFSoundHandle;
```

File

VSTypes.h

Members

Members	Description
`void * a;`	pointer to Data Object
`ppInt32 b;`	Cos Object Identifier

Description

Primary Data Object Type

17.35 PDFExplicitDest Type

```
typedef struct {
  PDFExplicitDestType Type;
  ppInt32 Page;
  ppReal a;
  ppReal b;
  ppReal c;
  ppReal d;
} PDFExplicitDest, * PPDFExplicitDest;
```

File

VSActionA.h

Members

Members	Description
`PDFExplicitDestType Type;`	Explicit destination type
`ppInt32 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

17.36 PDFLaunchP Type

```
typedef struct PDFLaunch {
  PDFString FileName;
  PDFString DefaultDir;
  PDFString Operation;
  PDFString Params;
} * PDFLaunchP;
```

File

VSActionA.h

Members

Members	Description
`PDFString FileName;`	The application to be launched or the document to be opened or printed. If this entry is absent and the viewer application does not understand any of the alternative entries, it should do nothing.
`PDFString DefaultDir;`	A string specifying the default directory in standard DOS syntax.
`PDFString Operation;`	A string specifying the operation to perform:

`PDFString Params;`	A parameter string to be passed to the application.

Description

This structure used by PDFActionNewLaunch (see page 107) to create new Launch action

17.37 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

17.38 PDFOutlineHandle Type

```
typedef struct _t_PDFCosHandle {
  void * a;
  ppInt32 b;
} PDFCosHandle, PDFActionHandle, PDFAnnotationHandle, PDFOutlineHandle,
PDFDestinationHandle, PDFThreadHandle, PDFBeadHandle, PDFSoundHandle;
```

File

VSTypes.h

Members

Members	Description
`void * a;`	pointer to Data Object
`ppInt32 b;`	Cos Object Identifier

Description

Primary Data Object Type

17.39 PDFRenderingIntents Type

```
typedef enum {
  ReIntAbsoluteColormetric,
  ReIntRelativeColorMetrics,
  ReIntSaturation,
  ReIntPerceptual
} PDFRenderingIntents;
```

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

17.40 PDFSoundHandle Type

```
typedef struct _t_PDFCosHandle {
  void * a;
  ppInt32 b;
} PDFCosHandle, PDFActionHandle, PDFAnnotationHandle, PDFOutlineHandle,
PDFDestinationHandle, PDFThreadHandle, PDFBeadHandle, PDFSoundHandle;
```

File

VSTypes.h

Members

Members	Description
`void * a;`	pointer to Data Object
`ppInt32 b;`	Cos Object Identifier

Description

Primary Data Object Type

17.41 PDFString Type

```
typedef struct {
  char * String;
  ppInt32 Length;
} PDFString;
```

File

VSTypes.h

Members

Members	Description
`char * String;`	Pointer to first character of String
`ppInt32 Length;`	Length of String Data

Description

PDF String Type

17.42 PDFThreadActionParamP Type

```
typedef struct PDFThreadActionParam {
  PDFThreadActionType DestThreadType;
  union {
    PDFThreadHandle ThreadHandle;
    ppInt32 ThreadIndex;
    PDFString 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;`
`PDFString 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

17.43 PDFThreadHandle Type

```
typedef struct _t_PDFCosHandle {
  void * a;
  ppInt32 b;
} PDFCosHandle, PDFActionHandle, PDFAnnotationHandle, PDFOutlineHandle,
PDFDestinationHandle, PDFThreadHandle, PDFBeadHandle, PDFSoundHandle;
```

File

VSTypes.h

Members

Members	Description
`void * a;`	pointer to Data Object
`ppInt32 b;`	Cos Object Identifier

Description

Sound Handle

17.44 PEffectName Type

```
typedef enum {
  enDefault,
  enCloudy
} * PEffectName, TEffectName;
```

File

VSAnnotA.h

Members

Members	Description
`enDefault`	Default border effects
`enCloudy`	Cloudy border effects

Description

Available names representing the border effects

17.45 PFileAttachAnnotDict Type

```
typedef struct {
  annFlag AnFlags;
  TDeviceRGB Color;
  char * Contents;
  ppInt32 ContLength;
  char * DateTime;
  ppBool Open;
  TPageRect Rectangle;
  float Transparency;
  char * FileName;
  int FNLength;
  TAttachType FSIcon;
} * PFileAttachAnnotDict, TFileAttachAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation’s icon when closed • The title bar of the annotation’s pop-up window • The border of a link annotation
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation’s contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.

`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.
`char * FileName;`	(Required) The filename associated with this annotation.
`int FNLength;`	Filename Length
`TAttachType FSIcon;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: atGraph, atPushPin atPaperclip, atTag Additional names may be supported as well.

Description

File attached annotation structure

17.46 PFreeAnnotDict Type

```
typedef struct {
  ppBool Open;
  TPageRect Rectangle;
  THighlighMode AnnotHighLight;
  PDFActionHandle PageAction;
  char * DA;
  ppInt32 DALength;
  TJustifyMode Quadding;
  PBSDict BSDict;
  PDFActionHandle Action;
  char * Contents;
  ppInt32 ContLength;
  TAnotName IconName;
  TDeviceRGB Color;
  char * DateTime;
  annFlag AnFlags;
  char * TitleText;
  ppInt32 TTLength;
  float Transparency;
  PDFCosHandle Popup;
  PDFActionHandle AdditAction;
} * PFreeAnnotDict, TFreeAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`THighlighMode AnnotHighLight;`	(Optional) The annotation’s highlighting mode, the visual effect to be used when the mouse button is pressed or held down inside its active area: hmInvert - (Invert) Invert the contents of the annotation rectangle. hmNone - (None) No highlighting. hmOutline - (Outline) Invert the annotation’s border. hmPush - (Push) Display the annotation’s down appearance, if any. If no down appearance is defined, offset the contents of the annotation rectangle to appear as if it were being “pushed” below the surface of the page.
`PDFActionHandle PageAction;`	(Optional) A URI action formerly associated with this annotation. When Web Capture changes an annotation from a URI to a go-to action, it uses this entry to save the data from the original URI action so that it can be changed back in case the target page for the go-to action is subsequently deleted.
`char * DA;`	(Required) The default appearance string to be used in formatting the text
`ppInt32 DALength;`	The default appearance string length
`TJustifyMode Quadding;`	(Optional) A code specifying the form of quadding (justification) to be used in displaying the annotation’s text: 0 Left-justified 1 Centered 2 Right-justified

`PBSDict BSDict;`	(Optional) A border style dictionary specifying the characteristics of the annotation's border
`PDFActionHandle Action;`	(Optional) An action to be performed when the annotation is activated
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation's contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation's icon when closed • The title bar of the annotation's pop-up window • The border of a link annotation
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`char * TitleText;`	(Optional) The text label to be displayed in the title bar of the annotation's pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
`ppInt32 TTLength;`	Length of title
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.
`PDFCosHandle Popup;`	(Optional) An indirect reference to a pop-up annotation for entering or editing the text associated with this annotation.
`PDFActionHandle AdditAction;`	(Optional) An additional-actions dictionary defining the annotation's behavior in response to various trigger events

Description

Free annotation structure

17.47 PLineAnnotDict Type

```
typedef struct {
  ppBool Open;
  TPageRect Rectangle;
  THighlighMode AnnotHighLight;
  PDFActionHandle PageAction;
  ppInt32 LineCoordinates[4];
  float InteriorColor[3];
  int LineEnding[2];
  PDFActionHandle Action;
  char * Contents;
  ppInt32 ContLength;
  PBSDict BSDict;
  TAnotName IconName;
  TDeviceRGB Color;
  char * DateTime;
  annFlag AnFlags;
  char * TitleText;
  ppInt32 TTLength;
  float Transparency;
  PDFCosHandle Popup;
  PDFActionHandle AdditAction;
} * PLineAnnotDict, TLineAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`THighlighMode AnnotHighLight;`	(Optional) The annotation's highlighting mode, the visual effect to be used when the mouse button is pressed or held down inside its active area: hmInvert - (Invert) Invert the contents of the annotation rectangle. hmNone - (None) No highlighting. hmOutline - (Outline) Invert the annotation's border. hmPush - (Push) Display the annotation's down appearance, if any. If no down appearance is defined, offset the contents of the annotation rectangle to appear as if it were being "pushed" below the surface of the page.
`PDFActionHandle PageAction;`	(Optional) A URI action formerly associated with this annotation. When Web Capture changes an annotation from a URI to a go-to action, it uses this entry to save the data from the original URI action so that it can be changed back in case the target page for the go-to action is subsequently deleted.
`ppInt32 LineCoordinates[4];`	(Required) An array of four numbers, [x1 y1 x2 y2], specifying the starting and ending coordinates of the line in default user space.
`float InteriorColor[3];`	(Optional) An array of three numbers in the range 0.0 to 1.0 specifying the components, in the DeviceRGB color space, of the interior color with which to fill the annotation's line endings (see Table 8.19). If this entry is absent, the interiors of the line endings are left transparent.
`int LineEnding[2];`	(Optional; PDF 1.4) An array of two names specifying the line ending styles to be used in drawing the line. The first and second elements of the array specify the line ending styles for the endpoints defined, respectively, by the first and second pairs of coordinates, (x1, y1) and (x2, y2), in the L array.
`PDFActionHandle Action;`	(Optional) An action to be performed when the annotation is activated
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation's contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`PBSDict BSDict;`	(Optional) A border style dictionary specifying the characteristics of the annotation's border
`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation's icon when closed • The title bar of the annotation's pop-up window • The border of a link annotation
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`char * TitleText;`	(Optional) The text label to be displayed in the title bar of the annotation's pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
`ppInt32 TTLength;`	Length of title
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.
`PDFCosHandle Popup;`	(Optional) An indirect reference to a pop-up annotation for entering or editing the text associated with this annotation.
`PDFActionHandle AdditAction;`	(Optional) An additional-actions dictionary defining the annotation's behavior in response to various trigger events

Description

Line annotation structure

17.48 PLinkAnnotDict Type

```
typedef struct {
  ppBool Open;
```

```
    TPageRect Rectangle;
    THighlighMode AnnotHighLight;
    PDFActionHandle PageAction;
    PDFActionHandle Action;
    char * Contents;
    ppInt32 ContLength;
    PBSDict BSDict;
    TAnotName IconName;
    TDeviceRGB Color;
    char * DateTime;
    annFlag AnFlags;
    char * TitleText;
    ppInt32 TTLength;
    float Transparency;
    PDFCosHandle Popup;
    PDFActionHandle AdditAction;
  } * PLinkAnnotDict, TLinkAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`THighlighMode AnnotHighLight;`	(Optional) The annotation’s highlighting mode, the visual effect to be used when the mouse button is pressed or held down inside its active area: hmInvert - (Invert) Invert the contents of the annotation rectangle. hmNone - (None) No highlighting. hmOutline - (Outline) Invert the annotation’s border. hmPush - (Push) Display the annotation’s down appearance, if any. If no down appearance is defined, offset the contents of the annotation rectangle to appear as if it were being “pushed” below the surface of the page.
`PDFActionHandle PageAction;`	(Optional) A URI action formerly associated with this annotation. When Web Capture changes an annotation from a URI to a go-to action, it uses this entry to save the data from the original URI action so that it can be changed back in case the target page for the go-to action is subsequently deleted.
`PDFActionHandle Action;`	(Optional) An action to be performed when the annotation is activated
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation’s contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`PBSDict BSDict;`	(Optional) A border style dictionary specifying the characteristics of the annotation’s border
`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation’s icon when closed • The title bar of the annotation’s pop-up window • The border of a link annotation
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`char * TitleText;`	(Optional) The text label to be displayed in the title bar of the annotation’s pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
`ppInt32 TTLength;`	Length of title
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.
`PDFCosHandle Popup;`	(Optional) An indirect reference to a pop-up annotation for entering or editing the text associated with this annotation.
`PDFActionHandle AdditAction;`	(Optional) An additional-actions dictionary defining the annotation’s behavior in response to various trigger events

Description

Link annotation structure

17.49 PMovieAnnotDict Type

```
typedef struct {
  annFlag AnFlags;
  TDeviceRGB Color;
  char * Contents;
  ppInt32 ContLength;
  char * DateTime;
  TPageRect Rectangle;
  char * FileName;
  ppInt32 FNLength;
  ppBool Activation;
  float Transparency;
} * PMovieAnnotDict, TMovieAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation's icon when closed • The title bar of the annotation's pop-up window • The border of a link annotation
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation's contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`char * FileName;`	(Optional) Movie filename.
`ppInt32 FNLength;`	Length of filename
`ppBool Activation;`	if it boolean value true, the movie should be played using default activation parameters; if it is false, the movie should not be played at all.
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.

Description

Movie annotation structure

17.50 PMovieDict Type

```
typedef struct {
  char * fileSpecific;
  int fsLength;
  TPagePoint Aspect;
  int Rotate;
  TBoolStream FlagStream;
} * PMovieDict, TMovieDict;
```

File

VSAnnotA.h

Members

Members	Description
`char * fileSpecific;`	Movie specification
`int fsLength;`	Movie specification length
`TPagePoint Aspect;`	The width and height of the movie's bounding box, in pixels.
`int Rotate;`	Rotation angle
`TBoolStream FlagStream;`	Boolean stream

Description

Movie dictionary structure

17.51 PPagePoint Type

```
typedef struct {
  int X;
  int Y;
} * PPagePoint, TPagePoint;
```

File

VSAnnotA.h

Members

Members	Description
`int X;`	Point abscissa
`int Y;`	Point ordinate

Description

Point structure

17.52 PPageRect Type

```
typedef struct {
  TPagePoint pt1;
  TPagePoint pt2;
} * PPageRect, TPageRect;
```

File

VSAnnotA.h

Members

Members	Description
`TPagePoint pt1;`	Top - left point
`TPagePoint pt2;`	Bottom - right point

Description

Rectangle structure

17.53 PPDFBorder Type

```
typedef struct {
  TPDFRect Rect;
  TPDFColor BorderColor;
  TPDFColor FillColor;
  ppReal Width;
} TPDFBorder, * PPDFBorder;
```

File

VSAcroObjects.h

Members

Members	Description
`TPDFRect Rect;`	Rectangle region for active area of Acro Form Object. Four coordinates - left, top, right and bottom of border. See TPDFRect (↗ see page 236)
`TPDFColor BorderColor;`	Color of border for displaying, see TPDFColor (↗ see page 225)
`TPDFColor FillColor;`	Color for filling inside area, background color
`ppReal Width;`	Width of border line in points

Description

Border Type. Specifies position of the annotation on the page. (Acroform Object Characteristic)

17.54 PPDFCheckBox Type

```
typedef struct {
  char * Name;
  ppUns32 Flag;
  TPDFFont Font;
  TPDFBorder Border;
  TPDFCheckBoxStyle Style;
  TPDFCheckBoxSign Sign;
  ppBool Value;
  PDFPaintContent PaintContentOn;
  PDFPaintContent PaintContentOff;
} TPDFCheckBox, * PPDFCheckBox, TPDFRadioButton, * PPDFRadioButton;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Name;`	Name(text string) of Acroform object, Name (text string) of Acroform field is used to export when the PDF document submitted.
`ppUns32 Flag;`	Specify the behavior of the annotation when is printed, rotated etc. AcroField Type (see TPDFAcroFlags)
`TPDFFont Font;`	Text font for appearance text label. Attributes of text written on or in the control. Use only Font's color for displaying Mark character.
`TPDFBorder Border;`	CheckBox or RadioButton rectangle specifies position of the annotation on the page. Border width and colors
`TPDFCheckBoxStyle Style;`	Style of CheckBox or RadioButton - rectangle or circle (see TPDFCheckBoxStyle (↗ see page 224)).
`TPDFCheckBoxSign Sign;`	Code of Mark character in CheckBox or RadioButton (see TPDFCheckBoxSign).
`ppBool Value;`	Value of CheckBox or RadioButton. Variable interactive value on Acroform (see VSAcroForm.h).

`PDFPaintContent PaintContentOn;`	Pointer to overload function to repaint CheckBox in checked state (optional), instead of default appearance.
`PDFPaintContent PaintContentOff;`	Pointer to overload function to repaint CheckBox in unchecked state (optional), instead of default appearance.

Description

Determination of CheckBox object and RadioButton object for setting on Acroform Content. Item selection.

17.55 PPDFComboBox Type

```
typedef struct {
  char * Name;
  ppUns32 Flag;
  char * Caption;
  TPDFFont Font;
  TPDFBorder Border;
  PDFPaintContent PaintContent;
} TPDFComboBox, * PPDFComboBox, TPDFListBox, * PPDFListBox, TPDFItemsBox;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Name;`	Name(text string) of Acroform object, Name (text string) of Acroform field used for export when the PDF document submitted.
`ppUns32 Flag;`	Specify the behavior of the annotation when is printed, rotated etc. AcroField Type (see TPDFAcroFlags)
`char * Caption;`	Default text string for appearance in ComboBox. Wasted in ListBox. Text is displayed in ComboBox when control is created.
`TPDFFont Font;`	Text font for appearance text label. Attributes of text written on or in the control.
`TPDFBorder Border;`	ComboBox or ListBox rectangle specifies position of the annotation on the page. Border width and colors.
`PDFPaintContent PaintContent;`	Pointer to overload function to repaint ComboBox (optional), instead of default.

Description

Determination of ComboBox object and ListBox object for setting on Acroform Content. Item(s) selection from items list.

17.56 PPDFDocumentConnection Type

```
typedef struct _t_TPDFDocumentConnection {
  PDFDocHandle OldDocument;
  PDFDocHandle NewDocument;
  ppUns32 Size;
  PppUns32 Pages;
} TPDFDocumentConnection, * PPDFDocumentConnection;
```

File

VSPagesA.h

Members

Members	Description
`PDFDocHandle OldDocument;`	Source Document where pages are taken from
`PDFDocHandle NewDocument;`	Destination Document where to put pages
`ppUns32 Size;`	Size of Queue of Page Numbers

`PppUns32 Pages;`	Queue Page Numbers, with repeated numbers possibility

Description

Document Connection Structure, Page Objects Container

17.57 PPDFDocumentSignature Type

```
typedef struct {
  char * Name;
  char * Owner;
  char * Reason;
  ppBool PKCS7;
  char * FileName;
  char * Password;
} TPDFDocumentSignature, * PPDFDocumentSignature;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Name;`	Name(text string) of Acroform object, Name (text string) of Acroform field is used to export when the PDF document is submitted.
`char * Owner;`	Owner of Signature, Person Name. Text string, for example "Ted Thompson"
`char * Reason;`	Reason of Sign this document. Text string, for example "I agree..."
`ppBool PKCS7;`	Boolean flag of coding type : true - 'Adobe.PPKMS' and 'adbe.pkcs7.sha1' crypt system sub filter false - 'Adobe.PPKLite' and 'adbe.x509.rsa_sha1' crypt system sub filter
`char * FileName;`	PFX Personal Signature FileName. Text string.
`char * Password;`	Owner Password for Personal Signature. Text string.

Description

Determination of Personal Invisible Signature object to sign Document

17.58 PPDFEditBox Type

```
typedef struct {
  char * Name;
  ppUns32 Flag;
  char * Caption;
  TPDFFont Font;
  TPDFBorder Border;
  ppUns32 MaxLen;
  TPDFAcroQuadding Align;
  PDFPaintContent PaintContent;
} TPDFEditBox, * PPDFEditBox;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Name;`	Name(text string) of Acroform object, Name (text string) of Acroform field is used to export when the PDF document is submitted.
`ppUns32 Flag;`	Specify the behavior of the annotation when is printed, rotated etc. AcroField Type (see TPDFAcroFlags)

`char * Caption;`	Default text string for appearance in EditBox. Text is displayed in EditBox when control is created.
`TPDFFont Font;`	Text font for appearance text label. Attributes of text written on or in the control.
`TPDFBorder Border;`	EditBox rectangle specifies position of the annotation on the page. Border width and colors.
`ppUns32 MaxLen;`	The maximum length of the field's text, in characters.
`TPDFAcroQuadding Align;`	Text alignment in edit box, justification of input text.
`PDFPaintContent PaintContent;`	Pointer to overload function to repaint edit box, instead of default (optional)

Description

Determination of Variable Text object for setting on Acroform Content. For text entering from document.

17.59 PPDFExplicitDest Type

```
typedef struct {
  PDFExplicitDestType Type;
  ppInt32 Page;
  ppReal a;
  ppReal b;
  ppReal c;
  ppReal d;
} PDFExplicitDest, * PPDFExplicitDest;
```

File

VSActionA.h

Members

Members	Description
`PDFExplicitDestType Type;`	Explicit destination type
`ppInt32 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

17.60 PPDFFont Type

```
typedef struct {
  TPDFFontID ID;
  ppReal Size;
  TPDFColor Color;
} TPDFFont, * PPDFFont;
```

File

VSAcroObjects.h

Members

Members	Description
`TPDFFontID ID;`	Identifier of font type, see TPDFFontID (↗ see page 228)
`ppReal Size;`	Size of font in points
`TPDFColor Color;`	Color of font for displaying, see TPDFColor (↗ see page 225)

Description

Font Type. Specifies text font properties of the annotation on the page. (Acroform Object Characteristic)

17.61 PPDFFontID Type

```
typedef struct {
  ppBool IsStdFont;
  union {
    ppInt32 Index;
    TPDFStdandardFont StandardFont;
  } From;
} TPDFFontID, * PPDFFontID;
```

File

VSAcroObjects.h

Members

Members	Description
`ppBool IsStdFont;`	Boolean flag an accessories to 14 standard fonts

`union {`
`ppInt32 Index;`
`TPDFStdandardFont StandardFont;`
`} From;`
	Font source

`ppInt32 Index;`	Index of loaded font in Document
`TPDFStdandardFont StandardFont;`	If standard font then it must be named, see TPDFStdandardFont (see page 237)

Description

Font Index Structure - to support early PDF versions (1.2 and below)

17.62 PPDFListBox Type

```
typedef struct {
  char * Name;
  ppUns32 Flag;
  char * Caption;
  TPDFFont Font;
  TPDFBorder Border;
  PDFPaintContent PaintContent;
} TPDFComboBox, * PPDFComboBox, TPDFListBox, * PPDFListBox, TPDFItemsBox;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Name;`	Name(text string) of Acroform object, Name (text string) of Acroform field used for export when the PDF document submitted.
`ppUns32 Flag;`	Specify the behavior of the annotation when is printed, rotated etc. AcroField Type (see TPDFAcroFlags)
`char * Caption;`	Default text string for appearance in ComboBox. Wasted in ListBox. Text is displayed in ComboBox when control is created.
`TPDFFont Font;`	Text font for appearance text label. Attributes of text written on or in the control.
`TPDFBorder Border;`	ComboBox or ListBox rectangle specifies position of the annotation on the page. Border width and colors.

`PDFPaintContent PaintContent;`	Pointer to overload function to repaint ComboBox (optional), instead of default.

Description

Determination of ComboBox object and ListBox object for setting on Acroform Content. Item(s) selection from items list.

17.63 PPDFPushButton Type

```
typedef struct {
  char * Name;
  ppUns32 Flag;
  char * Caption;
  TPDFFont Font;
  TPDFBorder Border;
  ppReal Miter;
  PDFPaintContent PaintContentUp;
  PDFPaintContent PaintContentDown;
} TPDFPushButton, * PPDFPushButton;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Name;`	Name(text string) of Acroform object, Name (text string) of Acroform field is used to export when the PDF document is submitted.
`ppUns32 Flag;`	Specify the behavior of the annotation when is printed, rotated etc. AcroField Type (see TPDFAcroFlags)
`char * Caption;`	Specifies a text string that identifies the control to the user. Text label for appearance on button.
`TPDFFont Font;`	Text font for appearance text label. Attributes of text written on or in the control.
`TPDFBorder Border;`	Pushbutton rectangle specifies position of the annotation on the page. Border width and colors.
`ppReal Miter;`	Miter of pushbutton, bevel size.
`PDFPaintContent PaintContentUp;`	Pointer to overload function to repaint pushbutton in normal state (optional), instead of default appearance.
`PDFPaintContent PaintContentDown;`	Pointer to overload function to repaint pushbutton in pressed state (optional), instead of default appearance

Description

Determination of Pushbutton object for setting on Acroform Content. Action selection. Submit action.

17.64 PPDFRadioButton Type

```
typedef struct {
  char * Name;
  ppUns32 Flag;
  TPDFFont Font;
  TPDFBorder Border;
  TPDFCheckBoxStyle Style;
  TPDFCheckBoxSign Sign;
  ppBool Value;
  PDFPaintContent PaintContentOn;
  PDFPaintContent PaintContentOff;
} TPDFCheckBox, * PPDFCheckBox, TPDFRadioButton, * PPDFRadioButton;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Name;`	Name(text string) of Acroform object, Name (text string) of Acroform field is used to export when the PDF document submitted.
`ppUns32 Flag;`	Specify the behavior of the annotation when is printed, rotated etc. AcroField Type (see TPDFAcroFlags)
`TPDFFont Font;`	Text font for appearance text label. Attributes of text written on or in the control. Use only Font's color for displaying Mark character.
`TPDFBorder Border;`	CheckBox or RadioButton rectangle specifies position of the annotation on the page. Border width and colors
`TPDFCheckBoxStyle Style;`	Style of CheckBox or RadioButton - rectangle or circle (see TPDFCheckBoxStyle (⧉ see page 224)).
`TPDFCheckBoxSign Sign;`	Code of Mark character in CheckBox or RadioButton (see TPDFCheckBoxSign).
`ppBool Value;`	Value of CheckBox or RadioButton. Variable interactive value on Acroform (see VSAcroForm.h).
`PDFPaintContent PaintContentOn;`	Pointer to overload function to repaint CheckBox in checked state (optional), instead of default appearance.
`PDFPaintContent PaintContentOff;`	Pointer to overload function to repaint CheckBox in unchecked state (optional), instead of default appearance.

Description

Determination of CheckBox object and RadioButton object for setting on Acroform Content. Item selection.

17.65 PPDFSignature Type

```
typedef struct {
  char * Name;
  TPDFBorder Border;
  TPDFAcroSigFlags SigFlags;
  TPDFAnnotFlags AnnotFlag;
} TPDFSignature, * PPDFSignature;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Name;`	Name(text string) of Acroform object, Name (text string) of Acroform field is used to export when the PDF document is submitted.
`TPDFBorder Border;`	Signature specifies position of the annotation on the page.
`TPDFAcroSigFlags SigFlags;`	A set of flags specifying various document-level characteristics related to signature fields. See TPDFAcroSigFlags
`TPDFAnnotFlags AnnotFlag;`	Specify the behavior of the annotation when is printed, rotated etc. See TPDFAnnotFlags

Description

Determination of Empty Signature object for setting on Acroform Content. Item to sign document

17.66 PPDFTextBox Type

```
typedef struct {
  char * Caption;
```

```
    TPDFFont Font;
    TPDFBorder Border;
    TPDFAcroQuadding Align;
    ppReal Orientation;
  } TPDFTextBox, * PPDFTextBox;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Caption;`	Text string of label
`TPDFFont Font;`	Font for displaying text, see TPDFFont (↗ see page 228)
`TPDFBorder Border;`	Border of text label, see TPDFBorder (↗ see page 223)
`TPDFAcroQuadding Align;`	Alignment text label option, see TPDFAcroQuadding (↗ see page 221)
`ppReal Orientation;`	Incline level, angle in degrees

Description

Determination of Text object for setting on Page or Acroform Content

17.67 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

17.68 PPolyAnnotDict Type

```
typedef struct {
  PDFActionHandle Action;
  PDFActionHandle AdditAction;
  annFlag AnFlags;
  PBSDict BSDict;
  PDFCosHandle BorderEffect;
  TDeviceRGB Color;
  int * Vertices;
  int VertLength;
  float * InteriorColor;
  char * Contents;
  ppInt32 ContLength;
  int LineEnding[2];
  char * DateTime;
  TAnotName IconName;
  TPolyType Type;
```

```
    ppBool Open;
    PDFCosHandle Popup;
    TPageRect Rectangle;
    char * TitleText;
    ppInt32 TTLength;
    float Transparency;
  } * PPolyAnnotDict, TPolyAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`PDFActionHandle Action;`	(Optional) An action to be performed when the annotation is activated
`PDFActionHandle AdditAction;`	(Optional) An additional-actions dictionary defining the annotation’s behavior in response to various trigger events
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`PBSDict BSDict;`	(Optional) A border style dictionary specifying the characteristics of the annotation’s border
`PDFCosHandle BorderEffect;`	(Optional) A border effect dictionary describing an effect applied to the border described by the BS entry
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation’s icon when closed • The title bar of the annotation’s pop-up window • The border of a link annotation
`int * Vertices;`	(Required) An array of numbers representing the alternating horizontal and vertical coordinates, respectively, of each vertex, in default user space.
`int VertLength;`	Length of the vertices array
`float * InteriorColor;`	(Optional) An array of three numbers in the range 0.0 to 1.0 specifying the components, in the DeviceRGB color space, of the interior color with which to fill the annotation’s line endings (see Table 8.19). If this entry is absent, the interiors of the line endings are left transparent.
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation’s contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`int LineEnding[2];`	(Optional; PDF 1.4) An array of two names specifying the line ending styles to be used in drawing the line. The first and second elements of the array specify the line ending styles for the endpoints defined, respectively, by the first and second pairs of coordinates, (x1, y1) and (x2, y2), in the L array.
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`TPolyType Type;`	(Required) The type of annotation that this dictionary describes; must be ptPolygon or ptPolyline for a polygon or polyline annotation, respectively.
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.
`PDFCosHandle Popup;`	(Optional) An indirect reference to a pop-up annotation for entering or editing the text associated with this annotation.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`char * TitleText;`	(Optional) The text label to be displayed in the title bar of the annotation’s pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
`ppInt32 TTLength;`	Length of title
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.

Description

Poly annotation structure

17.69 PPopupAnnotDict Type

```
typedef struct {
  PDFCosHandle Popup;
  PDFActionHandle Action;
  PDFActionHandle AdditAction;
  annFlag AnFlags;
  PBSDict BSDict;
  TDeviceRGB Color;
  char * Contents;
  ppInt32 ContLength;
  char * DateTime;
  TAnotName IconName;
  ppBool Open;
  TPageRect Rectangle;
  char * TitleText;
  ppInt32 TTLength;
  float Transparency;
  PDFCosHandle Parent;
} * PPopupAnnotDict, TPopupAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`PDFCosHandle Popup;`	(Optional) An indirect reference to a pop-up annotation for entering or editing the text associated with this annotation.
`PDFActionHandle Action;`	(Optional) An action to be performed when the annotation is activated
`PDFActionHandle AdditAction;`	(Optional) An additional-actions dictionary defining the annotation’s behavior in response to various trigger events
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`PBSDict BSDict;`	(Optional) A border style dictionary specifying the characteristics of the annotation’s border
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation’s icon when closed • The title bar of the annotation’s pop-up window • The border of a link annotation
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation’s contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`char * TitleText;`	(Optional) The text label to be displayed in the title bar of the annotation’s pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
`ppInt32 TTLength;`	Length of title
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.
`PDFCosHandle Parent;`	(Optional) The parent annotation with which this pop-up annotation is associated.

Description

Popup annotation structure

17.70 PRubberStampAnnotDict Type

```
typedef struct {
  PDFCosHandle AppearanceStream;
  PDFCosHandle Popup;
  PDFActionHandle Action;
  PDFActionHandle AdditAction;
  annFlag AnFlags;
  PBSDict BSDict;
  TDeviceRGB Color;
  char * Contents;
  ppInt32 ContLength;
  char * DateTime;
  TAnotName IconName;
  ppBool Open;
  TPageRect Rectangle;
  char * TitleText;
  ppInt32 TTLength;
  float Transparency;
  TIconName IconStyleName;
} * PRubberStampAnnotDict, TRubberStampAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`PDFCosHandle Popup;`	(Optional) An indirect reference to a pop-up annotation for entering or editing the text associated with this annotation.
`PDFActionHandle Action;`	(Optional) An action to be performed when the annotation is activated
`PDFActionHandle AdditAction;`	(Optional) An additional-actions dictionary defining the annotation’s behavior in response to various trigger events
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`PBSDict BSDict;`	(Optional) A border style dictionary specifying the characteristics of the annotation’s border
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation’s icon when closed • The title bar of the annotation’s pop-up window • The border of a link annotation
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation’s contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`char * TitleText;`	(Optional) The text label to be displayed in the title bar of the annotation’s pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
`ppInt32 TTLength;`	Length of title

`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.
`TIconName IconStyleName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: inApproved, inAsIs, inConfidential, inDepartmental, inDraft, inExperimental, inExpired, inFinal, inForComment, inForPublicRelease, inNotApproved, inNotForPublicRelease, inSold, inTopSecret Additional names may be supported as well.

Description

Rubber stamp annotation structure

17.71 PSCAnnotDict Type

```
typedef struct {
  ppBool Open;
  TPageRect Rectangle;
  THighlighMode AnnotHighLight;
  PDFActionHandle PageAction;
  float * InteriorColor;
  PDFActionHandle Action;
  char * Contents;
  ppInt32 ContLength;
  PBSDict BSDict;
  TAnotName IconName;
  TDeviceRGB Color;
  char * DateTime;
  annFlag AnFlags;
  char * TitleText;
  ppInt32 TTLength;
  float Transparency;
  PDFCosHandle Popup;
  PDFActionHandle AdditAction;
} * PSCAnnotDict, TSCAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`THighlighMode AnnotHighLight;`	(Optional) The annotation’s highlighting mode, the visual effect to be used when the mouse button is pressed or held down inside its active area: hmInvert - (Invert) Invert the contents of the annotation rectangle. hmNone - (None) No highlighting. hmOutline - (Outline) Invert the annotation’s border. hmPush - (Push) Display the annotation’s down appearance, if any. If no down appearance is defined, offset the contents of the annotation rectangle to appear as if it were being “pushed” below the surface of the page.
`PDFActionHandle PageAction;`	(Optional) A URI action formerly associated with this annotation. When Web Capture changes an annotation from a URI to a go-to action, it uses this entry to save the data from the original URI action so that it can be changed back in case the target page for the go-to action is subsequently deleted.
`float * InteriorColor;`	(Optional) An array of three numbers in the range 0.0 to 1.0 specifying the components, in the DeviceRGB color space, of the interior color with which to fill the annotation’s line endings (see Table 8.19). If this entry is absent, the interiors of the line endings are left transparent.
`PDFActionHandle Action;`	(Optional) An action to be performed when the annotation is activated
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation’s contents in human-readable form.

`ppInt32 ContLength;`	Count of characters in contents
`PBSDict BSDict;`	(Optional) A border style dictionary specifying the characteristics of the annotation's border
`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation's icon when closed • The title bar of the annotation's pop-up window • The border of a link annotation
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`char * TitleText;`	(Optional) The text label to be displayed in the title bar of the annotation's pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
`ppInt32 TTLength;`	Length of title
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.
`PDFCosHandle Popup;`	(Optional) An indirect reference to a pop-up annotation for entering or editing the text associated with this annotation.
`PDFActionHandle AdditAction;`	(Optional) An additional-actions dictionary defining the annotation's behavior in response to various trigger events

Description

Square and circle annotation structure

17.72 PSoundAnnotDict Type

```
typedef struct {
  annFlag AnFlags;
  TDeviceRGB Color;
  char * Contents;
  ppInt32 ContLength;
  char * DateTime;
  TAnotName IconName;
  ppBool Open;
  TPageRect Rectangle;
  char * Filename;
  ppInt32 FNLength;
  float Transparency;
} * PSoundAnnotDict, TSoundAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation's icon when closed • The title bar of the annotation's pop-up window • The border of a link annotation
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation's contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`char * DateTime;`	(Optional) The date and time when the annotation was created.

`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`char * Filename;`	(Required) Sound filename.
`ppInt32 FNLength;`	Filename length
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.

Description

Sound annotation structure

17.73 PSoundDict Type

```
typedef struct {
  int SamplingRate;
  int Channels;
  int BitsPerSample;
  TSEFormat EncFormat;
} * PSoundDict, TSoundDict;
```

File

VSAnnotA.h

Members

Members	Description
`int SamplingRate;`	Sound dictionary sampling rate
`int Channels;`	Sound dictionary channels
`int BitsPerSample;`	Sound dictionary bits per sample
`TSEFormat EncFormat;`	Sound dictionary encoding format structure

Description

Sound dictionary structure

17.74 TAnotFlags Type

```
typedef enum {
  afDefault = 0,
  afInvisible = 1,
  afHidden = 2,
  afPrint = 4,
  afNoZoom = 8,
  afNoRotate = 16,
  afNoView = 32,
  afReadOnly = 64
} TAnotFlags;
```

File

VSAnnotA.h

Members

Members	Description
`afDefault = 0`	Default annotation flag
`afInvisible = 1`	Invisible annotation flag
`afHidden = 2`	Hidden annotation flag
`afPrint = 4`	Print annotation flag
`afNoZoom = 8`	No Zoom annotation flag
`afNoRotate = 16`	No Rotate annotation flag
`afNoView = 32`	No View annotation flag
`afReadOnly = 64`	Read Only annotation flag

Description

Available names representing the annotation flags

17.75 TAnotName Type

```
typedef enum {
  anDefault = 0,
  anComment = 1,
  anHelp = 2,
  anInsert = 3,
  anKey = 4,
  anNewParagraph = 5,
  anParagraph = 6
} TAnotName;
```

File

VSAnnotA.h

Members

Members	Description
`anDefault = 0`	Note annotation
`anComment = 1`	Comment annotation
`anHelp = 2`	Help annotation
`anInsert = 3`	Insert annotation
`anKey = 4`	Key annotation
`anNewParagraph = 5`	New paragraph annotation
`anParagraph = 6`	Paragraph annotation

Description

Available annotations names

17.76 TAttachType Type

```
typedef enum {
  atGraph = 1,
  atPushPin = 2,
  atPaperclip = 3,
  atTag = 4
} TAttachType;
```

File

VSAnnotA.h

Members

Members	Description
`atGraph = 1`	Graph icon
`atPushPin = 2`	PushPin icon
`atPaperclip = 3`	Paper clip icon
`atTag = 4`	Tag icon

Description

Available names of an icon to be used in displaying the file attach annotation.

17.77 TBoolStream Type

```
typedef struct {
  ppBool isBool;
  ppBool BoolVal;
  PDFCosHandle DictBool;
} * PBoolStream, TBoolStream;
```

File

VSAnnotA.h

Members

Members	Description
`ppBool isBool;`	Stream is boolean
`ppBool BoolVal;`	Stream value
`PDFCosHandle DictBool;`	Stream body

Description

Boolean stream structure

17.78 TBSDict Type

```
typedef struct {
  int width;
  TBStyleName name;
  int * array;
  int arrLength;
  TEffectName BordEffect;
  int intensity;
} * PBSDict, TBSDict;
```

File

VSAnnotA.h

Members

Members	Description
`int width;`	(Optional) The border width in points. If this value is 0, no border is drawn.
`TBStyleName name;`	(Optional) The border style: bsnSolid (Solid) A solid rectangle surrounding the annotation. bsnDashed (Dashed) A dashed rectangle surrounding the annotation. The dash pattern is specified by the D entry (see below). bsnBeveled (Beveled) A simulated embossed rectangle that appears to be raised above the surface of the page. bsnInset (Inset) A simulated engraved rectangle that appears to be recessed below the surface of the page. bsnUnderline (Underline) A single line along the bottom of the annotation rectangle.

`int * array;`	A dash array defining a pattern of dashes and gaps to be used in drawing dashed border
`int arrLength;`	Dash array length
`TEffectName BordEffect;`	(Optional) A name representing the border effect to apply. Possible values are: enDefault: No effect: the border is as described by the annotation dictionary's BS entry. enCloudy: The border should appear “cloudy”. The width and dash array specified by BS are honored.
`int intensity;`	A number describing the intensity of the effect. Suggested values range from 0 to 2.

Description

Border style dictionary

17.79 TBStyleName Type

```
typedef enum {
  bsnSolid,
  bsnDashed,
  bsnBeveled,
  bsnInset,
  bsnUnderline
} TBStyleName;
```

File

VSAnnotA.h

Members

Members	Description
`bsnSolid`	Solid style
`bsnDashed`	Dashed style
`bsnBeveled`	Beveled style
`bsnInset`	Inset style
`bsnUnderline`	Underline style

Description

Available types of border styles

17.80 TCaretAnnotDict Type

```
typedef struct {
  PDFCosHandle Popup;
  PDFActionHandle Action;
  PDFActionHandle AdditAction;
  annFlag AnFlags;
  PBSDict BSDict;
  TDeviceRGB Color;
  char * Contents;
  ppInt32 ContLength;
  char * DateTime;
  TAnotName IconName;
  ppBool Open;
  TPageRect Rectangle;
  char * TitleText;
  ppInt32 TTLength;
  float Transparency;
  int * RectBound;
  TCaretSymbol CaretSymbol;
} * PCaretAnnotDict, TCaretAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`PDFCosHandle Popup;`	(Optional) An indirect reference to a pop-up annotation for entering or editing the text is associated with this annotation.
`PDFActionHandle Action;`	(Optional) An action to be performed when the annotation is activated
`PDFActionHandle AdditAction;`	(Optional) An additional-actions dictionary defining the annotation's behavior in response to various trigger events
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`PBSDict BSDict;`	(Optional) A border style dictionary specifying the characteristics of the annotation's border
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation's icon when its closed • The title bar of the annotation's pop-up window • The border of a link annotation
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation's contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be opened.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`char * TitleText;`	(Optional) The text label to be displayed in the title bar of the annotation's pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
`ppInt32 TTLength;`	Length of title
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.
`int * RectBound;`	(Optional) A set of 4 numbers describing the numerical differences between two rectangles: the Rect entry of the annotation and the actual boundaries of the underlying caret. Such a difference can occur, for example, when a paragraph symbol specified by Sy is displayed along with the caret. The 4 numbers correspond to the differences in default user space between the left, top, right and bottom coordinates of Rect and those of the caret, respectively. Each value must be greater than or equal to 0. The sum of the top and bottom differences must be less than the height of Rect, and the sum of the left and right differences must be less than the width of Rect.
`TCaretSymbol CaretSymbol;`	(Optional) A name specifying a symbol to be associated with the caret: csNewPar - A new paragraph symbol ("¶") should be associated with the caret. csNone - No symbol should be associated with the caret.

Description

Caret annotation structure

17.81 TCaretSymbol Type

```
typedef enum {
  csNone,
  csNewPar
} TCaretSymbol;
```

File

VSAnnotA.h

Members

Members	Description
`csNone`	A new paragraph symbol (“¶”) should be associated with the caret.
`csNewPar`	No symbol should be associated with the caret.

Description

Available names specifying a symbol to be associated with the caret.

17.82 TColorSpace Type

```
typedef enum {
  pdfDeviceGray,
  pdfDeviceRGB,
  pdfDeviceCMYK
} TColorSpace;
```

File

VSTypes.h

Members

Members	Description
`pdfDeviceGray`	Gray scale Color Device
`pdfDeviceRGB`	RGB Color Device
`pdfDeviceCMYK`	CMYK Color Device

Description

Color Device Type

17.83 TDeviceRGB Type

```
typedef struct {
  float red;
  float green;
  float blue;
} * PDeviceRGB, TDeviceRGB;
```

File

VSAnnotA.h

Members

Members	Description
`float red;`	(0.0 - 1.0)
`float green;`	(0.0 - 1.0)
`float blue;`	(0.0 - 1.0)

Description

Color structure

17.84 TEffectName Type

```
typedef enum {
  enDefault,
  enCloudy
} * PEffectName, TEffectName;
```

File

VSAnnotA.h

Members

Members	Description
`enDefault`	Default border effects
`enCloudy`	Cloudy border effects

Description

Available names representing the border effects

17.85 TFileAttachAnnotDict Type

```
typedef struct {
  annFlag AnFlags;
  TDeviceRGB Color;
  char * Contents;
  ppInt32 ContLength;
  char * DateTime;
  ppBool Open;
  TPageRect Rectangle;
  float Transparency;
  char * FileName;
  int FNLength;
  TAttachType FSIcon;
} * PFileAttachAnnotDict, TFileAttachAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation’s icon when closed • The title bar of the annotation’s pop-up window • The border of a link annotation
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation’s contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units

`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.
`char * FileName;`	(Required) The filename associated with this annotation.
`int FNLength;`	Filename Length
`TAttachType FSIcon;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: atGraph, atPushPin atPaperclip, atTag Additional names may be supported as well.

Description

File attached annotation structure

17.86 TFreeAnnotDict Type

```
typedef struct {
  ppBool Open;
  TPageRect Rectangle;
  THighlighMode AnnotHighLight;
  PDFActionHandle PageAction;
  char * DA;
  ppInt32 DALength;
  TJustifyMode Quadding;
  PBSDict BSDict;
  PDFActionHandle Action;
  char * Contents;
  ppInt32 ContLength;
  TAnotName IconName;
  TDeviceRGB Color;
  char * DateTime;
  annFlag AnFlags;
  char * TitleText;
  ppInt32 TTLength;
  float Transparency;
  PDFCosHandle Popup;
  PDFActionHandle AdditAction;
} * PFreeAnnotDict, TFreeAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`THighlighMode AnnotHighLight;`	(Optional) The annotation’s highlighting mode, the visual effect to be used when the mouse button is pressed or held down inside its active area: hmInvert - (Invert) Invert the contents of the annotation rectangle. hmNone - (None) No highlighting. hmOutline - (Outline) Invert the annotation’s border. hmPush - (Push) Display the annotation’s down appearance, if any. If no down appearance is defined, offset the contents of the annotation rectangle to appear as if it were being “pushed” below the surface of the page.
`PDFActionHandle PageAction;`	(Optional) A URI action formerly associated with this annotation. When Web Capture changes an annotation from a URI to a go-to action, it uses this entry to save the data from the original URI action so that it can be changed back in case the target page for the go-to action is subsequently deleted.
`char * DA;`	(Required) The default appearance string to be used in formatting the text
`ppInt32 DALength;`	The default appearance string length
`TJustifyMode Quadding;`	(Optional) A code specifying the form of quadding (justification) to be used in displaying the annotation’s text: 0 Left-justified 1 Centered 2 Right-justified
`PBSDict BSDict;`	(Optional) A border style dictionary specifying the characteristics of the annotation’s border

`PDFActionHandle Action;`	(Optional) An action to be performed when the annotation is activated
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation's contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation's icon when closed • The title bar of the annotation's pop-up window • The border of a link annotation
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`char * TitleText;`	(Optional) The text label to be displayed in the title bar of the annotation's pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
`ppInt32 TTLength;`	Length of title
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.
`PDFCosHandle Popup;`	(Optional) An indirect reference to a pop-up annotation for entering or editing the text associated with this annotation.
`PDFActionHandle AdditAction;`	(Optional) An additional-actions dictionary defining the annotation's behavior in response to various trigger events

Description

Free annotation structure

17.87 THighlighMode Type

```
typedef enum {
  hmInvert = 0,
  hmNone,
  hmOutline,
  hmPush
} THighlighMode;
```

File

VSAnnotA.h

Members

Members	Description
`hmInvert = 0`	Invert mode
`hmNone`	None highlight
`hmOutline`	Outline highlight
`hmPush`	Push highlight

Description

Available highlight modes

17.88 TIconName Type

```
typedef enum {
  inApproved,
  inAsIs,
  inConfidential,
  inDepartmental,
  inDraft,
  inExperimental,
  inExpired,
  inFinal,
  inForComment,
  inForPublicRelease,
  inNotApproved,
  inNotForPublicRelease,
  inSold,
  inTopSecret
} TIconName;
```

File

VSAnnotA.h

Members

Members	Description
`inApproved`	Approved icon type
`inAsIs`	AsIs icon type
`inConfidential`	Confidential icon type
`inDepartmental`	Departmental icon type
`inDraft`	Draft icon type
`inExperimental`	Experimental icon type
`inExpired`	Expired icon type
`inFinal`	Final icon type
`inForComment`	For comment icon type
`inForPublicRelease`	For public release icon type
`inNotApproved`	Not approved icon type
`inNotForPublicRelease`	Not for public release icon type
`inSold`	Sold icon type
`inTopSecret`	Top secret icon type

Description

Available names of an icon to be used in displaying the rubber stamp annotation.

17.89 TImageCompressionType Type

```
typedef enum _t_TImageCompressionType {
  itcJBIG2,
  itcFlate,
  itcJPEG,
  itcCCITT42D
} TImageCompressionType;
```

File

VSImageA.h

Members

Members	Description
`itcJBIG2`	JBIG2 compression
`itcFlate`	FLATE compression
`itcJPEG`	JPEG compression
`itcCCITT42D`	CCITT 4 compression

Description

Available image compression types

17.90 TJustifyMode Type

```
typedef enum {
  jmLeft = 0,
  jmCenter = 1,
  jmRight = 2
} TJustifyMode;
```

File

VSAnnotA.h

Members

Members	Description
`jmLeft = 0`	Left text justify mode
`jmCenter = 1`	Center text justify mode
`jmRight = 2`	Right text justify mode

Description

Available annotations text justify mode

17.91 TKeyValidType Type

```
typedef enum _t_TKeyValidType {
  kvtInvalid = 0,
  kvtUser = 1,
  kvtOwner = 2
} TKeyValidType;
```

File

VSDocA.h

Members

Members	Description
`kvtInvalid = 0`	Invalid Password
`kvtUser = 1`	User Password
`kvtOwner = 2`	Owner Password

Description

Password Type of Crypted PDF Document. Password Validity.

17.92 TLineAnnotDict Type

```
typedef struct {
  ppBool Open;
  TPageRect Rectangle;
  THighlighMode AnnotHighLight;
  PDFActionHandle PageAction;
```

```
    ppInt32 LineCoordinates[4];
    float InteriorColor[3];
    int LineEnding[2];
    PDFActionHandle Action;
    char * Contents;
    ppInt32 ContLength;
    PBSDict BSDict;
    TAnotName IconName;
    TDeviceRGB Color;
    char * DateTime;
    annFlag AnFlags;
    char * TitleText;
    ppInt32 TTLength;
    float Transparency;
    PDFCosHandle Popup;
    PDFActionHandle AdditAction;
  } * PLineAnnotDict, TLineAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`THighlighMode AnnotHighLight;`	(Optional) The annotation’s highlighting mode, the visual effect to be used when the mouse button is pressed or held down inside its active area: hmInvert - (Invert) Invert the contents of the annotation rectangle. hmNone - (None) No highlighting. hmOutline - (Outline) Invert the annotation’s border. hmPush - (Push) Display the annotation’s down appearance, if any. If no down appearance is defined, offset the contents of the annotation rectangle to appear as if it were being “pushed” below the surface of the page.
`PDFActionHandle PageAction;`	(Optional) A URI action formerly associated with this annotation. When Web Capture changes an annotation from a URI to a go-to action, it uses this entry to save the data from the original URI action so that it can be changed back in case the target page for the go-to action is subsequently deleted.
`ppInt32 LineCoordinates[4];`	(Required) An array of four numbers, [x1 y1 x2 y2], specifying the starting and ending coordinates of the line in default user space.
`float InteriorColor[3];`	(Optional) An array of three numbers in the range 0.0 to 1.0 specifying the components, in the DeviceRGB color space, of the interior color with which to fill the annotation’s line endings (see Table 8.19). If this entry is absent, the interiors of the line endings are left transparent.
`int LineEnding[2];`	(Optional; PDF 1.4) An array of two names specifying the line ending styles to be used in drawing the line. The first and second elements of the array specify the line ending styles for the endpoints defined, respectively, by the first and second pairs of coordinates, (x1, y1) and (x2, y2), in the L array.
`PDFActionHandle Action;`	(Optional) An action to be performed when the annotation is activated
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation’s contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`PBSDict BSDict;`	(Optional) A border style dictionary specifying the characteristics of the annotation’s border
`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation’s icon when closed • The title bar of the annotation’s pop-up window • The border of a link annotation
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`char * TitleText;`	(Optional) The text label to be displayed in the title bar of the annotation’s pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
`ppInt32 TTLength;`	Length of title

`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.
`PDFCosHandle Popup;`	(Optional) An indirect reference to a pop-up annotation for entering or editing the text associated with this annotation.
`PDFActionHandle AdditAction;`	(Optional) An additional-actions dictionary defining the annotation's behavior in response to various trigger events

Description

Line annotation structure

17.93 TLineEndingStyle Type

```
typedef enum {
  lesNone = 0,
  lesSquare = 1,
  lesCircle = 2,
  lesDiamond = 3,
  lesOpenArrow = 4,
  lesClosedArrow = 5
} TLineEndingStyle;
```

File

VSAnnotA.h

Members

Members	Description
`lesNone = 0`	No line ending
`lesSquare = 1`	A square filled with the annotation's interior color
`lesCircle = 2`	A circle filled with the annotation's interior color, if any
`lesDiamond = 3`	A diamond shape filled with the annotation's interior color.
`lesOpenArrow = 4`	Two short lines meeting in an acute angle, forming an open arrowhead
`lesClosedArrow = 5`	Two short lines meeting in an acute angle as in the OpenArrow style, connected by a third line to form a triangular closed arrowhead filled with the annotation's interior color, if any

Description

Available types of the PDF line ending styles

17.94 TLinkAnnotDict Type

```
typedef struct {
  ppBool Open;
  TPageRect Rectangle;
  THighlighMode AnnotHighLight;
  PDFActionHandle PageAction;
  PDFActionHandle Action;
  char * Contents;
  ppInt32 ContLength;
  PBSDict BSDict;
  TAnotName IconName;
  TDeviceRGB Color;
  char * DateTime;
  annFlag AnFlags;
  char * TitleText;
  ppInt32 TTLength;
  float Transparency;
```

```
    PDFCosHandle Popup;
    PDFActionHandle AdditAction;
  } * PLinkAnnotDict, TLinkAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`THighlighMode AnnotHighLight;`	(Optional) The annotation’s highlighting mode, the visual effect to be used when the mouse button is pressed or held down inside its active area: hmInvert - (Invert) Invert the contents of the annotation rectangle. hmNone - (None) No highlighting. hmOutline - (Outline) Invert the annotation’s border. hmPush - (Push) Display the annotation’s down appearance, if any. If no down appearance is defined, offset the contents of the annotation rectangle to appear as if it were being “pushed” below the surface of the page.
`PDFActionHandle PageAction;`	(Optional) A URI action formerly associated with this annotation. When Web Capture changes an annotation from a URI to a go-to action, it uses this entry to save the data from the original URI action so that it can be changed back in case the target page for the go-to action is subsequently deleted.
`PDFActionHandle Action;`	(Optional) An action to be performed when the annotation is activated
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation’s contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`PBSDict BSDict;`	(Optional) A border style dictionary specifying the characteristics of the annotation’s border
`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation’s icon when closed • The title bar of the annotation’s pop-up window • The border of a link annotation
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`char * TitleText;`	(Optional) The text label to be displayed in the title bar of the annotation’s pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
`ppInt32 TTLength;`	Length of title
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.
`PDFCosHandle Popup;`	(Optional) An indirect reference to a pop-up annotation for entering or editing the text associated with this annotation.
`PDFActionHandle AdditAction;`	(Optional) An additional-actions dictionary defining the annotation’s behavior in response to various trigger events

Description

Link annotation structure

17.95 TMovieAnnotDict Type

```
  typedef struct {
    annFlag AnFlags;
    TDeviceRGB Color;
    char * Contents;
    ppInt32 ContLength;
    char * DateTime;
    TPageRect Rectangle;
    char * FileName;
    ppInt32 FNLength;
    ppBool Activation;
    float Transparency;
  } * PMovieAnnotDict, TMovieAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation's icon when closed • The title bar of the annotation's pop-up window • The border of a link annotation
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation's contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`char * FileName;`	(Optional) Movie filename.
`ppInt32 FNLength;`	Length of filename
`ppBool Activation;`	if it boolean value true, the movie should be played using default activation parameters; if it is false, the movie should not be played at all.
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.

Description

Movie annotation structure

17.96 TMovieDict Type

```
  typedef struct {
    char * fileSpecific;
    int fsLength;
    TPagePoint Aspect;
    int Rotate;
    TBoolStream FlagStream;
  } * PMovieDict, TMovieDict;
```

File

VSAnnotA.h

Members

Members	Description
`char * fileSpecific;`	Movie specification
`int fsLength;`	Movie specification length
`TPagePoint Aspect;`	The width and height of the movie's bounding box, in pixels.
`int Rotate;`	Rotation angle
`TBoolStream FlagStream;`	Boolean stream

Description

Movie dictionary structure

17.97 TPagePoint Type

```
typedef struct {
  int X;
  int Y;
} * PPagePoint, TPagePoint;
```

File

VSAnnotA.h

Members

Members	Description
`int X;`	Point abscissa
`int Y;`	Point ordinate

Description

Point structure

17.98 TPageRect Type

```
typedef struct {
  TPagePoint pt1;
  TPagePoint pt2;
} * PPageRect, TPageRect;
```

File

VSAnnotA.h

Members

Members	Description
`TPagePoint pt1;`	Top - left point
`TPagePoint pt2;`	Bottom - right point

Description

Rectangle structure

17.99 TPDFAcroAppearance Type

```
typedef enum {
  PDFAcroNormalAppearance = 0,
  PDFAcroRolloverAppearance,
  PDFAcroDownAppearance
} TPDFAcroAppearance;
```

File

VSAcroObjects.h

Members

Members	Description
`PDFAcroNormalAppearance = 0`	The normal appearance is used when the annotation is not interacting with the user. This is also the appearance that is used for printing the annotation.
`PDFAcroRolloverAppearance`	The rollover appearance is used when the user moves the cursor into the annotation’s active area without pressing the mouse button.
`PDFAcroDownAppearance`	The down appearance is used when the mouse button is pressed or held down within the annotation’s active area.

Description

Acro Form Object Appearance Type. For point to appearance type.

17.100 TPDFAcroEventType Type

```
typedef enum {
  PDFAcroEventTypeActivate = 0,
  PDFAcroEventTypeEnter,
  PDFAcroEventTypeExit,
  PDFAcroEventTypePress,
  PDFAcroEventTypeRelease,
  PDFAcroEventTypeFocusOn,
  PDFAcroEventTypeFocusOff,
  PDFAcroEventTypeKeystroke,
  PDFAcroEventTypeFormat,
  PDFAcroEventTypeValidate,
  PDFAcroEventTypeCalculate
} TPDFAcroEventType;
```

File

VSAcroObjects.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.

17.101 TPDFAcroQuadding Type

```
typedef enum {
  PDFAcroQuaddingLeftTop = 0,
  PDFAcroQuaddingTop,
  PDFAcroQuaddingRightTop,
  PDFAcroQuaddingLeft,
  PDFAcroQuaddingCenter,
  PDFAcroQuaddingRight,
  PDFAcroQuaddingLeftBottom,
  PDFAcroQuaddingBottom,
  PDFAcroQuaddingRightBottom
} TPDFAcroQuadding;
```

File

VSAcroObjects.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.

17.102 TPDFAcroType Type

```
typedef enum {
  PDFAcroTypeUnknown = 0,
  PDFAcroTypePushButton,
  PDFAcroTypeCheckBox,
  PDFAcroTypeRadioButton,
  PDFAcroTypeEditBox,
  PDFAcroTypeComboBox,
  PDFAcroTypeListBox,
  PDFAcroTypeSignature
} TPDFAcroType;
```

File

VSAcroObjects.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.

17.103 TPDFBorder Type

```
typedef struct {
  TPDFRect Rect;
  TPDFColor BorderColor;
  TPDFColor FillColor;
  ppReal Width;
} TPDFBorder, * PPDFBorder;
```

File

VSAcroObjects.h

Members

Members	Description
`TPDFRect Rect;`	Rectangle region for active area of Acro Form Object. Four coordinates - left, top, right and bottom of border. See TPDFRect (⧉ see page 236)
`TPDFColor BorderColor;`	Color of border for displaying, see TPDFColor (⧉ see page 225)
`TPDFColor FillColor;`	Color for filling inside area, background color
`ppReal Width;`	Width of border line in points

Description

Border Type. Specifies position of the annotation on the page. (Acroform Object Characteristic)

17.104 TPDFCheckBox Type

```
typedef struct {
  char * Name;
  ppUns32 Flag;
  TPDFFont Font;
  TPDFBorder Border;
  TPDFCheckBoxStyle Style;
  TPDFCheckBoxSign Sign;
  ppBool Value;
  PDFPaintContent PaintContentOn;
  PDFPaintContent PaintContentOff;
} TPDFCheckBox, * PPDFCheckBox, TPDFRadioButton, * PPDFRadioButton;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Name;`	Name(text string) of Acroform object, Name (text string) of Acroform field is used to export when the PDF document submitted.
`ppUns32 Flag;`	Specify the behavior of the annotation when is printed, rotated etc. AcroField Type (see TPDFAcroFlags)
`TPDFFont Font;`	Text font for appearance text label. Attributes of text written on or in the control. Use only Font's color for displaying Mark character.
`TPDFBorder Border;`	CheckBox or RadioButton rectangle specifies position of the annotation on the page. Border width and colors
`TPDFCheckBoxStyle Style;`	Style of CheckBox or RadioButton - rectangle or circle (see TPDFCheckBoxStyle (⧉ see page 224)).
`TPDFCheckBoxSign Sign;`	Code of Mark character in CheckBox or RadioButton (see TPDFCheckBoxSign).
`ppBool Value;`	Value of CheckBox or RadioButton. Variable interactive value on Acroform (see VSAcroForm.h).

`PDFPaintContent PaintContentOn;`	Pointer to overload function to repaint CheckBox in checked state (optional), instead of default appearance.
`PDFPaintContent PaintContentOff;`	Pointer to overload function to repaint CheckBox in unchecked state (optional), instead of default appearance.

Description

Determination of CheckBox object and RadioButton object for setting on Acroform Content. Item selection.

17.105 TPDFCheckBoxStyle Type

```
typedef enum {
  cbfRectangle = 0,
  cbfCircle
} TPDFCheckBoxStyle;
```

File

VSAcroObjects.h

Members

Members	Description
`cbfRectangle = 0`	Rectangle style
`cbfCircle`	Circle style

Description

Type of CheckBox Style.

17.106 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

17.107 TPDFColor Type

```
typedef struct {
  TColorSpace Device;
  union {
    ppReal Gray;
    TPDFRGBColor RGB;
    TPDFCMYKColor CMYK;
  } Color;
} TPDFColor;
```

File

VSTypes.h

Members

Members	Description
`TColorSpace 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

17.108 TPDFComboBox Type

```
typedef struct {
  char * Name;
  ppUns32 Flag;
  char * Caption;
  TPDFFont Font;
  TPDFBorder Border;
  PDFPaintContent PaintContent;
} TPDFComboBox, * PPDFComboBox, TPDFListBox, * PPDFListBox, TPDFItemsBox;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Name;`	Name(text string) of Acroform object, Name (text string) of Acroform field used for export when the PDF document submitted.
`ppUns32 Flag;`	Specify the behavior of the annotation when is printed, rotated etc. AcroField Type (see TPDFAcroFlags)
`char * Caption;`	Default text string for appearance in ComboBox. Wasted in ListBox. Text is displayed in ComboBox when control is created.
`TPDFFont Font;`	Text font for appearance text label. Attributes of text written on or in the control.
`TPDFBorder Border;`	ComboBox or ListBox rectangle specifies position of the annotation on the page. Border width and colors.
`PDFPaintContent PaintContent;`	Pointer to overload function to repaint ComboBox (optional), instead of default.

Description

Determination of ComboBox object and ListBox object for setting on Acroform Content. Item(s) selection from items list.

17.109 TPDFDocumentConnection Type

```
typedef struct _t_TPDFDocumentConnection {
  PDFDocHandle OldDocument;
  PDFDocHandle NewDocument;
  ppUns32 Size;
  PppUns32 Pages;
} TPDFDocumentConnection, * PPDFDocumentConnection;
```

File

VSPagesA.h

Members

Members	Description
`PDFDocHandle OldDocument;`	Source Document where pages are taken from
`PDFDocHandle NewDocument;`	Destination Document where to put pages
`ppUns32 Size;`	Size of Queue of Page Numbers
`PppUns32 Pages;`	Queue Page Numbers, with repeated numbers possibility

Description

Document Connection Structure, Page Objects Container

17.110 TPDFDocumentSignature Type

```
typedef struct {
  char * Name;
  char * Owner;
  char * Reason;
  ppBool PKCS7;
  char * FileName;
  char * Password;
} TPDFDocumentSignature, * PPDFDocumentSignature;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Name;`	Name(text string) of Acroform object, Name (text string) of Acroform field is used to export when the PDF document is submitted.
`char * Owner;`	Owner of Signature, Person Name. Text string, for example "Ted Thompson"
`char * Reason;`	Reason of Sign this document. Text string, for example "I agree..."
`ppBool PKCS7;`	Boolean flag of coding type : true - 'Adobe.PPKMS' and 'adbe.pkcs7.sha1' crypt system sub filter false - 'Adobe.PPKLite' and 'adbe.x509.rsa_sha1' crypt system sub filter
`char * FileName;`	PFX Personal Signature FileName. Text string.
`char * Password;`	Owner Password for Personal Signature. Text string.

Description

Determination of Personal Invisible Signature object to sign Document

17.111 TPDFEditBox Type

```
typedef struct {
  char * Name;
  ppUns32 Flag;
  char * Caption;
  TPDFFont Font;
  TPDFBorder Border;
  ppUns32 MaxLen;
  TPDFAcroQuadding Align;
  PDFPaintContent PaintContent;
} TPDFEditBox, * PPDFEditBox;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Name;`	Name(text string) of Acroform object, Name (text string) of Acroform field is used to export when the PDF document is submitted.
`ppUns32 Flag;`	Specify the behavior of the annotation when is printed, rotated etc. AcroField Type (see TPDFAcroFlags)
`char * Caption;`	Default text string for appearance in EditBox. Text is displayed in EditBox when control is created.
`TPDFFont Font;`	Text font for appearance text label. Attributes of text written on or in the control.
`TPDFBorder Border;`	EditBox rectangle specifies position of the annotation on the page. Border width and colors.
`ppUns32 MaxLen;`	The maximum length of the field's text, in characters.
`TPDFAcroQuadding Align;`	Text alignment in edit box, justification of input text.
`PDFPaintContent PaintContent;`	Pointer to overload function to repaint edit box, instead of default (optional)

Description

Determination of Variable Text object for setting on Acroform Content. For text entering from document.

17.112 TPDFEncodingType Type

```
typedef enum _t_TPDFEncodingType {
  etPDFDocEncoding,
  etWinAnsiEncoding,
  etMacRomanEncoding,
  etStandardEncoding
} TPDFEncodingType;
```

File

VSFontA.h

Members

Members	Description
`etPDFDocEncoding`	PDF Document encoding
`etWinAnsiEncoding`	ANSI windows encoding
`etMacRomanEncoding`	Apple encoding
`etStandardEncoding`	Standard encoding

Description

Font encoding

17.113 TPDFFont Type

```
typedef struct {
  TPDFFontID ID;
  ppReal Size;
  TPDFColor Color;
} TPDFFont, * PPDFFont;
```

File

VSAcroObjects.h

Members

Members	Description
`TPDFFontID ID;`	Identifier of font type, see TPDFFontID (⧉ see page 228)
`ppReal Size;`	Size of font in points
`TPDFColor Color;`	Color of font for displaying, see TPDFColor (⧉ see page 225)

Description

Font Type. Specifies text font properties of the annotation on the page. (Acroform Object Characteristic)

17.114 TPDFFontID Type

```
typedef struct {
  ppBool IsStdFont;
  union {
    ppInt32 Index;
    TPDFStdandardFont StandardFont;
  } From;
} TPDFFontID, * PPDFFontID;
```

File

VSAcroObjects.h

Members

Members	Description
`ppBool IsStdFont;`	Boolean flag an accessories to 14 standard fonts

`union {`
`ppInt32 Index;`
`TPDFStdandardFont StandardFont;`
`} From;`
	Font source

`ppInt32 Index;`	Index of loaded font in Document
`TPDFStdandardFont StandardFont;`	If standard font then it must be named, see TPDFStdandardFont (⧉ see page 237)

Description

Font Index Structure - to support early PDF versions (1.2 and below)

17.115 TPDFHorJust Type

```
typedef enum {
  hjLeft,
  hjCenter,
  hjRight
} TPDFHorJust;
```

File

VSCanvasA.h

17.116 TPDFImageCompression Type

```
typedef enum _t_TPDFImageCompression {
  pdfiCCITT,
  pdfiJbig2,
  pdfiFlate
} TPDFImageCompression;
```

File

VSImageA.h

Members

Members	Description
`pdfiCCITT`	CCITT compression
`pdfiJbig2`	JBIG2 compression
`pdfiFlate`	FLATE compression

Description

Available black and white compressions

17.117 TPDFInformation Type

```
typedef enum _t_TPDFInformation {
  piCreator = 0,
  piAuthor,
  piDate,
  piProducer,
  piTitle,
  piSubject,
  piKeyWords,
  piModificationData
} TPDFInformation;
```

File

VSDocA.h

Members

Members	Description
`piCreator = 0`	Information about creator of the PDF Document
`piAuthor`	Information about author of the PDF Document
`piDate`	Information about date of the creation PDF Document
`piProducer`	Information about producer of the PDF Document
`piTitle`	Information about title of the PDF Document
`piSubject`	Information about subject of the PDF Document
`piKeyWords`	Information about keywords
`piModificationData`	Information about modification data

17.118 TPDFItemsBox Type

```
typedef struct {
  char * Name;
  ppUns32 Flag;
  char * Caption;
  TPDFFont Font;
  TPDFBorder Border;
  PDFPaintContent PaintContent;
} TPDFComboBox, * PPDFComboBox, TPDFListBox, * PPDFListBox, TPDFItemsBox;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Name;`	Name(text string) of Acroform object, Name (text string) of Acroform field used for export when the PDF document submitted.
`ppUns32 Flag;`	Specify the behavior of the annotation when is printed, rotated etc. AcroField Type (see TPDFAcroFlags)
`char * Caption;`	Default text string for appearance in ComboBox. Wasted in ListBox. Text is displayed in ComboBox when control is created.
`TPDFFont Font;`	Text font for appearance text label. Attributes of text written on or in the control.
`TPDFBorder Border;`	ComboBox or ListBox rectangle specifies position of the annotation on the page. Border width and colors.
`PDFPaintContent PaintContent;`	Pointer to overload function to repaint ComboBox (optional), instead of default.

Description

Determination of ComboBox object and ListBox object for setting on Acroform Content. Item(s) selection from items list.

17.119 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.

17.120 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.

17.121 TPDFListBox Type

```
typedef struct {
  char * Name;
  ppUns32 Flag;
  char * Caption;
  TPDFFont Font;
  TPDFBorder Border;
  PDFPaintContent PaintContent;
} TPDFComboBox, * PPDFComboBox, TPDFListBox, * PPDFListBox, TPDFItemsBox;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Name;`	Name(text string) of Acroform object, Name (text string) of Acroform field used for export when the PDF document submitted.
`ppUns32 Flag;`	Specify the behavior of the annotation when is printed, rotated etc. AcroField Type (see TPDFAcroFlags)
`char * Caption;`	Default text string for appearance in ComboBox. Wasted in ListBox. Text is displayed in ComboBox when control is created.
`TPDFFont Font;`	Text font for appearance text label. Attributes of text written on or in the control.
`TPDFBorder Border;`	ComboBox or ListBox rectangle specifies position of the annotation on the page. Border width and colors.
`PDFPaintContent PaintContent;`	Pointer to overload function to repaint ComboBox (optional), instead of default.

Description

Determination of ComboBox object and ListBox object for setting on Acroform Content. Item(s) selection from items list.

17.122 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

17.123 TPDFPageOrientation Type

```
typedef enum _t_TPDFPageOrientation {
  poPagePortrait = 0,
  poPageLandScape
} TPDFPageOrientation;
```

File

VSDocA.h

Members

Members	Description
`poPagePortrait = 0`	Orientation of Page is Portrait
`poPageLandScape`	Orientation of Page is Landscape

Description

Page Orientation Type

17.124 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.

17.125 TPDFPageSize Type

```
typedef enum _t_TPDFPageSize {
  psLetter = 0,
  psA4,
  psA3,
  psLegal,
  psB5,
  psC5,
  ps8x11,
  psB4,
  psA5,
  psFolio,
  psExecutive,
  psEnvB4,
  psEnvB5,
  psEnvC6,
  psEnvDL,
  psEnvMonarch,
  psEnv9,
  psEnv10,
  psEnv11
} TPDFPageSize;
```

File

VSDocA.h

Members

Members	Description
`psLetter = 0`	Document's Page Size is 792 x 612
`psA4`	Document's Page Size is 842 x 595
`psA3`	Document's Page Size is 1190 x 842
`psLegal`	Document's Page Size is 1008 x 612

`psB5`	Document's Page Size is 728 x 516
`psC5`	Document's Page Size is 649 x 459
`ps8x11`	Document's Page Size is 792 x 595
`psB4`	Document's Page Size is 1031 x 728
`psA5`	Document's Page Size is 595 x 419
`psFolio`	Document's Page Size is 936 x 612
`psExecutive`	Document's Page Size is 756 x 522
`psEnvB4`	Document's Page Size is 1031 x 728
`psEnvB5`	Document's Page Size is 708 x 499
`psEnvC6`	Document's Page Size is 459 x 323
`psEnvDL`	Document's Page Size is 623 x 312
`psEnvMonarch`	Document's Page Size is 540 x 279
`psEnv9`	Document's Page Size is 639 x 279
`psEnv10`	Document's Page Size is 684 x 297
`psEnv11`	Document's Page Size is 747 x 324

Description

Type of usual PDF Document's Page Sizes

17.126 TPDFProtectionType Type

```
typedef enum _t_TPDFProtectionType {
  pt40BitProtection = 0,
  pt128BitProtection = 1
} TPDFProtectionType;
```

File

VSDocA.h

Members

Members	Description
`pt40BitProtection = 0`	40 Bit protection key length
`pt128BitProtection = 1`	128 Bit protection key length

Description

Protection Key-Length Type of Crypted PDF Document

17.127 TPDFPushButton Type

```
typedef struct {
  char * Name;
  ppUns32 Flag;
  char * Caption;
  TPDFFont Font;
  TPDFBorder Border;
  ppReal Miter;
  PDFPaintContent PaintContentUp;
  PDFPaintContent PaintContentDown;
} TPDFPushButton, * PPDFPushButton;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Name;`	Name(text string) of Acroform object, Name (text string) of Acroform field is used to export when the PDF document is submitted.
`ppUns32 Flag;`	Specify the behavior of the annotation when is printed, rotated etc. AcroField Type (see TPDFAcroFlags)
`char * Caption;`	Specifies a text string that identifies the control to the user. Text label for appearance on button.
`TPDFFont Font;`	Text font for appearance text label. Attributes of text written on or in the control.
`TPDFBorder Border;`	Pushbutton rectangle specifies position of the annotation on the page. Border width and colors.
`ppReal Miter;`	Miter of pushbutton, bevel size.
`PDFPaintContent PaintContentUp;`	Pointer to overload function to repaint pushbutton in normal state (optional), instead of default appearance.
`PDFPaintContent PaintContentDown;`	Pointer to overload function to repaint pushbutton in pressed state (optional), instead of default appearance

Description

Determination of Pushbutton object for setting on Acroform Content. Action selection. Submit action.

17.128 TPDFRadioButton Type

```
typedef struct {
  char * Name;
  ppUns32 Flag;
  TPDFFont Font;
  TPDFBorder Border;
  TPDFCheckBoxStyle Style;
  TPDFCheckBoxSign Sign;
  ppBool Value;
  PDFPaintContent PaintContentOn;
  PDFPaintContent PaintContentOff;
} TPDFCheckBox, * PPDFCheckBox, TPDFRadioButton, * PPDFRadioButton;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Name;`	Name(text string) of Acroform object, Name (text string) of Acroform field is used to export when the PDF document submitted.
`ppUns32 Flag;`	Specify the behavior of the annotation when is printed, rotated etc. AcroField Type (see TPDFAcroFlags)
`TPDFFont Font;`	Text font for appearance text label. Attributes of text written on or in the control. Use only Font's color for displaying Mark character.
`TPDFBorder Border;`	CheckBox or RadioButton rectangle specifies position of the annotation on the page. Border width and colors
`TPDFCheckBoxStyle Style;`	Style of CheckBox or RadioButton - rectangle or circle (see TPDFCheckBoxStyle (⧉ see page 224)).
`TPDFCheckBoxSign Sign;`	Code of Mark character in CheckBox or RadioButton (see TPDFCheckBoxSign).
`ppBool Value;`	Value of CheckBox or RadioButton. Variable interactive value on Acroform (see VSAcroForm.h).
`PDFPaintContent PaintContentOn;`	Pointer to overload function to repaint CheckBox in checked state (optional), instead of default appearance.
`PDFPaintContent PaintContentOff;`	Pointer to overload function to repaint CheckBox in unchecked state (optional), instead of default appearance.

Description

Determination of CheckBox object and RadioButton object for setting on Acroform Content. Item selection.

17.129 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

17.130 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

17.131 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

17.132 TPDFSignature Type

```
typedef struct {
  char * Name;
  TPDFBorder Border;
  TPDFAcroSigFlags SigFlags;
  TPDFAnnotFlags AnnotFlag;
} TPDFSignature, * PPDFSignature;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Name;`	Name(text string) of Acroform object, Name (text string) of Acroform field is used to export when the PDF document is submitted.
`TPDFBorder Border;`	Signature specifies position of the annotation on the page.
`TPDFAcroSigFlags SigFlags;`	A set of flags specifying various document-level characteristics related to signature fields. See TPDFAcroSigFlags
`TPDFAnnotFlags AnnotFlag;`	Specify the behavior of the annotation when is printed, rotated etc. See TPDFAnnotFlags

Description

Determination of Empty Signature object for setting on Acroform Content. Item to sign document

17.133 TPDFStdandardFont Type

```
typedef enum _t_TPDFStdandardFont {
  stdfHelvetica,
  stdfHelveticaBold,
  stdfHelveticaOblique,
  stdfHelveticaBoldOblique,
  stdfTimesRoman,
  stdfTimesBold,
  stdfTimesItalic,
  stdfTimesBoldItalic,
  stdfCourier,
  stdfCourierBold,
  stdfCourierOblique,
  stdfCourierBoldOblique,
  stdfSymbol,
  stdfZapfDingbats
} TPDFStdandardFont;
```

File

VSFontA.h

Members

Members	Description
`stdfHelvetica`	Helvetica font
`stdfHelveticaBold`	Helvetica Bold font
`stdfHelveticaOblique`	Helvetica Oblique font
`stdfHelveticaBoldOblique`	Helvetica Bold Obliquefont
`stdfTimesRoman`	Times Roman font
`stdfTimesBold`	Times Bold font
`stdfTimesItalic`	Times Italic font
`stdfTimesBoldItalic`	Times Bold Italicfont
`stdfCourier`	Courier font
`stdfCourierBold`	Courier Bold font
`stdfCourierOblique`	Courier Oblique font
`stdfCourierBoldOblique`	Courier BoldOblique font
`stdfSymbol`	Symbol font
`stdfZapfDingbats`	Zapf Dingbats font

Description

Standard 14 fonts enum

17.134 TPDFTextBox Type

```
typedef struct {
  char * Caption;
  TPDFFont Font;
  TPDFBorder Border;
  TPDFAcroQuadding Align;
  ppReal Orientation;
} TPDFTextBox, * PPDFTextBox;
```

File

VSAcroObjects.h

Members

Members	Description
`char * Caption;`	Text string of label
`TPDFFont Font;`	Font for displaying text, see TPDFFont (⧉ see page 228)
`TPDFBorder Border;`	Border of text label, see TPDFBorder (⧉ see page 223)
`TPDFAcroQuadding Align;`	Alignment text label option, see TPDFAcroQuadding (⧉ see page 221)
`ppReal Orientation;`	Incline level, angle in degrees

Description

Determination of Text object for setting on Page or Acroform Content

17.135 TPDFVersion Type

```
typedef enum PDFVersion {
  pdfver10 = 0,
  pdfver11 = 1,
  pdfver12 = 2,
  pdfver13 = 3,
  pdfver14 = 4,
  pdfver15 = 5,
  pdfver16 = 6
} 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

Description

Type of supported PDF Document Versions

17.136 TPDFVertJust Type

```
typedef enum {
  vjTop,
  vjCenter,
  vjBottom
} TPDFVertJust;
```

File

VSCanvasA.h

17.137 TPolyAnnotDict Type

```
typedef struct {
  PDFActionHandle Action;
  PDFActionHandle AdditAction;
  annFlag AnFlags;
  PBSDict BSDict;
  PDFCosHandle BorderEffect;
  TDeviceRGB Color;
  int * Vertices;
  int VertLength;
  float * InteriorColor;
  char * Contents;
  ppInt32 ContLength;
  int LineEnding[2];
  char * DateTime;
  TAnotName IconName;
  TPolyType Type;
  ppBool Open;
  PDFCosHandle Popup;
  TPageRect Rectangle;
  char * TitleText;
  ppInt32 TTLength;
  float Transparency;
} * PPolyAnnotDict, TPolyAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`PDFActionHandle Action;`	(Optional) An action to be performed when the annotation is activated
`PDFActionHandle AdditAction;`	(Optional) An additional-actions dictionary defining the annotation’s behavior in response to various trigger events
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`PBSDict BSDict;`	(Optional) A border style dictionary specifying the characteristics of the annotation’s border
`PDFCosHandle BorderEffect;`	(Optional) A border effect dictionary describing an effect applied to the border described by the BS entry
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation’s icon when closed • The title bar of the annotation’s pop-up window • The border of a link annotation
`int * Vertices;`	(Required) An array of numbers representing the alternating horizontal and vertical coordinates, respectively, of each vertex, in default user space.
`int VertLength;`	Length of the vertices array
`float * InteriorColor;`	(Optional) An array of three numbers in the range 0.0 to 1.0 specifying the components, in the DeviceRGB color space, of the interior color with which to fill the annotation’s line endings (see Table 8.19). If this entry is absent, the interiors of the line endings are left transparent.
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation’s contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`int LineEnding[2];`	(Optional; PDF 1.4) An array of two names specifying the line ending styles to be used in drawing the line. The first and second elements of the array specify the line ending styles for the endpoints defined, respectively, by the first and second pairs of coordinates, (x1, y1) and (x2, y2), in the L array.
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`TPolyType Type;`	(Required) The type of annotation that this dictionary describes; must be ptPolygon or ptPolyline for a polygon or polyline annotation, respectively.
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.
`PDFCosHandle Popup;`	(Optional) An indirect reference to a pop-up annotation for entering or editing the text associated with this annotation.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`char * TitleText;`	(Optional) The text label to be displayed in the title bar of the annotation’s pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
`ppInt32 TTLength;`	Length of title
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.

Description

Poly annotation structure

17.138 TPolyType Type

```
typedef enum {
  ptPolygon,
  ptPolyline
} TPolyType;
```

File

VSAnnotA.h

Members

Members	Description
`ptPolygon`	The polygon annotation
`ptPolyline`	The polyline annotation

Description

Available types of poly annotation

17.139 TPopupAnnotDict Type

```
typedef struct {
  PDFCosHandle Popup;
  PDFActionHandle Action;
  PDFActionHandle AdditAction;
  annFlag AnFlags;
  PBSDict BSDict;
  TDeviceRGB Color;
  char * Contents;
  ppInt32 ContLength;
  char * DateTime;
  TAnotName IconName;
  ppBool Open;
  TPageRect Rectangle;
  char * TitleText;
  ppInt32 TTLength;
  float Transparency;
  PDFCosHandle Parent;
} * PPopupAnnotDict, TPopupAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`PDFCosHandle Popup;`	(Optional) An indirect reference to a pop-up annotation for entering or editing the text associated with this annotation.
`PDFActionHandle Action;`	(Optional) An action to be performed when the annotation is activated
`PDFActionHandle AdditAction;`	(Optional) An additional-actions dictionary defining the annotation's behavior in response to various trigger events
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`PBSDict BSDict;`	(Optional) A border style dictionary specifying the characteristics of the annotation's border
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation's icon when closed • The title bar of the annotation's pop-up window • The border of a link annotation
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation's contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.

`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`char * TitleText;`	(Optional) The text label to be displayed in the title bar of the annotation's pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
`ppInt32 TTLength;`	Length of title
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.
`PDFCosHandle Parent;`	(Optional) The parent annotation with which this pop-up annotation is associated.

Description

Popup annotation structure

17.140 TRubberStampAnnotDict Type

```
typedef struct {
  PDFCosHandle AppearanceStream;
  PDFCosHandle Popup;
  PDFActionHandle Action;
  PDFActionHandle AdditAction;
  annFlag AnFlags;
  PBSDict BSDict;
  TDeviceRGB Color;
  char * Contents;
  ppInt32 ContLength;
  char * DateTime;
  TAnotName IconName;
  ppBool Open;
  TPageRect Rectangle;
  char * TitleText;
  ppInt32 TTLength;
  float Transparency;
  TIconName IconStyleName;
} * PRubberStampAnnotDict, TRubberStampAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`PDFCosHandle Popup;`	(Optional) An indirect reference to a pop-up annotation for entering or editing the text associated with this annotation.
`PDFActionHandle Action;`	(Optional) An action to be performed when the annotation is activated
`PDFActionHandle AdditAction;`	(Optional) An additional-actions dictionary defining the annotation's behavior in response to various trigger events
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`PBSDict BSDict;`	(Optional) A border style dictionary specifying the characteristics of the annotation's border
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation's icon when closed • The title bar of the annotation's pop-up window • The border of a link annotation
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation's contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`char * DateTime;`	(Optional) The date and time when the annotation was created.

`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`char * TitleText;`	(Optional) The text label to be displayed in the title bar of the annotation's pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
`ppInt32 TTLength;`	Length of title
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.
`TIconName IconStyleName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: inApproved, inAsIs, inConfidential, inDepartmental, inDraft, inExperimental, inExpired, inFinal, inForComment, inForPublicRelease, inNotApproved, inNotForPublicRelease, inSold, inTopSecret Additional names may be supported as well.

Description

Rubber stamp annotation structure

17.141 TSCAnnotDict Type

```
typedef struct {
  ppBool Open;
  TPageRect Rectangle;
  THighlighMode AnnotHighLight;
  PDFActionHandle PageAction;
  float * InteriorColor;
  PDFActionHandle Action;
  char * Contents;
  ppInt32 ContLength;
  PBSDict BSDict;
  TAnotName IconName;
  TDeviceRGB Color;
  char * DateTime;
  annFlag AnFlags;
  char * TitleText;
  ppInt32 TTLength;
  float Transparency;
  PDFCosHandle Popup;
  PDFActionHandle AdditAction;
} * PSCAnnotDict, TSCAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units

`THighlighMode AnnotHighLight;`	(Optional) The annotation’s highlighting mode, the visual effect to be used when the mouse button is pressed or held down inside its active area: hmInvert - (Invert) Invert the contents of the annotation rectangle. hmNone - (None) No highlighting. hmOutline - (Outline) Invert the annotation’s border. hmPush - (Push) Display the annotation’s down appearance, if any. If no down appearance is defined, offset the contents of the annotation rectangle to appear as if it were being “pushed” below the surface of the page.
`PDFActionHandle PageAction;`	(Optional) A URI action formerly associated with this annotation. When Web Capture changes an annotation from a URI to a go-to action, it uses this entry to save the data from the original URI action so that it can be changed back in case the target page for the go-to action is subsequently deleted.
`float * InteriorColor;`	(Optional) An array of three numbers in the range 0.0 to 1.0 specifying the components, in the DeviceRGB color space, of the interior color with which to fill the annotation’s line endings (see Table 8.19). If this entry is absent, the interiors of the line endings are left transparent.
`PDFActionHandle Action;`	(Optional) An action to be performed when the annotation is activated
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation’s contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`PBSDict BSDict;`	(Optional) A border style dictionary specifying the characteristics of the annotation’s border
`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation’s icon when closed • The title bar of the annotation’s pop-up window • The border of a link annotation
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`char * TitleText;`	(Optional) The text label to be displayed in the title bar of the annotation’s pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
`ppInt32 TTLength;`	Length of title
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.
`PDFCosHandle Popup;`	(Optional) An indirect reference to a pop-up annotation for entering or editing the text associated with this annotation.
`PDFActionHandle AdditAction;`	(Optional) An additional-actions dictionary defining the annotation’s behavior in response to various trigger events

Description

Square and circle annotation structure

17.142 TSCType Type

```
typedef enum {
  sctSquare,
  sctCircle
} TSCType;
```

File

VSAnnotA.h

Members

Members	Description
`sctSquare`	Square annotation
`sctCircle`	Circle annotation

Description

Available types of SC annotation

17.143 TSEFormat Type

```
typedef enum {
  sefRaw = 0,
  sefSigned,
  sefmuLaw,
  sefALaw
} TSEFormat;
```

File

VSAnnotA.h

Members

Members	Description
`sefRaw = 0`	Unspecified or unsigned values in the range 0 to 2B ? 1
`sefSigned`	Twos-complement values
`sefmuLaw`	μ-law–encoded samples
`sefALaw`	A-law–encoded samples

Description

Available types of the PDF sound encoding format

17.144 TSoundAnnotDict Type

```
typedef struct {
  annFlag AnFlags;
  TDeviceRGB Color;
  char * Contents;
  ppInt32 ContLength;
  char * DateTime;
  TAnotName IconName;
  ppBool Open;
  TPageRect Rectangle;
  char * Filename;
  ppInt32 FNLength;
  float Transparency;
} * PSoundAnnotDict, TSoundAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation's icon when closed • The title bar of the annotation's pop-up window • The border of a link annotation
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation's contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents

`char * DateTime;`	(Optional) The date and time when the annotation was created.
`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`char * Filename;`	(Required) Sound filename.
`ppInt32 FNLength;`	Filename length
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.

Description

Sound annotation structure

17.145 TSoundDict Type

```
typedef struct {
  int SamplingRate;
  int Channels;
  int BitsPerSample;
  TSEFormat EncFormat;
} * PSoundDict, TSoundDict;
```

File

VSAnnotA.h

Members

Members	Description
`int SamplingRate;`	Sound dictionary sampling rate
`int Channels;`	Sound dictionary channels
`int BitsPerSample;`	Sound dictionary bits per sample
`TSEFormat EncFormat;`	Sound dictionary encoding format structure

Description

Sound dictionary structure

17.146 TTextAnnotDict Type

```
typedef struct {
  PDFCosHandle Popup;
  PDFActionHandle Action;
  PDFActionHandle AdditAction;
  annFlag AnFlags;
  PBSDict BSDict;
  TDeviceRGB Color;
  char * Contents;
  ppInt32 ContLength;
  char * DateTime;
  TAnotName IconName;
  ppBool Open;
  TPageRect Rectangle;
  char * TitleText;
```

```
    ppInt32 TTLength;
    float Transparency;
  } TTextAnnotDict;
```

File

VSAnnotA.h

Members

Members	Description
`PDFCosHandle Popup;`	(Optional) An indirect reference to a pop-up annotation for entering or editing the text associated with this annotation.
`PDFActionHandle Action;`	(Optional) An action to be performed when the annotation is activated
`PDFActionHandle AdditAction;`	(Optional) An additional-actions dictionary defining the annotation’s behavior in response to various trigger events
`annFlag AnFlags;`	(Optional) A set of flags specifying various characteristics of the annotation.
`PBSDict BSDict;`	(Optional) A border style dictionary specifying the characteristics of the annotation’s border
`TDeviceRGB Color;`	(Optional) An array of three numbers in the range 0.0 to 1.0, representing the components of a color in the DeviceRGB color space. This color will be used for the following purposes: • The background of the annotation’s icon when closed • The title bar of the annotation’s pop-up window • The border of a link annotation
`char * Contents;`	(Required) Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation’s contents in human-readable form.
`ppInt32 ContLength;`	Count of characters in contents
`char * DateTime;`	(Optional) The date and time when the annotation was created.
`TAnotName IconName;`	(Optional) The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: anDefault anComment anHelp anInsert anKey anNewParagraph anParagraph
`ppBool Open;`	(Optional) A flag specifying whether the annotation should initially be displayed open.
`TPageRect Rectangle;`	(Required) The annotation rectangle, defining the location of the annotation in default user space units
`char * TitleText;`	(Optional) The text label to be displayed in the title bar of the annotation’s pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
`ppInt32 TTLength;`	Length of title
`float Transparency;`	(Optional) The constant opacity value to be used in painting the annotation This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened.

Description

Text annotation structure

Index