



Acrobat Core API Reference

Technical Note #5191

Version: Acrobat 6.0



ADOBE SYSTEMS INCORPORATED

Corporate Headquarters

345 Park Avenue

San Jose, CA 95110-2704

(408) 536-6000

<http://partners.adobe.com>

March 2004



Copyright 2004 Adobe Systems Incorporated. All rights reserved.

NOTICE: All information contained herein is the property of Adobe Systems Incorporated. No part of this publication (whether in hardcopy or electronic form) may be reproduced or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the Adobe Systems Incorporated.

PostScript is a registered trademark of Adobe Systems Incorporated. All instances of the name PostScript in the text are references to the PostScript language as defined by Adobe Systems Incorporated unless otherwise stated. The name PostScript also is used as a product trademark for Adobe Systems' implementation of the PostScript language interpreter.

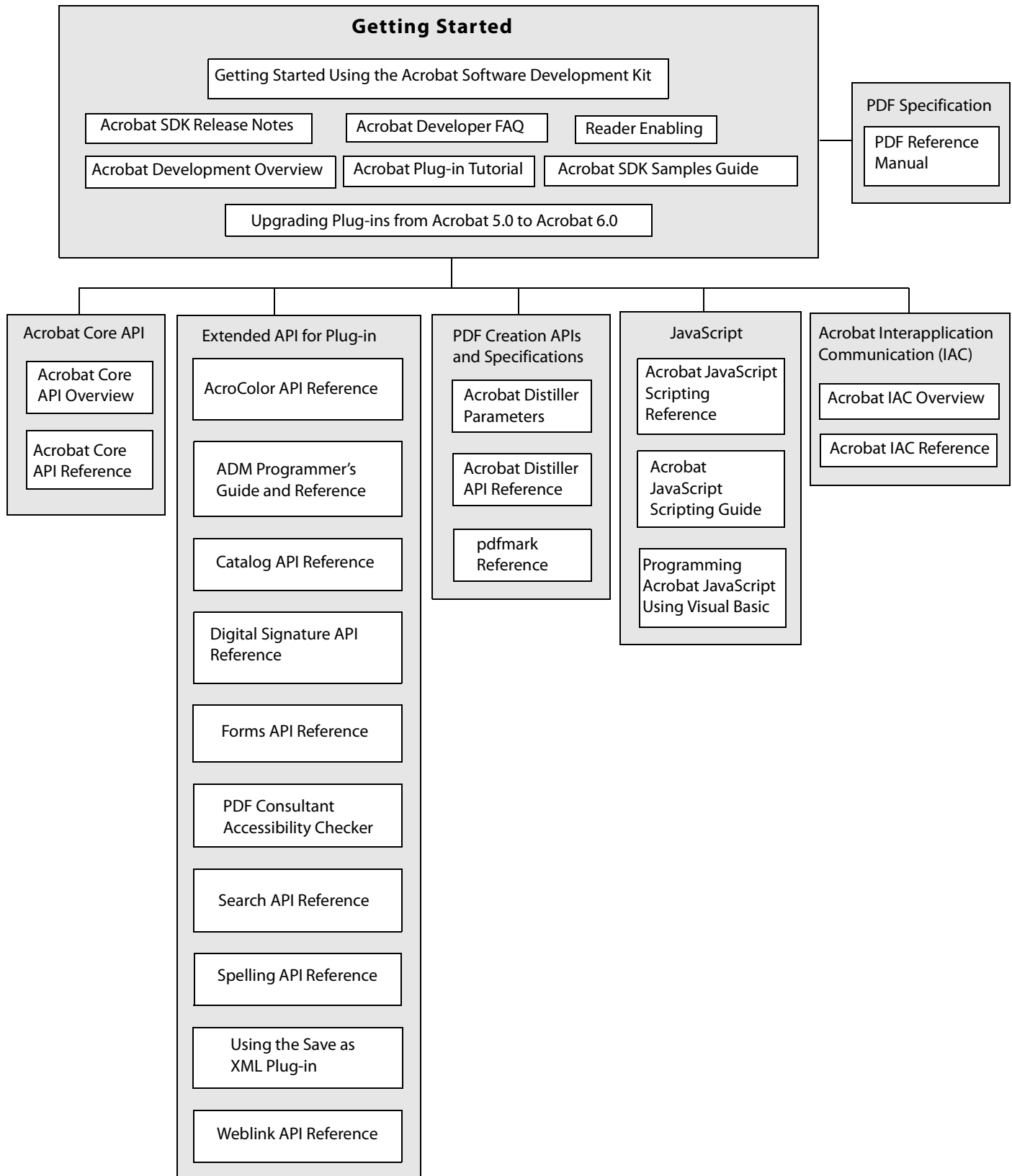
Except as otherwise stated, any reference to a "PostScript printing device," "PostScript display device," or similar item refers to a printing device, display device or item (respectively) that contains PostScript technology created or licensed by Adobe Systems Incorporated and not to devices or items that purport to be merely compatible with the PostScript language.

Adobe, the Adobe logo, Acrobat, the Acrobat logo, Acrobat Capture, Distiller, PostScript, the PostScript logo and Reader are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Apple, Macintosh, and Power Macintosh are trademarks of Apple Computer, Inc., registered in the United States and other countries. PowerPC is a registered trademark of IBM Corporation in the United States. ActiveX, Microsoft, Windows, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. UNIX is a registered trademark of The Open Group. All other trademarks are the property of their respective owners.

This publication and the information herein is furnished AS IS, is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies, makes no warranty of any kind (express, implied, or statutory) with respect to this publication, and expressly disclaims any and all warranties of merchantability, fitness for particular purposes, and noninfringement of third party rights.

Acrobat SDK Documentation Roadmap





How to Use the Core API Reference

Description

This Core API technical note is intended for experienced users of the Adobe Acrobat Software Development Kit (SDK). It describes in detail the objects and methods provided by the Acrobat Application Program Interface (API).

In addition to having a working knowledge of ANSI C programming, it is useful to have a good understanding of object-oriented programming concepts, such as the use of methods, classes, objects, and member functions (callbacks).

If you received this technical note without obtaining the entire Acrobat Software Development Kit (SDK), you can get the complete SDK by visiting:

<http://partners.adobe.com/asn/developer/acrosdk/main.html>

Contents

This technical note contains the following sections:

- **Objects.** Descriptions of the objects, list of the methods that obtain and dispose of the objects, the functions used to access them, Cos conversion, and validity testing.
- **Methods.** Detailed description of each method, including its parameters and return value. Method descriptions are divided into the following layers or models:
 - **AS Layer Methods:** The Acrobat Support (AS) layer provides a variety of utility methods, including platform-independent memory allocation and fixed-point math utilities.
 - **AV Layer Methods:** The Acrobat Viewer (AV) layer deals with the viewer's user interface.
 - **Cos Layer Methods:** The Cos Object System (Cos) layer provides access to the low-level building blocks of PDF files.
 - **PD Layer Methods:** The Portable Document (PD) layer provides access to components of PDF documents.
 - **PDFEdit Methods:** The PDFEdit layer provides access to PDF page contents associated with the **Contents** entry in the page's dictionary.
 - **PDSEdit Methods:** The PDSEdit layer provides access to the logical structure of a PDF document.
 - OS-specific functions are documented in the sections [Macintosh-specific Methods](#), [UNIX-specific Methods](#), and [Windows-specific Methods](#).

Within layers, methods are organized by object. Within each object, methods are listed alphabetically.

NOTE: Method descriptions list exceptions that can be raised by each method. However, this list is not necessarily complete, and in general, any method can raise an exception.

- [Callbacks](#). Detailed description of callback functions, in which the Acrobat viewer and Library call back clients and applications.
- [Declarations](#). Detailed descriptions of data structures used by the Acrobat viewer and Library.
- [Lists](#). Lists of useful information, such as UI element names.
- [Notifications](#). Description of objects that allow a client or application to indicate an interest in a specified event, and provide a procedure that is called each time that event occurs.
- [Errors](#). List of error categories and error codes. These error codes typically represent exceptions raised by Acrobat.
- [Macros](#). Detailed description of each macro, which includes operators on fixed-point numbers, conversions between integers and fixed-point numbers, conversions between C strings and fixed-point numbers, math and rectangle utilities, and matrix operations.
- [API Changes](#). Descriptions of new objects, methods, callbacks, and so forth, that have been added or changed extensively.

Other Useful Documentation

You should be familiar with the Acrobat core API and Portable Document Format (PDF). The following technical notes provide this information.

[Getting Started Using the Adobe Acrobat Software Development Kit](#) introduces the SDK, clients, Interapplication Communication (IAC), libraries, toolkits, and Host Function Tables (HFTs). In addition to SDK installation instructions, this manual has an extensive “road map” to other documentation.

[Acrobat Core API Overview](#), Technical Note #5190. Gives an overview of the objects and methods provided by the Acrobat core API.

[PDF Reference](#), Version 1.5. Provides a description of the PDF file format, as well as suggestions for producing efficient PDF files. It is intended for application developers who wish to produce PDF files directly.

The Acrobat SDK includes many other books that you might find useful. When mentioned in this document, those books will often appear as live links (blue italic links). However, in order to actually jump from this document to those books, those books must exist in the proper directories within your computer’s file system. This happens automatically when you install the SDK onto your system.

If for some reason you did not install the entire SDK onto your system and you do not have all of the documentation, please visit the Adobe Solutions Network web site (

<http://partners.adobe.com/asn>) to find the books you need. Then download them and install them in the proper directories, which can be determined by looking at the Acrobat SDK Documentation roadmap, included at the beginning of each book in the SDK.

Conventions Used in This Book

The Acrobat documentation uses text styles according to the following conventions.

Font	Used for	Examples
monospaced	Paths and filenames	<code>C:\templates\mytmpl.fm</code>
	Code examples set off from plain text	These are variable declarations: <code>AVMenu commandMenu, helpMenu;</code>
monospaced bold	Code items within plain text	The GetExtensionID method ...
	Parameter names and literal values in reference documents	The enumeration terminates if proc returns false .
monospaced italic	Pseudocode	<code>ACCB1 void ACCB2 ExeProc(void)</code> <code>{ do something }</code>
	Placeholders in code examples	<code>AFSimple_Calculate(cFunction, cFields)</code>
blue	Live links to Web pages	The Acrobat Solutions Network URL is: http://partners.adobe.com/asn/
	Live links to sections within this document	See Using the SDK .
	Live links to other Acrobat SDK documents	See the Acrobat Core API Overview .
	Live links to code items within this document	Test whether an ASAtom exists.
bold	PostScript language and PDF operators, keywords, dictionary key names	The setpagedevice operator
	User interface names	The File menu

Font	Used for	Examples
italic	Document titles that are not live links	<i>Acrobat Core API Overview</i>
	New terms	<i>User space</i> specifies coordinates for...
	PostScript variables	<i>filename</i> deletefile

Product Availability

For each method description, a Product section lists the Adobe products in which that method is available. The products are listed as follows:

reader	Adobe Reader 6.0
base	Adobe Acrobat 6.0 Standard
pro	Adobe Acrobat 6.0 Professional
pdf	Adobe PDF Library

Objects

AS Layer	ASAtom	ASFileSys	ASUUID
	ASCab	ASPathName	HFT
	ASCallback	ASPlatformPath	HFTServer
	ASDate	ASStm	MDFile
	ASExtension	ASText	
	ASFile	ASTimeSpan	
AV Layer	AVActionHandler	AVMenuBar	AVToolBar
	AVAnnotHandler	AVMenuItem	AVToolButton
	AVApp	AVPageView	AVUndo
	AVCommand	AVSweetPea	AVWindow
	AVDoc	AVSys	
	AVMenu	AVTool	
Cos Layer	CosArray	CosFixed	CosObj
	CosBoolean	CosInteger	CosObjCollection
	CosDict	CosName	CosStream
	CosDoc	CosNull	CosString
PD Layer	PDAction	PDLinkAnnot	PDTextAnnot
	PDAnnot	PDNameTree	PDTextSelect
	PDBead	PDNumTree	PDThread
	PDBookmark	PDOCConfig	PDThumb
	PDCharProc	PDOCContext	PDTrans
	PDDoc	PDOCG	PDViewDestination
	PDFileSpec	PDOCMD	PDWord
	PDFont	PDPage	PDWordFinder
	PDForm	PDPageLabel	PDXObject
	PDGraphic	PDPath	
	PDImage	PDStyle	
	PDInlineImage	PDText	

PDFEdit	PDEBeginContainer	PDEExtGState	PDEShading
	PDEBeginGroup	PDEFont	PDESoftMask
	PDEClip	PDEForm	PDEText
	PDEColorSpace	PDEGroup	PDEUnknown
	PDEContainer	PDEImage	PDEXGroup
	PDEContent	PDEObject	PDEXObject
	PDEDeviceNColors	PDEPath	PDSysEncoding
	PDEElement	PDEPattern	PDSysFont
	PDEEndContainer	PDEPlace	
	PDEEndGroup	PDEPS	
PDSEdit	PDSAttrObj		
	PDSClassMap		
	PDSElement		
	PDSMC		
	PDSRoleMap		
	PDSTreeRoot		



AS Layer

Acrobat Support layer. Platform-independent objects and utility functions used throughout the API.

ASAtom

A hashed token used in place of strings to optimize performance (it is much faster to compare **ASAtoms** than strings). Many methods return **ASAtoms**.

Obtaining

[ASAtomFromString](#)

Attributes

Get	Set
ASAtomGetString	ASAtomFromString

Validity testing

[ASAtomExistsForString](#)

ASCab

ASCab objects (“cabinets”) can be used to store arbitrary key-value pairs. The keys are always null-terminated strings containing only low-ASCII alphanumeric characters and spaces (ASCII character 32). Key names cannot begin or end with a space.

Every time you place a non-scalar value inside a cabinet you are handing that value to the **ASCab** implementation to manage—that is, putting a value in a cabinet is always a hand-off operation. For example, if you create an **ASText** object and add it as a value in an **ASCab**, the **ASText** object is no longer managed by you; it is managed by the **ASCab**. The **ASCab** will destroy the **ASText** object when its associated key is removed or the key’s value is over-written. Pointer values are a special case discussed in more detail below.

The routine naming convention is as follows:

- “Get” routines return a value. These objects are owned by the **ASCab** and should not be destroyed by the caller of “Get”.
- “GetCopy” routines make a copy of the data; the “GetCopy” client owns the resulting information and can modify it at will; he is also responsible for destroying it.
- “Detach” routines work the same way as “Get” routines but the key is removed from the **ASCab** without destroying the associated value that is passed back to the client of “Detach”. The client is responsible for destroying the returned object.

Normally pointers are treated the same way as scalars; the **ASCab** passes the pointer value back and forth but doesn’t manage the data to which it points. This all changes if the pointer has an associated “destroyProc”. If the “destroyProc” is set, the **ASCab** will reference count the pointer to track how many times the pointer is referenced from any **ASCab** (for example, the reference count will be bumped up whenever the pointer is copied via **ASCabCopy** or added to another **ASCab** via **ASCabPutPointer**) and will destroy the data associated with the pointer when the ref count goes to 0. The data is destroyed by calling the “destroyProc”. Detaching a pointer removes one reference to the pointer without ever destroying the information pointed to. **ASCabDetachPointer** returns a separate value indicating whether the pointer can safely be destroyed by the client or is still referred to by other key-value pairs inside any **ASCabs**—that is, whether the reference count went to zero when the pointer was detached from the **ASCab**.

Any of the **ASCab** API’s can take a compound name: a string consisting of multiple keys separated by the colon (:) character—for example, “Grandparent:Parent:Child:Key”. The implementation will burrow down through such a compound string until it reaches the most deeply nested cabinet. Also, any of the “Put” routines can take a **NULL** key name. If the key name is **NULL**, the routine creates a new, numeric key name. If the cabinet is empty, the first generated key name will be “0” and subsequent names will increase in ascending order. This is useful when treating an **ASCab** as a bag of unnamed items, or when adding an ordered list of items to an empty **ASCab**.

Obtaining:

ASCabNew

ASCabFromEntryList

[ASCabCopy](#)

Disposing:

[ASCabDestroy](#)

Enumeration:

[ASCabEnum](#)

[ASConstCabEnum](#)

Actions:

[ASCabEqual](#)

[ASCabValueEqual](#)

Attributes

Get	Set
ASCabGetAtom	ASCabPutAtom
ASCabGetBinary , ASCabGetBinaryCopy	ASCabPutBinary
ASCabGetBool	ASCabPutBool
ASCabGetCab	ASCabPutCab
ASCabGetDouble	ASCabPutDouble
ASCabGetInt	ASCabPutInt
ASCabGetPathNameCopy	ASCabPutPathName
ASCabGetPointer	ASCabPutPointer
ASCabGetPointerDestroyProc	—
ASCabGetPointerType	—
ASCabGetString , ASCabGetStringCopy	ASCabPutString
ASCabGetText	ASCabPutText
ASCabGetType	—
ASCabGetUns	ASCabPutUns
—	ASCabPutNull
ASCabKnown	—



Get	Set
—	ASCabMakeEmpty
—	ASCabMunge
—	ASCabRemove
ASCabReadFromStream	ASCabWriteToStream



ASCallback

Callbacks allow the Acrobat viewer or the Library to call functions in an application or plug-in.

Obtaining

- [ASCallbackCreate](#)
- [ASCallbackCreateNotification](#)
- [ASCallbackCreateProto](#)
- [ASCallbackCreateReplacement](#)

Disposing

- [ASCallbackDestroy](#)

ASDate

Date objects represent a particular date and time. All date objects are guaranteed to give accurate representation of UTC time (not adjusted for leap seconds, as the addition of leap seconds to the international calendar does not happen according to a well-defined rule).

Date objects are not guaranteed to represent local time accurately, because some operating systems cannot always determine the prevailing daylight saving rule for the time zone. However, the date object can be set to use the same rule as the operating system.

Obtaining

[ASDateNew](#)

[ASDateDup](#)

Disposing

[ASDateDestroy](#)

Attributes

Get	Set
ASDateClear	—
ASDateCopy	
ASDateGetLocalTime	ASDateSetToCurrentLocalTime
ASDateGetTimeString	ASDateSetTimeFromString
ASDateGetUTCTime	ASDateSetToCurrentUTCTime
—	ASDateSetLocalTimeOffset
—	ASDateSetTimeFromRec

Actions

[ASDateAddCalendarTimeSpan](#)

[ASDateAddTimeSpan](#)

[ASDateCalendarDiff](#)

[ASDateCompare](#)

[ASDateExactDiff](#)

[ASDateSubtractCalendarTimeSpan](#)

[ASDateSubtractTimeSpan](#)

Declarations

[ASCalendarTimeSpan](#)

[ASDateTimeFormat](#)

[ASTimeRec](#)



ASExtension

An opaque pointer to an object that identifies a specific loaded client. An unique **ASExtension** object is created for each client when it is loaded. If the client fails to initialize, the **ASExtension** remains but is marked as inactive.

Obtaining

[ASEnumExtensions](#)

Attributes

Get	Set
ASExtensionGetFileName	—
ASExtensionGetRegisteredName	—

ASFile

An opaque representation of an open file.

Obtaining

[PDDocGetFile](#)
[ASFileSysOpenFile](#)
[ASFileFromMDFile](#)

Attributes

Get	Set
ASFileAcquirePathName	—
ASFileGetEOF	ASFileSetEOF
ASFileGetFileSys	—
ASFileGetMDFile	—
ASFileGetOpenMode	ASFileSetMode
ASFileGetPos	ASFileSetPos
ASFileGetURL	—

Actions

[ASFileClose](#)
[ASFileFlush](#)
[ASFileHardFlush](#)
[ASFileIsSame](#)
[ASFilePushData](#)
[ASFileRead](#)
[ASFileReopen](#)
[ASFileWrite](#)
[ASFileRegisterFileSys](#)
[ASFileUnregisterFileSys](#)

Declarations

[ASErrorCode](#)
[ASFileMode](#)
[ASFile Flags](#)
[ASFileMode Flags](#)
[ASFileStatus Flags](#)

ASFileSys

A collection of functions that implement file system services, such as opening files, deleting files, reading data from a file, and writing data to a file. Each **ASFileSys** provides these services for one class of devices.

To create an **ASFileSys**, complete an **ASFileSysRec** structure.

Obtaining

[ASGetDefaultFileSys](#)
[ASFileGetFileSys](#)
[ASFileGetFileSysByName](#)
[PDFFileSpecGetFileSys](#)

Attributes

Get	Set
ASFileSysAcquireFileSysPath , ASFileSysAcquireParent , ASFileSysAcquirePlatformPath	—
ASFileSysCanPerformOpOnItem	—
ASFileSysCopyPath ASFileSysCreatePathName	—
ASFileSysDIPathFromPath ASFileSysPathFromDIPath	—
ASFileSysDisplayASTextFromPath , ASFileSysDisplayStringFromPath	—
ASFileSysGetItemProps ASFileSysGetItemPropsAsCab	ASFileSysConvertCabToItemProps ASFileSysConvertCabToItemProps ASFileSysConvertItemPropsToCab
ASFileSysGetNameFromPath , ASFileSysGetNameFromPathAsASText	—
ASFileSysGetStorageFreeSpace	—
ASFileSysGetTempPathName	—
ASFileSysGetTypeAndCreator	ASFileSysSetTypeAndCreator
ASFileSysURLFromPath	—



Get	Set
ASFileSysFirstFolderItem, ASFileSysNextFolderItem	ASFileSysCreateFolder ASFileSysDestroyFolderIterator ASFileSysRemoveFolder

Actions

ASFileRegisterFileSys
 ASFileSysFlushVolume
 ASFileSysOpenFile
 ASFileSysPerformOpOnItem
 ASFileSysReleasePath
 ASFileSysRemoveFile
 ASFileUnregisterFileSys

Declarations

ASFileStatus Flags
 ASFileSysRec



ASPlatformPath

Description

An opaque object used to retrieve a platform-specific path object.

Obtaining

[ASFileSysAcquirePlatformPath](#)

Disposing

[ASFileSysReleasePlatformPath](#)

Attributes:

Get	Set
ASPlatformPathGetCFURLRefRecPtr	—
ASPlatformPathGetCstringPtr	—
ASPlatformPathGetFSRefPtr	—
ASPlatformPathGetFSRefWithCFStringRefRecPtr	—
ASPlatformPathGetFSSpecPtr	—
ASPlatformPathGetPOSIXPathPtr	—

Declarations

[CFURLRefRec_Ptr](#)
[Cstring_Ptr](#)
[FSRef_Ptr](#)
[FSRefWithCFStringRefRec_Ptr](#)
[FSSpec_Ptr](#)
[POSIXPath_Ptr](#)



ASStm

A data stream that may be a buffer in memory, a file, or an arbitrary user-written procedure. Typically used to extract data from a PDF file. When writing or extracting data streams, the **ASStm** must be connected to a Cos stream.

Obtaining

- [ASFileStmRdOpen](#)
- [ASFileStmWrOpen](#)
- [ASMemStmRdOpen](#)
- [ASProcStmRdOpenEx](#)
- [ASProcStmWrOpen](#)
- [CosStreamOpenStm](#)

Disposing

- [ASStmClose](#)

Actions

- [ASStmRead](#)
- [ASStmWrite](#)

AStext

An **AStext** object represents a Unicode string. **AStext** objects can also be used to convert between Unicode and various platform-specific text encodings as well as conversions between various Unicode formats (for example, UTF-16, UTF-8). Since it is common for a Unicode string to be repeatedly converted to or from the same platform-specific text encoding, **AStext** objects are optimized for this operation—for example, they can cache both the Unicode and platform-specific text strings.

There are several ways of creating an **AStext** object depending on the type and format of the original text data. The following terminology is used throughout this API to describe the various text formats.

- **Encoded**—A multi-byte string terminated with a single 0 character and coupled with a specific host encoding indicator. In the Macintosh OS, the text encoding is specified using a script code. In the Windows OS, the text encoding is specified using a CHARSET code. On Unix the only valid host encoding indicator is 0, which specifies text in the platform's default Roman encoding. On all platforms Asian text is typically specified using multi-byte strings.
- **ScriptText**—A multi-byte string terminated with a single 0 character and coupled with an **ASScript** code. This is merely another way of specifying the Encoded case; the **ASScript** code is converted to a host encoding using **ASScriptToHostEncoding**.
- **Unicode**—Text specified using UTF-16 or UTF-8. In the UTF-16 case the bytes can be in either big-endian format or the endian-ness that matches the platform and are always terminated with a single **ASUns16** 0 value. In the UTF-8 case the text is always terminated with a trailing 0 byte. Unicode refers to straight Unicode without the **0xFE 0xFF** prefix or language and country codes that can be encoded inside a PDF document.
- **PDText**—A string of text pulled out of a PDF document. This will either be a big-endian Unicode string pre-appended with the bytes **0xFE 0xFF** or a string in **PDFDocEncoding**. In the Unicode case, this string may have embedded language and country identifiers. **AStext** objects strip language and country information out of the **PDText** string and track them separately. See below for more details.

AStexts can also be used to accomplish encoding and format conversions; you can request a string in any of the formats specified above.

In all cases the **AStext** code attempts to preserve all characters. For example, if you attempt to concatenate strings in separate host encodings the implementation may convert both to Unicode and perform the concatenation in Unicode space.

When creating a new **AStext** object, or putting new data into an existing object, the implementation will always copy the supplied data into the **AStext** object. The original data is yours to do with as you will (and release if necessary).

The size of **AStext** data is always specified in bytes—for example, the **len** argument to **AStextFromSizedUnicode** specifies the number of bytes in the string, not the number of Unicode characters.

Host encoding and Unicode strings are always terminated with a null character (which consists of one null byte for host-encoded strings and two null bytes for Unicode strings). You cannot create a string with an embedded NULL character even using the calls which take an explicit length parameter.

The “Getxxx” calls return pointers to data held by the **ASText** object. You cannot free or manipulate this data directly. The “GetxxxCopy” calls return data you can manipulate at will and that you're responsible for freeing.

An **ASText** object can have language and country codes associated with it. A language code is a 2-character ISO 639 language code. A country code is a 2-character ISO 3166 country code. In both cases the 2-character codes are packed into an **ASUns16** value—the first character in bits 8-15, and the second character in bits 0-7. These language and country codes can be encoded into a UTF-16 variant of **PDText** encoding using an escape sequence; see Section 3.8 in the [PDF Reference](#). The **ASText** calls will automatically parse the language and country codes embedded inside UTF-16 **PDText** and will also author appropriate escape sequences to embed the language and country codes (if present) when generating UTF-16 **PDText**.

Obtaining

- [ASTextNew](#)
- [ASTextCopy](#)
- [ASTextDup](#)
- [ASTextFromEncoded](#)
- [ASTextFromInt32](#)
- [ASTextFromPDText](#)
- [ASTextFromScriptText](#)
- [ASTextFromSizedEncoded](#)
- [ASTextFromSizedPDText](#)
- [ASTextFromSizedScriptText](#)
- [ASTextFromSizedUnicode](#)
- [ASTextFromUnicode](#)
- [ASTextFromUns32](#)

Disposing

- [ASTextDestroy](#)
- [ASTextMakeEmpty](#)

Actions

- [ASTextCat](#)
- [ASTextCatMany](#)
- [ASTextCmp](#)



Attributes

Get	Set
AS<code>Text</code>GetBestEncoding	—
AS<code>Text</code>GetBestScript	—
AS<code>Text</code>GetCountry	AS<code>Text</code>SetCountry
AS<code>Text</code>GetEncoded , AS<code>Text</code>GetEncodedCopy	AS<code>Text</code>SetEncoded
AS<code>Text</code>GetLanguage	AS<code>Text</code>SetLanguage
AS<code>Text</code>GetPDTextCopy	AS<code>Text</code>SetPDText
AS<code>Text</code>GetScriptText , AS<code>Text</code>GetScriptTextCopy	AS<code>Text</code>SetScriptText
AS<code>Text</code>GetUnicode , AS<code>Text</code>GetUnicodeCopy	AS<code>Text</code>SetUnicode
AS<code>Text</code>IsEmpty	—
—	AS<code>Text</code>NormalizeEndOfLine
—	AS<code>Text</code>SetSizedEncoded
—	AS<code>Text</code>SetSizedPDText
—	AS<code>Text</code>SetSizedScriptText
—	AS<code>Text</code>SetSizedUnicode
—	AS<code>Text</code>Replace , AS<code>Text</code>ReplaceASCII , AS<code>Text</code>ReplaceBadChars

ASpan

Represents an exact time span, measured in seconds. The internal representation uses 64-bit signed integers (to avoid the 2038 problem caused by 32-bit representation). Negative timespans are allowed.

Obtaining

[ASpanNew](#)
[ASpanDup](#)

Disposing

[ASpanDestroy](#)

Attributes

Get	Set
	ASpanCopy
ASpanGetASInt32	ASpanSetFromASInt32 ASpanSet ASpanSetFromString
	ASpanNegate
	ASCalendarTimeSpanAddWithBase ASpanAdd
	ASCalendarTimeSpanDiff ASpanDiff

Actions

[ASCalendarTimeSpanCompare](#)
[ASpanCompare](#)



ASUUID

A universal unique identifier for the current user or the current session.

Obtaining

- AVAppGetUUID
- ASUUIDFromCString
- ASUUIDGenFromHash
- ASUUIDGenFromName
- ASUUIDGenUnique

Actions

- ASUUIDToCString

Declarations

- AVAppUUIDType
- ASUUID
- ASUUIDMaxStringLen

HFT

A table of function pointers (actually callbacks) called a host function table. This is the mechanism through which clients call methods in the Acrobat viewer or in other clients.

Obtaining

[ASExtensionMgrGetHFT](#)
[HFTNew](#)
[HFTNewEx](#)

Disposing

[HFTDestroy](#)

Attributes

Get	Set
HFTGetReplacedEntry	HFTReplaceEntry , HFTReplaceEntryEx , HFTUnreplaceEntry
HFTGetVersion	

Validity testing

[HFTIsValid](#)

Declarations

[HFTData](#)
[HFTEntry](#)
[HFTEntryReplaceable](#)
[HFT Values](#)
[ASVersion](#)



HFTServer

Each **HFT** is serviced by an HFT server. The HFT server is responsible for handling requests to obtain or destroy its **HFT**.

Obtaining

[HFTServerNew](#)

Disposing

[HFTServerDestroy](#)



MDFFile

A file-system specific representation of an individual file. It uses the machine's native platform-specific data structure to represent a file. In the client API, it is primarily used by Replacement FileSystem implementors. A replacement file system can choose it's own implementation of an **MDFFile** that is mapped by the viewer to an **ASFile** for use by clients of the replacement file system. In Acrobat 5, **MDFFile** was rename **ASMDFFile**.

Obtaining

[ASFileGetMDFFile](#)

Attributes

Get	Set
ASFileFromMDFFile	—



AV Layer

Acrobat viewer layer. A set of objects whose methods allow clients to manipulate components of the Acrobat viewer application itself, such as menus and menu items.

AVActionHandler

Carries out an action. When the Acrobat viewer executes an action, it looks for the action handler with a **type** matching that of the action it is trying to execute. The Acrobat viewer invokes the matching handler to perform the action. If no match is found, the Acrobat viewer ignores the user action. Clients can add new action handlers by using [AVAppRegisterActionHandler](#).

Obtaining

[AVAppRegisterActionHandler](#)
[AVAppGetActionHandlerByType](#)

Enumerating

[AVAppEnumActionHandlers](#)

Attributes

Get	Set
AVActionHandlerGetProcs	—
AVActionHandlerGetType	—
AVActionHandlerGetUIName	—

Declarations

[AVActionHandlerProcs](#)

AVAnnotHandler

Responsible for creating, displaying, selecting, and deleting a particular type of annotation. There is one annotation handler for each annotation type. The Acrobat viewer contains two built-in annotation types (notes and links), and clients can add new annotation handlers by using [AVAppRegisterAnnotHandler](#).

Obtaining

[AVAppGetAnnotHandlerByName](#)
[AVAppEnumAnnotHandlers](#)

Enumerating

[AVAppEnumAnnotHandlers](#)

Attributes

Get	Set
AVAnnotHandlerGetInfo	AVAnnotHandlerDeleteInfo

Declarations

[AVAnnotHandler](#)

AVApp

The Acrobat viewer application itself. From the application layer, you can control the appearance of Acrobat, whether Acrobat appears, and the size of the application window. Your application has access to the menubar and the toolbar through this object. The application layer also provides access to the visual representation of a PDF file on the screen, that is, an **AVDoc**.

Enumerating

[AVAppEnumActionHandlers](#)
[AVAppEnumAnnotHandlers](#)
[AVAppEnumDocs](#)
[AVAppEnumSystemFonts](#)
[AVAppEnumTools](#)
[AVAppEnumTransHandlers](#)

Attributes

Get	Set
AVAppGetHowToPanelAutoShow	AVAppRegisterHowToPanel , AVAppSetHowToPanelAutoShow , AVAppSetHowToPanelAutoShowText
AVAppCanQuit	—
AVAppGetActionHandlerByType	AVAppRegisterActionHandler
AVAppGetActiveDoc	—
AVAppGetActiveTool	AVAppSetActiveTool , AVAppRegisterTool
AVAppGetAnnotHandlerByName	AVAppRegisterAnnotHandler
—	AVAppRegisterForContextMenuAddition
—	AVAppRegisterForPageViewAdjustCursor , AVAppUnregisterForPageViewAdjustCursor
—	AVAppRegisterForPageViewClicks , AVAppUnregisterForPageViewClicks
—	AVAppRegisterForPageViewDrawing , AVAppUnregisterForPageViewDrawing
—	AVAppRegisterIdleProc , AVAppUnregisterIdleProc

Get	Set
—	AVAppRegisterNotification, AVAppUnregisterNotification
AVAppGetCancelProc	—
AVAppGetDefaultTool	—
AVAppGetDocProgressMonitor	—
AVAppGetLanguage	—
AVAppGetLastActiveTool	—
AVAppGetMenubar	—
AVAppGetName	—
AVAppGetNumDocs	—
AVAppGetPreference	AVAppSetPreference
AVAppGetReportProc	—
AVAppGetToolBar	—
AVAppGetToolByName	—
AVAppGetTransHandlerByType	—
AVAppGetVersion	—
AVHasAuxDataHandler	AVRegisterAuxDataHandler, AVUnregisterAuxDataHandler

Actions

AVAppBeginModal
 AVAppEndModal
 AVAppModalWindowIsOpen
 AVAppBeginFullScreen
 AVAppDoingFullScreen
 AVAppEndFullScreen
 AVAppHandlePlatformEvent
 AVAppHelpSearch
 AVAppHelpShowContents
 AVAppHelpShowIndex
 AVAppOpenHelpFileWithParams
 AVAppAutoShowHowToPanel



AVCommand

An **AVCommand** represents an action that the user can perform on the current document or the current selection in the current document. An **AVCommand** can be added to a command sequence and executed either interactively or via batch processing, using [AVCommandExecute](#). Commands can be cancelled with [AVCommandCancel](#).

Obtaining:

[AVCommandNew](#)

Disposing:

[AVCommandDestroy](#)

Attributes

Get	Set
AVCommandGetAVDoc	—
AVCommandGetCab	AVCommandPutCab
AVCommandGetCancelProc	—
AVCommandGetConfig	AVCommandSetConfig
AVCommandGetInputs	AVCommandSetInputs
AVCommandGetName	—
AVCommandGetParams	AVCommandSetParams
AVCommandGetPDDoc	—
AVCommandGetProgressMonitor	—
AVCommandGetProps	—
AVCommandGetReportProc	—
AVCommandGetStatus	—
AVCommandGetUIPolicy	—

Actions

[AVCommandCancel](#)
[AVCommandExecute](#)
[AVCommandReset](#)
[AVCommandShowDialog](#)
[AVCommandWork](#)



AVConversion

This is not a data structure, but a conceptual entity that collects methods and data structures used to convert from and to PDF format.

Actions

- AVConversionConvertFromPDFWithHandler
- AVConversionConvertStreamFromPDFWithHandler
- AVConversionConvertStreamFromStructNodeWithHandler
- AVConversionConvertStreamToPDF
- AVConversionConvertStreamToPDFWithHandler
- AVConversionConvertToPDF
- AVConversionConvertToPDFWithHandler
- AVConversionEnumFromPDFConverters
- AVConversionEnumToPDFConverters

Declarations

- AVConversionClientData
- AVConversionEnumProcData
- AVConversionFlags
- AVConversionFromPDFHandler
- AVConversionMimeTypeString
- AVConversionStatus
- AVConversionToPDFHandler



AVDoc

A view of a PDF document in a window. There is one **AVDoc** per displayed document. Unlike a **PDDoc**, an **AVDoc** has a window associated with it.

Obtaining:

[AVAppGetActiveDoc](#)
[AVAppEnumDocs](#)
[AVDocOpenFromASFileWithParams](#)
[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromFile](#)
[AVDocOpenFromASFileWithParams](#)
[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromPDDoc](#)
[AVDocOpenFromPDDocWithParams](#)
[AVDocOpenFromPDDocWithParamString](#)
[AVPageViewGetAVDoc](#)
[AVUndoGetAVDoc](#)

Disposing

[AVDocClose](#)

Enumerating

[AVAppEnumDocs](#)
[AVDocEnumSelection](#)

Attributes

Get	Set
AVDocGetAVWindow	—
AVDocGetClientName	AVDocSetClientName
AVDocGetPageText	—
AVDocGetPageText, AVDocGetPageView, AVDocGetNthPageView, AVDocGetNumPageViews	—
AVDocGetPDDoc	—
—	AVDocClearSelection
—	AVDocDeleteSelection

Get	Set
—	AVDocShowSelection
AVDocGetSelection	AVDocSetSelection
AVDocGetSelectionServerByType	—
AVDocGetSelectionType	—
AVDocGetSplitterPosition	AVDocSetSplitterPosition
AVDocGetTopUndo	AVDocClearUndos
AVDocGetViewDef , AVDocGetViewDefEx	AVDocSetViewDef , AVDocSetViewDefEx
AVDocGetViewMode	AVDocSetViewMode
AVDocIsExternal	—
—	AVDocRegisterSelectionServer
—	AVDocSetDead
AVDocIsReadOnly	AVDocSetReadOnly

Actions

[AVDocBeginUndoOperation](#)
[AVDocDoActionPropsDialog](#)
[AVDocDoCopyAs](#)
[AVDocDoPrint](#)
[AVDocDoSaveAs](#)
[AVDocDoSaveAsWithParams](#)
[AVDocDoSelectionProperties](#)
[AVDocEndUndoOperation](#)
[AVDocPerformAction](#)

Declarations

[AVDocOpenParams](#)
[AVDocPrintParams](#)
[AVDocSelectionServer](#)
[AVDocViewDef](#)



AVGrafSelect

A graphics selection on a page in a PDF file. It is a rectangular region of a page that can be copied to the clipboard as a sampled image.

Obtaining

[AVDocGetSelection](#)
[AVGrafSelectCreate](#)

Disposing

[AVGrafSelectDestroy](#)

Attributes

Get	Set
AVGrafSelectGetBoundingRect	—

AVMenu

A menu in the Acrobat viewer's menubar. Clients can create new menus, add menu items at any location in any menu, and remove menu items. Deleting an **AVMenu** removes it from the menubar (if it was attached) and deletes all the menu items it contains.

Obtaining

[AVMenuAcquire](#)
[AVMenuNew](#)
[AVMenuItemAcquireSubmenu](#)
[AVMenuItemGetParentMenu](#)
[AVMenubarAcquireMenuByName](#)
[AVMenubarAcquireMenuByIndex](#)
[AVMenubarAcquireMenuByPredicate](#)

Disposing

[AVMenuRelease](#)
[AVMenuRemove](#)

Attributes

Get	Set
—	AVMenuAddMenuItem
AVMenuGetMenuItemIndex	—
AVMenuGetName	—
AVMenuGetNumMenuItems	—
AVMenuGetParentMenubar	—
AVMenuGetParentMenuItem	—
AVMenuGetTitle	—

AVMenubar

The Acrobat viewer's menubar and a list of all menus. There is only one **AVMenubar**. Clients can add new menus to or remove any menu from the menubar. The menubar can be hidden from the user's view.

Obtaining

[AVAppGetMenubar](#)

[AVMenuGetParentMenubar](#)

[AVAppGetMenubar](#) is the standard way to obtain the menubar.

Attributes:

Get	Set
—	AVMenubarAddMenu
—	AVMenuRemove
AVMenubarAcquireMenuByIndex	—
AVMenubarAcquireMenuByName	—
AVMenubarAcquireMenuByPredicate	—
AVMenubarAcquireMenuItemByName	—
AVMenubarAcquireMenuItemByPredicate	—
AVMenubarGetMenuIndex	—
AVMenubarGetNumMenus	—
AVMenuItemRemove	—
—	AVMenubarHide
—	AVMenubarShow

AVMenuItem

A menu item under a menu in the Acrobat viewer. It has a number of attributes, including a name, a keyboard shortcut, a procedure to execute when the menu item is selected, a procedure to compute whether the menu item is enabled, a procedure to compute whether the menu item is check marked, and whether it has a submenu.

Obtaining

[AVMenuItemNew](#)
[AVMenuItemAcquire](#)
[AVMenubarAcquireMenuByName](#)
[AVMenubarAcquireMenuItemByPredicate](#)
[AVMenuAcquireMenuItemByIndex](#)
[AVMenuGetParentMenuItem](#)

[AVMenuDoPopUp](#)
[AVPageViewDoPopupMenu](#)

Disposing

[AVMenuItemRelease](#)
[AVMenuItemRemove](#)

Attributes

Get	Set
AVMenuGetMenuItemIndex	—
AVMenuItemAcquireSubmenu	—
AVMenuItemGetLongOnly	—
AVMenuItemGetName	—
AVMenuItemGetParentMenu	—
AVMenuItemGetShortcut	—
AVMenuItemGetTitle	AVMenuItemSetTitle
AVMenuItemIsEnabled	AVMenuItemSetComputeEnabledProc
AVMenuItemIsMarked	AVMenuItemSetComputeMarkedProc
AVMenuItemIsVisible	AVMenuItemSetComputeVisibleProc
—	AVMenuItemSetExecuteProc

AVPageView

The area of the Acrobat viewer's window that displays the contents of a document page. Every **AVDoc** has an **AVPageView** and vice versa. It contains references to the **PDDoc** and **PDPPage** objects for the document being displayed.

Obtaining

[AVDocGetPageView](#)

Attributes

Get	Set
AVPageViewAppearanceGetAVMatrix	—
AVPageViewAcquireMachinePort	AVPageViewReleaseMachinePort
AVPageViewGetActiveBead	—
AVPageViewGetAnnotRect	—
—	AVPageViewSetAnnotLocation
AVPageViewGetAperture	—
AVPageViewGetAVDoc	—
AVPageViewGetColor	AVPageViewSetColor
—	AVPageViewShowControl
AVPageViewGetDevToPageMatrix , AVPageViewGetPageToDevScaling AVPageViewGetPageToDevMatrix	—
AVPageViewGetFirstVisiblePageNum , AVPageViewGetLastVisiblePageNum	—
AVPageViewGetFocusAnnot	AVPageViewSetFocusAnnot
AVPageViewGetLayoutMode	AVPageViewSetLayoutMode
AVPageViewGetMousePosition , AVPageViewGetMousePositionSnapped	—
AVPageViewGetNextView	—
AVPageViewGetPage	—
AVPageViewGetPageNum	AVPageViewSetPageNum

Get	Set
AVPageViewGetSelectedAnnotPageNum	—
AVPageViewGetThreadIndex	—
AVPageViewGetVisibleAnnotPage	—
AVPageViewGetZoom, AVPageViewGetZoomType	
—	AVPageViewDevicePointToPage, AVPageViewDeviceRectToPage, AVPageViewDeviceRectToPageRZ, AVPageViewInfoToDevice AVPageViewInfoToPoint AVPageViewPointToDevice, AVPageViewPointToInfo, AVPageViewRectToDevice
—	AVPageViewInvalidateRect, AVPageViewInvalidateText
AVPageViewIsAnnotAtPoint, AVPageViewIsAnnotOfTypeAtPoint, AVPageViewIsFocusAnnot	AVPageViewSetAnnotLocation
AVPageViewIsBeadAtPoint	—
AVPageViewPageNumIsVisible	—
AVPageViewPointInText	—
	AVPageViewUseDestInfo, AVPageViewUseThisDestination
	AVPageViewUpdateInfoPanel

Actions

AVPageViewDragRect
 AVPageViewDragRectSnapped
 AVPageViewDragRectSnappedEx
 AVPageViewDrawCosObj
 AVPageViewDrawNow
 AVPageViewDrawRect
 AVPageViewDrawRectOutline
 AVPageViewGoBack



AVPageViewGoForward
AVPageViewGoTo
AVPageViewHighlightText
AVPageViewInsetRect
AVPageViewInvertQuad
AVPageViewInvertRect
AVPageViewInvertRectOutline
AVPageViewReadPageDown
AVPageViewReadPageUp
AVPageViewResumeOffscreenDrawing
AVPageViewScrollTo
AVPageViewScrollToRect
AVPageViewSnapPoint
AVPageViewSnapRect
AVPageViewStartReadingThread
AVPageViewSuspendOffscreenDrawing
AVPageViewToDestInfo
AVPageViewToViewDest
AVPageViewTrackText
AVPageViewZoomTo

Declarations

AVDestInfo
AVDragRectParams
AVDragType
AVInfoPanelUpdateType
AVPageViewControlID
AVRect
AVZoomType



AVSweetPea

The Sweet Pea methods are used to implement the Adobe Dialog Manager (ADM). See [Adobe Dialog Manager Reference](#).

Attributes

Get	Set
AVSweetPeaGetBasicSuiteP	—
AVSweetPeaGetPluginRef	—
AVSweetPeaGetResourceAccess	AVSweetPeaSetResourceAccess

Declarations

See [Adobe Dialog Manager Reference](#).



AVSys

Provides various system-wide utilities, including setting the cursor shape and beeping.

Attributes

Get	Set
AVSysGetCursor , AVSysGetStandardCursor	AVSysSetCursor , AVSysSetWaitCursor
AVSysGetIconFromFilename , AVSysGetIconFromMimeType , AVSysGetIconFromTypeAndCreator	—
AVSysGetModifiers	—
AVSysGetUsePenForInput	—
AVSysMouseIsStillDown	—

Actions

[AVSysAllocTimeStringFromTimeRec](#)
[AVSysBeep](#)



AVTool

Handles key presses and mouse clicks in the content region of an **AVPageView**. **AVTools** do not handle mouse clicks in other parts of the viewer's window, such as in the bookmark pane. At any time, there is one active tool.

Obtaining

[AVAppGetActiveTool](#)
[AVAppGetLastActiveTool](#)
[AVAppGetDefaultTool](#)
[AVAppGetToolByName](#)
[AVAppEnumTools](#)

Enumerating

[AVAppEnumTools](#)

Attributes

Get	Set
AVToolGetType	—
AVToolIsPersistent	—

Declarations

[AVTool](#)



AVToolBar

The Acrobat viewer's toolbar (the palette of buttons).

Obtaining

[AVAppGetToolBar](#)
[AVToolBarNew](#)
[AVToolBarNewFlyout](#)
[AVToolButtonGetFlyout](#)

Attributes

Get	Set
AVToolBarGetButtonByName	—
AVToolBarGetFrame	—
AVToolBarGetNumButtons	—
AVToolBarIsRoomFor	—
—	AVToolBarAddButton
—	AVToolBarUpdateButtonStates
—	AVToolButtonDestroy
—	AVToolButtonRemove
—	AVToolButtonSetExternal



AVToolButton

A button in the Acrobat viewer's toolbar. Like menu items, the procedure that executes when the button is clicked can be set by a client. Although not required, there is generally a menu item corresponding to each button, allowing users to select a function using either the button or the menu item. Buttons are added to the toolbar by specifying which existing button they appear before or after.

Obtaining

[AVToolBarGetButtonByName](#)
[AVToolButtonNew](#)
[AVToolBarEnumButtons](#)

Disposing

[AVToolButtonDestroy](#)
[AVToolButtonRemove](#)

Enumerating

[AVToolBarEnumButtons](#)

Attributes

Get	Set
AVToolButtonGetFlyout	AVToolButtonSetFlyout
AVToolButtonGetIcon	AVToolButtonSetIcon
AVToolButtonGetLabelText	AVToolButtonSetLabelText
AVToolButtonGetMenu	AVToolButtonSetMenu
AVToolButtonIsEnabled	AVToolButtonSetComputeEnabledProc
AVToolButtonIsMarked	AVToolButtonSetComputeMarkedProc
AVToolButtonIsSeparator	—
—	AVToolButtonSetNotifyTooltipProc
AVToolButtonIsVisible	AVToolButtonSetComputeVisibleProc
—	AVToolButtonSetExecuteProc
—	AVToolButtonSetExternal
—	AVToolButtonSetHelpText



Declarations

`Tool Button Flags`



AVUndo

Represents an undo record for a document. An undo record allows a client to associate private data with a particular **AVDoc** for the purpose of undoing and redoing changes to the document. The client provides private data that encapsulates the changes, and an [AVUndoHandler](#) that contains callbacks which interpret the data when **Undo** and **Redo** commands are issued.

Obtaining

[AVUndoNew](#)
[AVDocBeginUndoOperation](#)

Disposing

[AVDocClearUndos](#)
[AVDocEndUndoOperation](#)

Attributes

Get	Set
AVUndoGetAVDoc	
AVUndoGetData	AVUndoSetData
AVUndoGetType	

Declarations

[AVUndoHandler](#)
[AVUndoHandlerData](#)

AVWindow

Creates and manages windows. Clients should use **AVWindows** for their own dialogs, floating palettes, and so forth, to ensure that those windows work well with the Acrobat viewer. For example, under Windows they are hidden when the Acrobat viewer is turned into an icon. Once the client creates an **AVWindow**, it is free to use platform-dependent code to put whatever it wants in the window.

Obtaining

[AVWindowNew](#)
[AVWindowNewFromPlatformThing](#)
[AVDocGetAVWindow](#)

Disposing

[AVWindowDestroy](#)
[AVWindowUserClose](#)

Attributes

Get	Set
AVWindowGetFrame	AVWindowSetFrame
AVWindowGetInterior	—
AVWindowGetOwnerData	AVWindowSetOwnerData
AVWindowGetPlatformThing	—
AVWindowGetTitle	AVWindowSetTitle
AVWindowIsKey	, AVWindowSetWantsKey ,
AVWindowIsVisible	
—	AVWindowInvalidateRect

Actions

[AVWindowBecomeKey](#)
[AVWindowBringToFront](#)
[AVWindowDoModal](#)
[AVWindowDrawNow](#)
[AVWindowEndModal](#)
[AVWindowHide](#)
[AVWindowMaximize](#)



AVWindowResignKey
AVWindowShow

Declarations

AVWindow Flags
AVWindowHandler
AVWindowLayer



Cos Layer

A set of objects that provide access to the building blocks used to construct documents. Its methods allow applications to manipulate the low-level data in a PDF file, such as strings, numbers, and dictionaries.

CosArray

A Cos-level representation of an array.

Obtaining

[CosNewArray](#)

Attributes

Get	Set
CosArrayGet	CosArrayPut
CosArrayLength	—
	CosArrayInsert
	CosArrayRemove
	CosArrayRemoveNth



CosBoolean

A Cos-level representation of a boolean value.

Obtaining

[CosNewBoolean](#)

Attributes

Get	Set
CosBooleanValue	—



CosDict

A Cos-level representation of a dictionary.

Obtaining

[CosNewDict](#)
[CosDocGetInfoDict](#)
[CosStreamDict](#)

Attributes

Get	Set
CosDictGet	CosDictPut
CosDictKnown	—
—	CosDictRemove



CosDoc

A Cos-level representation of an entire PDF file.

Obtaining

[CosDocCreate](#)
[CosDocOpenWithParams](#)
[CosObjGetDoc](#)
[PDDocGetCosDoc](#)

Attributes

Get	Set
CosDocGetID	—
CosDocGetInfoDict	—
CosDocGetObjByID	—
CosDocHasFullCompression CosDocHasPartialCompression	
—	CosDocSetDirty

Actions

[CosDocClose](#)
[CosDocSaveToFile](#)
[CosDocSaveWithParams](#)
[CosSetMaxDocStorages](#)

Declarations

[CosDocOpenParams](#)
[CosDocOpenParams](#)
[CosDocSaveFlags](#)
[CosDocSaveParams](#)



CosFixed

A Cos-level representation of a fixed-point number.

Obtaining

[CosNewFixed](#)

Attributes

Get	Set
CosFixedValue	—



CosInteger

A Cos-level representation of an integer.

Obtaining

[CosNewInteger](#)

Attributes

Get	Set
CosIntegerValue	—



CosName

A Cos-level representation of a name.

Obtaining

[CosNewName](#)

Attributes

Get	Set
CosNameValue	—



CosNull

A Cos-level representation of a **null** PDF object.

Obtaining

[CosNewNull](#)



CosObj

A general object in a PDF file, which may be of any Cos object type.

Obtaining

[CosDocGetObjByID](#)
[PDActionGetCosObj](#)
[PDAnnotGetCosObj](#)
[PDBeadGetCosObj](#)
[PDBookmarkGetCosObj](#)
[PDCharProcGetCosObj](#)
[PDFFileSpecGetCosObj](#)
[PDFontGetCosObj](#)
[PDFormGetXUIDCosObj](#)
[PDNameTreeGetCosObj](#)
[PDPageGetCosObj](#)
[PDPageGetCosResources](#)
[PDPageLabelGetCosObj](#)
[PDThreadGetCosObj](#)
[PDTransGetCosObj](#)
[PDViewDestGetCosObj](#)
[PDXObjectGetCosObj](#)

Disposing

[CosObjDestroy](#)

Enumerating

[CosObjEnum](#)

Actions

[CosObjEqual](#)

Attributes

Get	Set
CosDocObjIsWithinRange	—
CosObjGetCollection	CosObjAddToCollection , CosObjRemoveFromCollection
CosObjGetCompressibility , CosObjIsCompressed	CosObjSetCompressibility
CosObjGetDoc	—



Get	Set
CosObjGetGeneration	—
CosObjGetID	—
CosObjHash	—
CosObjIsIndirect	—

Actions

[CosObjRefreshAfterLinearizedSave](#)

Cos conversion

See “Cos conversion” for each object.

Declarations

[CosType](#)

CosObjCollection

An opaque structure representing a set of Cos objects associated with a particular Cos document. The initial value of a variable of type **ObjCollection** is undefined. Use [CosNewObjCollection](#) to create a collection.

Any indirect object whose generation number is zero and which is not a stream may be added to at most one **CosObjCollection**. When the file is saved, all the objects in a given collection are stored together in the PDF file, in one or more “object streams” (see the [PDF Reference](#)), which are normally compressed in order to reduce file size.

Collections allow grouping of objects that are likely to have a similar usage pattern. If one of them is required (and therefore decompressed), they are likely to all be required, but it is possible that none will be required. For example, there could be a collection that stores bookmarks. If the user never opens the bookmarks panel in Acrobat, none of the objects need to be decoded. If the user opens the bookmarks panel, it is likely they will all be needed, and having them together in the PDF file reduces the amount of time needed to load them all.

NOTE: If the document is saved with full compression, partial compression, or in linearized format, any existing collections are ignored.

Obtaining

[CosNewObjCollection](#)
[CosObjGetCollection](#)

Enumerating

[CosObjCollectionEnum](#)

Attributes

Get	Set
CosObjCollectionEqual	—
CosObjCollectionIsNull	—
CosObjCollectionSize	—



CosStream

A Cos-level representation of a stream.

Obtaining

[CosNewStream](#)
[CosStreamOpenStm](#)

Attributes

Get	Set
CosStreamDict	—
CosStreamLength	—
CosStreamPos	—

Declarations

[CosStreamOpenMode](#)



CosString

A Cos-level representation of a string.

Obtaining

[CosNewString](#)

Attributes

Get	Set
CosStringGetHexFlag	CosStringSetHexFlag
CosCopyStringValue	
CosStringValue	
CosStringValueSafe	

PD Layer

A group of objects that provide access to components of PDF documents such as pages, annotations, and fonts. Its methods allow applications to manipulate document components.

PDAction

Actions are what happens when a user clicks on a link or bookmark. In addition, the Acrobat viewer allows a document to have an action that is executed automatically when the document is opened. Applications can also support actions in custom annotation types they add.

Obtaining

[PDActionNew](#)
[PDActionNewFromDest](#)
[PDActionNewFromFileSpec](#)
[PDLinkAnnotGetAction](#)
[PDBookmarkGetAction](#)
[PDDocGetOpenAction](#)
[PDActionCopy](#)
[PDActionPaste](#)

Disposing

[PDActionDestroy](#)
[PDActionDestroyClipboardData](#)

Attributes

Get	Set
PDActionCanCopy	—
PDActionCanPaste	—
PDActionEqual	—
PDActionGetDest	—
PDActionGetFileSpec	—
PDActionGetSubtype	—

Cos conversion

[PDActionGetCosObj](#)
[PDActionFromCosObj](#)

**Validity testing**

`PDActionIsValid`

Declarations

`PDActionClipboardData`

PDAnnot

An annotation on a page in a PDF file. Acrobat viewers have two built-in annotation types: **PDTextAnnot** and **PDLinkAnnot**. Physical attributes of the annotation can be set and queried. Plug-ins add movie and Widget (form field) annotations. Developers can define new annotation subtypes by creating new annotation handlers.

Obtaining

[AVPageViewIsAnnotAtPoint](#)
[PDAnnotFromCosObj](#)
[PDPageAddNewAnnot](#)
[PDPageCreateAnnot](#)
[PDPageGetAnnot](#)
[PDAnnotCopy](#)
[PDAnnotPaste](#)

Disposing

[PDPageRemoveAnnot](#)
[PDAnnotDestroyClipboardData](#)

Attributes

Get	Set
AVPageViewGetSelectedAnnotPageNum	—
—	AVPageViewSetAnnotLocation
PDAnnotCanCopy PDAnnotCanPaste	—
PDAnnotEqual	—
PDAnnotGetColor	PDAnnotSetColor
PDAnnotGetDate	PDAnnotSetDate
PDAnnotGetFlags	PDAnnotSetFlags
PDAnnotGetRect	PDAnnotSetRect
PDAnnotGetSubtype	—
PDAnnotGetTitle	PDAnnotSetTitle
PDAnnotGetOCMD	PDAnnotSetOCMD PDAnnotRemoveOCMD



Get	Set
PDAannotIsCurrentlyVisible	

Cos conversion

PDAannotGetCosObj
PDAannotFromCosObj

Validity testing

PDAannotIsValid

Declarations

PDAannotArray
PDAannotClipboardData
PDAannot Flags
PDAannotHandler
PDAannotInfo



PDBead

A single rectangle in an article thread. (Article threads are known simply as articles in the Acrobat viewer's user interface.) A bead remains valid as long as a thread is "current and active."

Obtaining

[AVPageViewGetActiveBead](#)
[AVPageViewIsBeadAtPoint](#)
[PDBeadNew](#)
[PDBeadGetNext](#)
[PDBeadGetPrev](#)
[PDThreadGetFirstBead](#)

Disposing

[PDBeadDestroy](#)

Attributes

Get	Set
PDBeadGetIndex	—
PDBeadGetNext	—
PDBeadGetPrev	—
PDBeadGetRect	PDBeadSetRect
PDBeadGetThread	—
PDBeadIsValid	—
—	PDBeadSetPage

Cos conversion

[PDBeadGetCosObj](#)
[PDBeadFromCosObj](#)

Validity testing

[PDBeadIsValid](#)

PDBookmark

A bookmark on a page in a PDF file. Each bookmark has a title that appears on screen, and an action that specifies what happens when a user clicks on the bookmark. Bookmarks can either be created interactively by the user through the Acrobat viewer's user interface or programmatically generated. The typical action for a user-created bookmark is to move to another location in the current document, although any action (see [PDAction](#)) can be specified.

Obtaining

[PDDocGetBookmarkRoot](#)
[PDBookmarkAddNewSibling](#)
[PDBookmarkAddNewChild](#)
[PDBookmarkFromCosObj](#)
[PDBookmarkGetByTitle](#)
[PDBookmarkGetParent](#)
[PDBookmarkGetFirstChild](#)
[PDBookmarkGetLastChild](#)
[PDBookmarkGetNext](#)
[PDBookmarkGetPrev](#)

Disposing

[PDBookmarkDestroy](#)
[PDBookmarkUnlink](#)

Attributes

Get	Set
PDBookmarkEqual	—
PDBookmarkGetAction	PDBookmarkSetAction PDBookmarkRemoveAction
PDBookmarkGetColor	PDBookmarkSetColor
PDBookmarkGetCount	—
PDBookmarkGetFirstChild	PDBookmarkAddChild PDBookmarkAddNewChild
PDBookmarkGetFlags	PDBookmarkSetFlags
PDBookmarkGetIndent	—



Get	Set
PDBookmarkGetLastChild	PDBookmarkAddChild PDBookmarkAddNewChild
PDBookmarkGetNext	PDBookmarkAddNext PDBookmarkAddNewSibling
PDBookmarkGetParent	PDBookmarkAddSubtree
PDBookmarkGetPrev	PDBookmarkAddPrev PDBookmarkAddNewSibling
PDBookmarkGetTitle	PDBookmarkSetTitle
PDBookmarkHasChildren	—
PDBookmarkIsOpen	PDBookmarkSetOpen

Cos conversion

PDBookmarkGetCosObj
PDBookmarkFromCosObj

Validity testing

PDBookmarkIsValid



PDCharProc

A character procedure, a stream of graphic operators (see [PDGraphic](#)) that draw a particular glyph of a Type 3 PostScript font.

Enumerating

- [PDCharProcEnum](#)
- [PDCharProcEnumWithParams](#)
- [PDFontEnumCharProcs](#)

Cos conversion

- [PDCharProcGetCosObj](#)

PDDoc

The underlying PDF representation of a document. There is a correspondence between a **PDDoc** and an **ASFile**; the **PDDoc** object is the hidden object behind every **AVDoc**. An **ASFile** may have zero or more underlying files, so a PDF file does not always correspond to a single disk file. For example, an **ASFile** may provide access to PDF data in a database.

Through **PDDocs**, your application can perform most of the Edit -> Pages menu items from Acrobat (delete, replace, and so on). Thumbnails can be created and deleted through this object. You can set and retrieve document information fields through this object as well.

The first page in a **PDDoc** is page 0.

Obtaining

[AVDocGetPDDoc](#)
[PDDocFromCosDoc](#)
[PDDocOpen](#)
[PDDocOpenFromASFile](#)
[PDDocOpenWithParams](#)
[PDDocCreate](#)
[PDOCConfigGetPDDoc](#)
[PDPageGetDoc](#)
[PDFFileSpecGetDoc](#)
[PDEnumDocs](#)

Disposing

[PDDocClose](#)
[PDDocRelease](#)

Enumerating

[PDDocEnumFonts](#)
[PDDocEnumLoadedFonts](#)
[PDDocEnumOCGs](#)
[PDDocEnumOCConfigs](#)
[PDEnumDocs](#)

Attributes

Get	Set
AVAuthOpen	PDDocAuthorize
PDCryptAuthorizeFilterAccess	PDDocSetNewCryptFilterData
	PDDocSetNewCryptFilterMethod
	PDDocSetNewDefaultFilters
—	PDDocAcquire

Get	Set
PDDocAcquirePage	PDDocCreatePage PDDocDeletePages PDDocImportNotes PDDocMovePage PDDocReplacePages
—	PDDocAuthorize
—	PDDocCreateThumbs PDDocDeleteThumbs
PDDocExportNotes	—
PDDocGetCryptHandlerClientData PDDocGetNewCryptHandler	PDDocSetNewCryptHandler PDDocSetNewCryptHandlerEx
PDDocGetFile	—
PDDocGetFlags	PDDocClearFlags PDDocSetFlags
PDDocGetFullScreen	PDDocSetFullScreen
PDDocGetID	—
PDDocGetInfo	PDDocSetInfo
PDDocGetLabelForPageNum	PDDocRemovePageLabel
PDDocGetNameTree	PDDocRemoveNameTree
PDDocGetNumPages	—
PDDocGetOCConfig PDDocGetOCContext PDDocGetOCGs PDDocGetNumOCGs PDDocHasOC	PDDocReplaceOCG
PDDocGetOpenAction	PDDocSetOpenAction PDDocRemoveOpenAction
PDDocGetPageLabel PDDocFindPageNumForLabel	PDDocSetPageLabel
PDDocGetPageObjByNum	—

Get	Set
PDDocGetPageMode	PDDocSetPageMode
PDDocGetPermissions	—
PDDocGetSecurityData	PDDocNewSecurityData
PDDocGetNewSecurityData	PDDocSetNewSecurityData
PDDocGetNewSecurityInfo	
PDDocGetStructTreeRoot	PDDocRemoveStructTreeRoot PDDocCreateStructTreeRoot
PDDocGetThread	PDDocAddThread
PDDocGetThreadIndex	PDDocRemoveThread
PDDocGetNumThreads	
PDDocGetTrapped	PDDocSetTrapped
PDDocGetWordFinder	PDDocCreateWordFinder PDDocCreateWordFinderUCS PDWordFinderDestroy
PDDocGetXAPMetadata	PDDocSetXAPMetadata
PDDocGetXAPMetadataProperty	PDDocSetXAPMetadataProperty
PDDocGetVersion	—
—	PDDocFlattenOC

Cos Conversion

[PDDocGetCosDoc](#)
[PDDocFromCosDoc](#)

Actions

[PDDocRequestEntireFile](#)
[PDDocRequestPages](#)
[PDDocSave](#)
[PDDocSaveWithParams](#)

Declarations

[PDDocCopyParams](#)
[PDDocFlags](#)
[PDDocInsertPagesParams](#)
[PDDocOCChangeType](#)
[PDDocOpenParams](#)
[PDDocPreSaveInfo](#)



PDDocReadAhead Flags
PDDocSaveParams



PDFFileSpec

The PDF file specification object. It is used to specify a file in an action (see [PDAction](#)). A file specification in a PDF file can take two forms:

- A single platform-independent pathname.
- A data structure containing one or more alternative ways to locate the file on different platforms.

PDFFileSpecs can be created from [ASPathNames](#) or from Cos objects.

Obtaining

[PDActionGetFileSpec](#)
[PDFFileSpecNewFromASPath](#)
[PDFFileSpecFromCosObj](#)

Attributes

Get	Set
PDFFileSpecAcquireASPath	—
PDFFileSpecGetDIPath	—
PDFFileSpecGetDoc	—
PDFFileSpecGetFileSys	—
PDFFileSpecGetFileSysName	—

Cos conversion

[PDFFileSpecGetCosObj](#)
[PDFFileSpecFromCosObj](#)

Validity testing

[PDFFileSpecIsValid](#)

Declarations

[PDFFileSpecHandler](#)

PDFont

A font that is used to draw text on a page. It corresponds to a Font Resource in a PDF file. Applications can get a list of **PDFonts** used on a **PDPage** or a range of **PDPages**. More than one **PDPage** may reference the same **PDFont** object.

A **PDFont** has a number of attributes whose values can be read or set, including an array of widths, the character encoding, and the font's resource name.

Obtaining

[PDDocEnumFonts](#)
[PDDocEnumLoadedFonts](#)
[PDFontGetDescendant](#)
[PDStyleGetFont](#)

Enumerating

[PDDocEnumFonts](#)
[PDFontEnumCharProcs](#)

Attributes

Get	Set
PDFontAcquireEncodingArray	PDFontEncodingArrayRelease
PDFontAcquireXlateTable	PDFontXlateTableRelease
PDFontGetBBox	—
PDFontGetCharSet	—
PDFontGetCIDSystemInfo	—
PDFontGetCIDSystemSupplement	—
PDFontGetDescendant	—
PDFontGetEncodingIndex	—
PDFontGetEncodingName	—
PDFontGetFontMatrix	—
PDFontGetMetrics	PDFontSetMetrics
PDFontGetName	—
PDFontGetSubtype	—
PDFontGetWidths	—



Get	Set
PDFontIsEmbedded	—

Cos conversion

PDFontGetCosObj

PDForm

A self-contained set of graphics operators (essentially a subroutine of PDF page-marking operators) that is used when a particular graphic is drawn more than once in the document. It corresponds to a Form resource. You can use any [PDXObject](#) method on a **PDImage**.

Enumerating

[PDFormEnumPaintProc](#)
[PDFormEnumPaintProcWithParams](#)
[PDFormEnumResources](#)

Attributes

Get	Set
PDFormGetBBox	—
PDFormGetFormType	—
PDFormGetMatrix	—
PDFormGetXUIDCosObj	—

Cos Conversion

[PDXObjectGetCosObj](#)



PDGraphic

All graphic objects that comprise page, charproc, and **PDForm** descriptions.

Attributes

Get	Set
PDGraphicGetBBox	—
PDGraphicGetCurrentMatrix	—
PDGraphicGetState	—

Declarations

[PDGraphicEnumMonitor](#)
[PDGraphicEnumParams](#)
[PDGraphicState](#)



PDImage

A sampled image or image mask that corresponds to a PDF Image resource. You can use any [PDXObject](#) method on a **PDImage**.

Attributes

Get	Set
PDImageColorSpaceGetIndexLookup	—
PDImageGetAttrs	—

Cos Conversion

[PDXObjectGetCosObj](#)

Declarations

[PDImageAttrs](#)



PDInlineImage

An image whose data is stored in the page description's contents stream—instead of being stored as an image resource (see [PDImage](#)).

Attributes

Get	Set
PDInlineImageColorSpaceGetIndexLookup	—
PDInlineImageGetAttrs	—
PDInlineImageGetData	—

PDLinkAnnot

A link annotation on a page in a PDF file. You can use any **PDAnnot** method on a **PDLinkAnnot**. Applications can:

- Get and set the bounding rectangle and color, using **PDAnnot** methods.
- Get and set the action that occurs when the link is activated, and the link's border, using **PDLinkAnnot** methods.
- Create new link annotations and delete existing ones, using the **PDPage** methods.

Obtaining:

Any of the **PDAnnot** calls, followed by **CastToPDLinkAnnot**. The annotation passed to **CastToPDLinkAnnot** must be a link annotation: other annotation types are not converted into link annotations.

Disposing

PDPageRemoveAnnot

Attributes

Get	Set
PDLinkAnnotGetAction	PDLinkAnnotSetAction
PDLinkAnnotGetBorder	PDLinkAnnotSetBorder
AVPageViewGetSelectedAnnotPageNum	—
—	AVPageViewSetAnnotLocation
PDAnnotEqual	—
PDAnnotGetColor	PDAnnotSetColor
PDAnnotGetDate	PDAnnotSetDate
PDAnnotGetFlags	PDAnnotSetFlags
PDAnnotGetRect	PDAnnotSetRect
PDAnnotGetSubtype	—
PDAnnotGetTitle	PDAnnotSetTitle

Cos conversion

PDAnnotGetCosObj
PDAnnotFromCosObj

**Validity testing**[PDAnnotIsValid](#)**Declarations**[PDLinkAnnotBorder](#)

PDNameTree

The dictionary used to store all of the Named Destinations in a PDF file. A name tree is used to map Cos strings to Cos objects just as a Cos dictionary is used to map Cos names to Cos objects. However, a name tree can have many more entries than a Cos dictionary can. You create a **PDNameTree** and locate it where you think is appropriate (perhaps under a page, but most often right under the catalog).

Name trees use Cos-style strings (not null-terminated C strings), which may use Unicode encoding, and these may contain bytes with zeroes in them (high bytes of ASCII characters).

Obtaining

[PDDocCreateNameTree](#)
[PDNameTreeNew](#)
[PDNameTreeFromCosObj](#)

Enumerating

[PDNameTreeEnum](#)

Attributes

Get	Set
PDDocGetNameTree	—
PDNameTreeEqual	—
PDNameTreeLookup	—
PDNameTreeGet	PDNameTreePut
—	PDNameTreeRemove

Cos conversion

[PDNameTreeGetCosObj](#)

Validity testing

[PDNameTreeIsValid](#)

PDNumTree

An object that points to the root node of a number tree inside a PDF file. A number tree is used to map integers to arbitrary Cos objects just as a Cos dictionary is used to map Cos names to Cos objects. However, a number tree can have many more entries than a Cos dictionary can.

Obtaining

[PDNumTreeNew](#)
[PDNumTreeFromCosObj](#)

Enumerating

[PDNumTreeEnum](#)

Attributes

Get	Set
PDNumTreeEqual	—
PDNumTreeGet	PDNumTreePut
—	PDNumTreeRemove

Cos conversion

[PDNumTreeGetCosObj](#)

Validity testing

[PDNumTreeIsValid](#)

PDOCConfig

An optional-content configuration structure, used to maintain a set of visibility states and other optional-content information in a PDF file for future use. A document has a default configuration, saved in the **D** entry in the **OCProperties** dictionary, and can have a list of other configurations, saved as an array in the **Configs** entry in the **OCProperties** dictionary.

Configurations are typically used to initialize the optional-content group (**PDOC**) ON-OFF states for an optional-content context (**PDOCContext**). The OCG order in the configuration is the order in which the groups appear in the **Layers** panel of Acrobat 6.0. The configuration can define a set of mutually exclusive OCGs, called a radio button group.

Obtaining

[PDOCConfigCreate](#)
[PDDocGetOCConfig](#)

Disposing

[PDOCConfigDestroy](#)

Enumerating

[PDDocEnumOCConfigs](#)

Attributes

Get	Set
PDOCConfigGetAllRadioButtonGroups PDOCConfigGetRadioButtonGroupForOCG	PDOCConfigMakeRadioButtonGroup
PDOCConfigGetCreator	PDOCConfigSetCreator
PDOCConfigGetInitState	PDOCConfigSetInitState
PDOCConfigGetIntent	PDOCConfigSetIntent
PDOCConfigGetName	PDOCConfigSetName
PDOCConfigGetOCGOrder	PDOCConfigSetOCGOrder
PDOCConfigGetPDDoc	—

Cos conversion

[PDOCConfigGetCosObj](#)

Declarations

[PDOCConfigBaseState](#)

PDOCContext

An optional-content context in a document, within which document objects such as words or annotations are visible or hidden. The context keeps track the ON-OFF states of all of the optional-content groups (OCGs, represented by the [PDOCG](#) object) in a document. Content is or is not visible with respect to the OCG states stored in a specific context. The context does not correspond to any explicit PDF specification.

The [PDDoc](#) has a default context that it uses for on-screen drawing and that determines the default state for any other drawing or content enumeration. The context has flags that control whether to draw or enumerate content that *is* marked as optional ([PDOCDrawEnumType](#)), and whether to draw content that is *not* marked as optional ([NonOCDrawing](#)).

There can be more than one **PDOCContext** object, representing different combinations of OCG states. You can change the states of OCGs within any context. You can build contexts with your own combination of OCG states, and issue drawing or enumeration commands using that context instead of the document's default context. For example, you can pass an optional-content context to [PDPageDrawContentsWithParams](#) through the [PDDrawParams](#) structure. You can save the resulting state information as part of the configuration, but the context itself has no corresponding PDF form, and is not saved.

Obtaining

[PDOCContextNew](#)
[PDOCContextNewWithInitialState](#)
[PDOCContextNewWithOCDisabled](#)
[PDOCContextInit](#)

[PDOCContextMakeCopy](#)
[PDOCContextMakeCopyWithAutoStateChanges](#)

[PDDocGetOCContext](#)

Disposing

[PDOCContextFree](#)

Attributes

Get	Set
PDOCContextContentIsVisible	—
PDOCContextFindAutoStateChanges	PDOCContextApplyAutoStateChanges PDOCContextClearAllUserOverrides
PDOCContextGetNonOCDrawing	PDOCContextSetNonOCDrawing
PDOCContextGetOCDrawEnumType	PDOCContextSetOCDrawEnumType



Get	Set
PDOCContextGetOCGStates	PDOCContextSetOCGStates
PDOCContextGetIntent	PDOCContextSetIntent
PDOCContextGetPDDoc	—
PDOCContextPopOCMD	PDOCContextPushOCMD PDOCContextResetOCMDStack
PDOCContextXObjectIsVisible	—

Declarations

PDOCContextChangeType
PDOCContextInitPolicy
PDOCDrawEnumType

PDOCG

An optional-content group. This corresponds to a PDF **OCG** dictionary representing a collection of graphic objects that can be made visible or invisible. Any graphic content of the PDF can be made optional, including page contents, XObjects, and annotations. The specific content objects in the group have an **OC** entry in the PDF. The group itself is a named object that you can manipulate in the **Layers** panel of Acrobat 6.0.

In the simplest case, the group's ON-OFF state makes the associated content visible or hidden. The ON-OFF state of a group can be toggled for a particular context ([PDOCContext](#)), and a set of states is kept in a configuration ([PDOCConfig](#)). The visibility can depend on more than one group in an optional-content membership dictionary ([PDOCMD](#)), and can also be affected by the context's or configuration's [PDOCDrawEnumType](#).

Obtaining

[PDOCGCreate](#)
[PDOCGCreateFromCosObj](#)
[PDOCGGetFromCosObj](#)

[PDOCMDGetOCGs](#)
[PDDocGetOCGs](#)
[PDPageGetOCGs](#)

Disposing

[PDOCGDestroy](#)

Enumerating

[PDPageEnumOCGs](#)
[PDDocEnumOCGs](#)

Attributes

Get	Set
PDOCGGetCurrentState	PDOCGSetCurrentState
PDOCGGetInitialState	PDOCGSetInitialState PDOCGRemoveInitialState
PDOCGGetIntent	PDOCGSetIntent
PDOCGGetName	PDOCGSetName
PDOCGGetPDDoc	—
PDOCGHasUsageInfo PDOCGGetUsageEntry	PDOCGSetUsageDictEntry



Get	Set
PDOCGUsedInOCConfig PDOCGUsedInOCContext	—
PDOCGGetUserOverride	PDOCGSetUserOverride

Cos conversion

PDOCGGetCosObj
PDOCGGetFromCosObj



PDOCMD

An optional-content membership dictionary (OCMD) that allows the visibility of optional content to depend on the states in a set of optional-content groups ([PDOC](#)). The object corresponds to the PDF **OCMD** dictionary.

An OCMD collects a set of OCGs. It sets a visibility policy, so that content in the member groups is visible only when all groups are ON or OFF, or when any of the groups is ON or OFF. This makes it possible to set up complex dependencies among groups.

Obtaining

[PDOCMDCreate](#)
[PDOCMDFindOrCreate](#)
[PDOCMDGetFromCosObj](#)
[PDAnnotGetOCMD](#)
[PDEElementGetOCMD](#)

Attributes

Get	Set
PDOCMDIsCurrentlyVisible PDOCMDsAreCurrentlyVisible	PDOCMDsMakeContentVisible
PDOCMDGetVisPolicy	—
PDOCMDGetOCGs	—
PDOCMDGetPDDoc	—

Cos conversion

[PDOCMDGetCosObj](#)
[PDOCMDGetFromCosObj](#)

Declarations

[PDOCMDVisPolicy](#)

PDPage

A single page in the PDF representation of a document. Just as PDF files are partially composed of their pages, **PDDocs** are composed of **PDPages**. A page contains a series of objects representing the objects drawn on the page (**PDGraphic**), a list of resources used in drawing the page, annotations (**PDAnnot**), an optional thumbnail image of the page, and the beads used in any articles that occur on the page. The first page in a **PDDoc** is page 0.

Obtaining

[PDDocCreatePage](#)
[PDBeadAcquirePage](#)
[PDDocAcquirePage](#)
[AVPageViewGetPage](#)

Disposing

[PDDocDeletePages](#)
[PDPageRelease](#)

Enumerating

[PDPageEnumContents](#) (Obsolete in Acrobat 4.0)
[PDPageEnumOCGs](#)
[PDPageEnumResources](#)

Attributes

Get	Set
—	PDDocAcquirePage
PDPageAcquirePDEContent	PDPageSetPDEContent PDPageSetPDEContentCanRaise
PDPageGetAnnotIndex PDPageGetAnnotSequence	PDPageAddAnnot PDPageAddNewAnnot PDPageCreateAnnot PDPageRemoveAnnot
PDPageGetBBox PDPageGetVisibleBBox	—
PDPageGetBox PDPageGetCropBox PDPageGetMediaBox	PDPageSetBox PDPageSetCropBox PDPageSetMediaBox

Get	Set
PDPageGetCosResources	PDPageAddCosResource PDPageRemoveCosResource
PDPageDrawContentsToWindow PDPageDrawContentsToWindowEx PDPageDrawContentsWithParams PDPageDrawContentsPlacedWithParams	PDPageAddCosContents PDPageRemoveCosContents
PDPageGetDefaultMatrix PDPageGetFlippedMatrix	—
PDPageGetDoc	—
PDPageGetDuration	PDPageSetDuration
PDPageGetNumAnnots	—
PDPageGetNumber	—
PDPageGetOCGs	—
PDPageGetPalette	—
PDPageGetPDEContentFilters PDPageGetPDEContentFlags	PDPageSetPDEContentFilters PDPageSetPDEContentFlags
PDPageGetRotate	PDPageSetRotate
PDPageGetTransition PDPageHasTransition	PDPageSetTransition
—	PDPageFlattenOC

Cos conversion[PDPageGetCosObj](#)**Declarations**

[PDPageDrawFlags](#)
[PDPageMode](#)
[PDPageNumber](#)
[PDPageRange](#)
[PDPageStmToken](#)

PDPageLabel

A label used to describe a page. This is used to allow for non-sequential page numbering or the addition of arbitrary labels for a page (such as the inclusion of Roman numerals at the beginning of a book). A **PDPageLabel** specifies the numbering style to use (for example, upper- or lower-case Roman, decimal, and so forth), the starting number for the first page, and an arbitrary prefix to be pre-appended to each number (for example, "A-" to generate "A-1", "A-2", "A-3", and so forth.)

Obtaining

[PDDocGetPageLabel](#)
[PDDocGetLabelForPageNum](#)
[PDPageLabelFromCosObj](#)
[PDPageLabelNew](#)

Disposing

[PDDocRemovePageLabel](#)

Attributes

Get	Set
PDPageLabelEqual	—
—	PDDocSetPageLabel PDDocRemovePageLabel
PDPageLabelGetStyle	—
PDPageLabelGetPrefix	—
PDPageLabelGetStart	—
PDDocFindPageNumForLabel	—

Cos conversion

[PDPageLabelGetCosObj](#)

Validity testing

[PDPageLabelIsValid](#)



PDPath

A **PDPath** is a graphic object representing a path in a page description. Paths are arbitrary shapes made of straight lines, rectangles, and cubic curves. Path objects can be stroked, filled, and/or serve as a clipping path.

Enumerating

[PDPathEnum](#)

Attributes

Get	Set
PDPathGetPaintOp	—

Declarations

[PDPathEnumMonitor](#)

[PDPathPaintOp](#)



PDStyle

Provides access to information about the fonts, font sizes, and colors used in a **PDWord**.

Obtaining

[PDWordGetNthCharStyle](#)

Attributes

Get	Set
PDStyleGetColor	—
PDStyleGetFont	—
PDStyleGetFontSize	—



PDText

A graphic object representing one or more character strings on a page in a PDF file. Like paths, text can be stroked, filled, and/or serve as a clipping path.

Enumerating

[PDTextEnum](#)

Attributes

Get	Set
PDTextGetState	—

PDTextAnnot

A PDF text annotation on a page in a PDF file. You can use any **PDAnnot** method on a **PDTextAnnot**.

Applications can:

- Get and set attributes including the rectangle, textual contents, and whether the annotation is open.
- Create new text annotations and delete or move existing ones using **PDAnnot** methods.
- Manipulate the behavior of text annotations by modifying the Text Annotation Handler.

Obtaining

Any of the **PDAnnot** calls, followed by **CastToPDTextAnnot**. The annotation passed to **CastToPDTextAnnot** must be a text annotation, it will not convert other annotation types into text annotations.

Disposing

PDPageRemoveAnnot

Attributes

Get	Set
PDTextAnnotGetContents	PDTextAnnotSetContents
PDTextAnnotIsOpen	PDTextAnnotSetOpen
AVPageViewGetSelectedAnnotPageNum	—
—	AVPageViewSetAnnotLocation
PDAnnotEqual	—
PDAnnotGetColor	PDAnnotSetColor
PDAnnotGetDate	PDAnnotSetDate
PDAnnotGetFlags	PDAnnotSetFlags
PDAnnotGetRect	PDAnnotSetRect
PDAnnotGetSubtype	—
PDAnnotGetTitle	PDAnnotSetTitle



Cos conversion

[PDAnnotGetCosObj](#)
[PDAnnotFromCosObj](#)

Validity testing

[PDAnnotIsValid](#)

PDTextSelect

A selection of text on a single page, and may contain more than one disjoint group of words. A text selection is specified by one or more *ranges* of text, with each range containing the word numbers of the selected words. Each range specifies a start and end word, where “start” is the first of a series of selected words and “end” is the first word *not* in the series.

Obtaining

[AVDocGetSelection](#)
[AVPageViewTrackText](#)
[PDDocCreateTextSelect](#)
[PDTextSelectCreatePageHilite](#)
[PDTextSelectCreatePageHiliteEx](#)
[PDTextSelectCreateWordHilite](#)
[PDTextSelectCreateWordHiliteEx](#)
[PDTextSelectCreateRanges](#)
[PDTextSelectCreateRangesEx](#)

Disposing

[PDTextSelectDestroy](#)

Enumerating

[PDTextSelectEnumQuads](#)
[PDTextSelectEnumText](#)

Attributes

Get	Set
—	PDTextSelectCreatePageHilite PDTextSelectCreatePageHiliteEx
—	PDTextSelectCreateRanges PDTextSelectCreateRangesEx
—	PDTextSelectCreateWordHilite PDTextSelectCreateWordHiliteEx
PDTextSelectGetBoundingRect	PDTextSelectGetBoundingRect
PDTextSelectGetPage	—
PDTextSelectGetRange	—
PDTextSelectGetRangeCount	—



Declarations

`PDTextSelectRange`



PDThread

An article in the Acrobat viewer's user interface, and contains an ordered sequence of rectangles that bound the article. Each rectangle is called a *bead*. Threads can be created interactively by the user, or programmatically.

Obtaining

[PDDocGetThread](#)
[PDThreadNew](#)
[PDThreadFromCosObj](#)
[PDBeadGetThread](#)

Disposing

[PDDocRemoveThread](#)
[PDThreadDestroy](#)

Attributes

Get	Set
PDThreadGetFirstBead	PDThreadSetFirstBead
PDThreadGetInfo	PDThreadSetInfo
—	PDBeadInsert

Cos conversion

[PDThreadGetCosObj](#)
[PDThreadFromCosObj](#)

Validity testing

[PDThreadIsValid](#)



PDThumb

A thumbnail preview image of a page.

Obtaining

[PDDocCreateThumbs](#)

Disposing

[PDDocDeleteThumbs](#)

Declarations

[PDThumbCreationServer](#)



PDTrans

A transition to a page. The **Trans** key in a Page dictionary specifies a Transition dictionary, which describes the effect to use when going to a page and the amount of time the transition should take.

Obtaining

[PDPageGetTransition](#)
[PDTransFromCosObj](#)
[PDTransNew](#)
[PDTransNewFromCosDoc](#)
[PDTransNull](#)

Attributes

Get	Set
PDTransEqual	—
PDTransGetDuration	—
PDTransGetSubtype	—

Cos conversion

[PDTransFromCosObj](#)
[PDTransGetCosObj](#)

Validity testing

[PDTransIsValid](#)

Declarations

[Transition Duration](#)



PDViewDestination

A particular view of a page in a document. It contains a reference to a page, a rectangle on that page, and information specifying how to adjust the view to fit the window's size and shape. It corresponds to a PDF Dest array and can be considered a special form of a **PDAction**.

Obtaining

- AVPageViewToViewDest
- PDActionGetDest
- PDViewDestCreate
- PDViewDestFromCosObj
- PDViewDestResolve

Disposing

- PDViewDestDestroy

Attributes

Get	Set
PDViewDestGetAttr	—

Cos Conversion

- PDViewDestFromCosObj
- PDViewDestGetCosObj

Validity testing

- PDViewDestIsValid



PDWord

A word in a PDF file. Each word contains a sequence of characters in one or more styles (see [PDStyle](#)).

Obtaining

[PDWordFinderGetNthWord](#)

[PDWordFinderEnumWords](#)

Enumerating

[PDWordFinderEnumWords](#)

Attributes

Get	Set
PDWordGetAttr	—
PDWordGetCharacterTypes	—
PDWordGetCharDelta	—
PDWordGetCharOffset	—
PDWordGetCharOffsetEx	—
PDWordGetCharQuad	—
PDWordGetCharEncFlags	—
PDWordGetLength	—
PDWordGetNthCharStyle	—
PDWordGetNthQuad	—
PDWordGetNumQuads	—
PDWordGetNumHiliteChar	—
PDWordGetByteIdxFromHiliteChar	—
PDWordGetString	—
PDWordGetStyleTransition	—
PDWordIsCurrentlyVisible	PDWordMakeVisible

Declarations

[Word Attributes](#)



PDWordFinder

Extracts words from a PDF file, and enumerates the words on a single page or on all pages in a document.

Obtaining

[PDDocCreateWordFinderEx](#)
[PDDocCreateWordFinder](#)
[PDDocCreateWordFinderUCS](#)
[PDDocGetWordFinder](#)

Disposing

[PDWordFinderDestroy](#)

Enumerating

[PDWordFinderEnumWords](#)
[PDWordFinderEnumVisibleWords](#)

Attributes

Get	Set
PDWordFinderAcquireWordList	PDWordFinderReleaseWordList
PDWordFinderGetLatestAlgVersion	—
PDWordFinderGetNthWord	—

Declarations

[WordFinder Character Encoding Flags](#)
[WordFinder Sort Order Flags](#)



PDXObject

A superclass used for PDF XObjects. Acrobat currently uses two XObject subclasses: [PDImage](#) and [PDForm](#). You can use any **PDXObject** method on these three objects.

Enumerating

[PDXObjectEnumFilters](#)
[PDXObjectGetData](#)

Attributes

Get	Set
PDXObjectGetData	—
PDXObjectGetDataLength	—
PDXObjectGetSubtype	—

Cos Conversion

[PDXObjectGetCosObj](#)

PDFEdit

Provides easy access to PDF page contents. With PDFEdit, you can treat a page's contents as a list of objects—rather than having to manipulate the content stream's PDF marking operators.

The PDFEdit API is meant to be used in conjunction with the Acrobat PModel and Cos APIs for manipulating PDF documents.

NOTE: The PDFEdit API is not available in Adobe Reader.

PDEBeginContainer

The PDFEdit representation of the opening bracket of a marked-content sequence. Elements of this type must be paired with elements of type [PDEEndContainer](#).

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclass of

[PDEElement](#)

Obtaining

[PDEBeginContainerCreate](#)

Disposing

[PDERelease](#)

Attributes

Get	Set
PDEBeginContainerGetDict	PDEBeginContainerSetDict
PDEBeginContainerGetMC_Tag	PDEBeginContainerSetMC_Tag



PDEBeginGroup

A group of **PDEElements** on a page in a PDF file.

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclass of

[PDEElement](#)

Obtaining

[PDEBeginGroupCreate](#)

Disposing

[PDERelease](#)

PDEClip

A list of **PDEElements** containing a list of **PDEPaths** and **PDETexts** that describe a clip state. **PDEClips** can be created and built up with **PDEClip** methods. Any **PDEElement** object can have **PDEClip** associated with it.

PDEClip objects can contain **PDEContainers** and **PDEGroups** to an arbitrary level of nesting. This allows **PDEContainers** to be used to mark clip objects.

PDEGroups inside **PDEClips** that contain at least one **PDEText** and no **PDEPaths** have a special meaning. All **PDEText** objects contained in such a **PDEGroup** are considered to be part of the same **BT/ET** block. This means that the union of these **PDETexts** makes up a single clipping path—as opposed to the *intersection* of the **PDETexts**.

NOTE: The PDFEdit API is not available in Adobe Reader.

Obtaining

[PDEClipCreate](#)
[PDEElementGetClip](#)

Disposing

[PDERelease](#)

Enumerating

[PDEClipFlattenedEnumElems](#)

Attributes

Get	Set
PDEClipGetElem	—
PDEClipGetNumElems	—
—	PDEClipAddElem
—	PDEClipRemoveElems



PDEColorSpace

A reference to a color space used on a page in a PDF file. The color space is part of the graphics state attributes of a **PDEElement**.

NOTE: The PDFEdit API is not available in Adobe Reader.

Obtaining

[PDEColorSpaceCreate](#)
[PDEColorSpaceCreateFromName](#)
[PDEImageGetColorSpace](#)

Disposing

[PDERelease](#)

Attributes

Get	Set
PDEColorSpaceGetBase	—
PDEColorSpaceGetBaseNumComps	—
PDEColorSpaceGetCTable	—
PDEColorSpaceGetHiVal	—
PDEColorSpaceGetName	—
PDEColorSpaceGetNumComps	—

Cos conversion

[PDEColorSpaceCreate](#)
[PDEColorSpaceGetCosObj](#)

Declarations

[PDEColorSpec](#)
[PDEColorValue](#)



PDEContainer

A group of **PDEElements** on a page in a PDF file. In the PDF file, containers are delimited by Marked Content **BMC/EMC** or **BDC/EMC** pairs. Every **PDEContainer** has a Marked Content tag associated with it. In addition to grouping a set of elements, a **BDC/EMC** pair specifies a property list to be associated with the grouping. Thus a **PDEContainer** corresponding to a **BDC/EMC** pair also has a property list dictionary associated with it.

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclass of

[PDEElement](#)

Obtaining

[PDEContainerCreate](#)
[PDSMCGetPDEContainer](#)

Disposing

[PDERelease](#)

Attributes

Get	Set
PDEContainerGetContent	PDEContainerSetContent
PDEContainerGetDict	PDEContainerSetDict
PDEContainerGetMCTag	PDEContainerSetMCTag



PDEContent

Contains the modifiable contents of a **PDPPage**. A **PDEContent** may be obtained from an existing page or from a Form **XObject** or from a Type 3 CharProc. You can create an empty **PDEContent**. A **PDEContent** contains **PDEElements**. In addition, a **PDEContent** may have attributes such as Form matrix and **setcachedevice** parameters.

NOTE: The PDFEdit API is not available in Adobe Reader.

Obtaining

[PDEContentCreate](#)
[PDEContainerGetContent](#)
[PDEContentCreateFromCosObj](#)
[PDEFormGetContent](#)
[PDPageAcquirePDEContent](#)

Disposing

[PDERelease](#)

Attributes

Get	Set
PDEContentGetAttrs	—
PDEContentGetElem	—
PDEContentGetNumElems	—
PDEContentGetResources	—
—	PDEContentAddElem PDEContentRemoveElem
—	PDEContentFlattenOC

Cos conversion

[PDEContentCreateFromCosObj](#)
[PDEContentToCosObj](#)

Declarations

[PDEContentAttrs](#)
[PDEContentFlags](#)
[PDEContentToCosObjFlags](#)



PDEDeviceNCOLORS

A color space with a variable number of device-dependent components. Usually used to store multiple spot colors in a single color space.

NOTE: The PDFEdit API is not available in Adobe Reader.

Obtaining

[PDEDeviceNCOLORSCreate](#)

Attributes

Get	Set
PDEDeviceNCOLORSGetColorValue	—



PDEElement

The base class for elements of a page display list (**PDEContent**) and for clip objects. The general **PDEElement** methods allow you to get and set general element properties.

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclasses

[PDEContainer](#)
[PDEForm](#)
[PDEGroup](#)
[PDEImage](#)
[PDEPath](#)
[PDEPlace](#)
[PDEPS](#)
[PDEShading](#)
[PDEText](#)
[PDEUnknown](#)
[PDEXObject](#)

Obtaining

[PDEClipGetElem](#)
[PDEContentGetElem](#)
[PDEElementCopy](#)

Disposing

[PDERelease](#)

Attributes

Get	Set
PDEElementGetBBox	—
PDEElementGetClip	—
PDEElementGetGState	PDEElementSetGState
PDEElementGetMatrix	PDEElementSetMatrix
PDEElementGetOCMD	PDEElementRemoveOCMD PDEElementSetOCMD
PDEElementIsAtPoint	—
PDEElementIsAtRect	—



Get	Set
PDEElementIsCurrentlyVisible	PDEElementMakeVisible
PDEElementGetAllVisibilities	

Declarations

[PDEElementCopyFlags](#)
[PDEEnumElementsFlags](#)
[PDEGraphicStateP](#)
[PDEGraphicStateWasSetFlags](#)



PDEndContainer

The PDFEdit representation of the closing bracket of a marked-content sequence. Elements of this type must be paired with elements of type [PDEBeginContainer](#).

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclass of

[PDEElement](#)

Obtaining

[PDEndContainerCreate](#)

Disposing

[PDERelease](#)



PDEndGroup

A group of **PDElements** on a page in a PDF file.

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclass of

[PDElement](#)

Obtaining

[PDEndGroupCreate](#)

Disposing

[PDERelease](#)

PDEExtGState

A reference to an **ExtGState** resource used on a page in a PDF file. It specifies a **PDEElement**'s extended graphics state, which is part of its graphics state.

NOTE: The PDFEdit API is not available in Adobe Reader.

Obtaining

[PDEExtGStateCreate](#)

Disposing

[PDERelease](#)

Attributes

Get	Set
PDEExtGStateGetAIS	PDEExtGStateSetAIS
PDEExtGStateGetBlendMode	PDEExtGStateSetBlendMode
PDEExtGStateGetOpacityFill	PDEExtGStateSetOpacityFill
PDEExtGStateGetOpacityStroke	PDEExtGStateSetOpacityStroke
PDEExtGStateGetOPFill	PDEExtGStateSetOPFill
PDEExtGStateGetOPM	PDEExtGStateSetOPM
PDEExtGStateGetOPStroke	PDEExtGStateSetOPStroke
PDEExtGStateGetSA	PDEExtGStateSetSA
—	PDEExtGStateSetSoftMask
PDEExtGStateGetTK	PDEExtGStateSetTK

Cos conversion

[PDEExtGStateGetCosObj](#)

PDEFont

A reference to a font used on a page in a PDF file. It may be equated with a font in the system. A **PDEFont** is not the same as a **PDFont**; a **PDFont** is associated with a particular document.

NOTE: The PDFEdit API is not available in Adobe Reader.

Obtaining

[PDEFontCreate](#)
[PDEFontCreateFromCosObj](#)
[PDEFontCreateFromSysFont](#)
[PDEFontCreateFromSysFontEx](#)
[PDEFontCreateWithParams](#)
[PDTextGetFont](#)

Disposing

[PDERelease](#)

Attributes

Get	Set
PDEFontCreateWithParams	—
—	PDEFontEmbedNow PDEFontEmbedNowDontSubset
PDEFontGetCreateNeedFlags	—
PDEFontGetNumCodeBytes	—
PDEFontGetOneByteEncoding	—
PDEFontGetSysEncoding	PDEFontSetSysEncoding
PDEFontGetSysFont	PDEFontSetSysFont
PDEFontGetWidths	—
PDEFontGetWidthsNow	—
PDEFontIsEmbedded	—
PDEFontIsMultiByte	—
—	PDEFontSubsetNow
PDEFontSumWidths	—

**Cos conversion**

[PDFontCreateFromCosObj](#)

[PDFontGetCosObj](#)

Declarations

[PDFontAttrs](#)

[PDFontCreateFlags](#)

[PDFontInfoP](#)

PDEForm

A **PDEElement** that corresponds to an instance of an **XObject** Form on a page (or other containing stream such as another **XObject** Form or annotation form). The context associated with this instance includes the actual **CosObj** stream that represents the **XObject** Form and the initial conditions of the graphics state. The latter consists of the transformation matrix, initial color values, and so forth. It is possible to have two **PDEForms** that refer to the same **XObject** Form. The forms will exist at different places on the same page, depending on the transformation matrix. They may also have different colors or line stroking parameters. In the case of a transparency group, the opacity is specified in the **gstate**.

Within a **PDEForm**, each **PDEElement** has it's own **gstate** (or is a container, place, or group object). These **gstates** are independent of the parent **PDEForm** **gstate**. **PDEForm** elements within the **PDEForm** may have their own opacity.

A **PDEContent** may be obtained from a **PDEForm** to edit the form's display list.

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclass of

[PDEElement](#)

Obtaining

[PDEFormCreateFromCosObj](#)
[PDEFormCreateClone](#)

Disposing

[PDERelease](#)

Attributes

Get	Set
PDEFormGetContent	PDEFormSetContent
—	PDEFormSetXGroup

Cos conversion

[PDEFormCreateFromCosObj](#)
[PDEFormGetCosObj](#)



PDEGroup

An in-memory representation of objects in a **PDEContent**. It has no state and is not represented in any way in a PDF content stream (that is, **PDEContent**).

When used in a **PDEClip**, this object is used to associate **PDEText** objects into a single clipping object.

NOTE: The PDFEdit API is not available in Adobe Reader.

Obtaining

[PDEGroupCreate](#)

Attributes

Get	Set
PDEGroupGetContent	PDEGroupSetContent

PDEImage

A **PDEElement** that contains an Image **XObject** or in-line image. You can associate data or a stream with an image.

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclass of

[PDEElement](#)

Obtaining

[PDEImageCreate](#)

[PDEImageCreateFromCosObj](#)

Disposing

[PDERelease](#)

Attributes

Get	Set
PDEImageDataIsEncoded	—
PDEImageGetAttrs	—
PDEImageGetColorSpace	PDEImageSetColorSpace
PDEImageGetData	PDEImageSetData
PDEImageGetDataLen	—
PDEImageGetDataStm	PDEImageSetDataStm
PDEImageGetFilterArray	—
PDEImageGetMatteArray	PDEImageSetMatteArray
PDEImageGetSMask	PDEImageSetSMask
PDEImageIsCosObj	—

Cos conversion

[PDEImageCreateFromCosObj](#)

[PDEImageGetCosObj](#)

Declarations

[PDEImageAttrFlags](#)

[PDEImageAttrs](#)

[PDEImageDataFlags](#)

PDEObject

The abstract super class of **PDFEdit** classes. You can find the type of any object with the **PDEObjectGetType** method. You can then cast and apply that class's methods to the object. In addition, you can cast any of the **PDFEdit** objects to a **PDEObject** and use it anywhere a **PDEObject** is called for, such as in the **PDEObject** methods.

NOTE: The PDFEdit API is not available in Adobe Reader.

Obtaining

Various since all **PDFEdit** objects are **PDEObjects**.

Disposing

[PDERelease](#)

Enumerating

[PDEObjectDump](#)

Attributes

Get	Set
PDEGetTag	PDEAddTag PDERemoveTag
PDEObjectGetType	—
—	PDEAcquire
—	PDERelease

Declarations

[PDEType](#)



PDEPath

A **PDEElement** that contains a path. Path objects can be stroked, filled, and/or serve as a clipping path.

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclass of

[PDEElement](#)

Obtaining

[PDEPathCreate](#)

Disposing

[PDERelease](#)

Attributes

Get	Set
PDEPathGetData	PDEPathSetData
PDEPathGetPaintOp	PDEPathSetPaintOp
—	PDEPathAddSegment

Declarations

[PDEPathElementType](#)

[PDEPathOpFlags](#)



PDEPattern

A reference to a Pattern resource used on a page in a PDF file.

NOTE: The PDFEdit API is not available in Adobe Reader.

Obtaining

[PDEPatternCreate](#)

Disposing

[PDERelease](#)

Cos conversion

[PDEPatternGetCosObj](#)

PDEPlace

A **PDEElement** that marks a place on a page in a PDF file. In a PDF file, a place is represented by the **MP** or **DP** Marked Content operators.

Marked content is useful for adding structure information to a PDF file. For instance, a drawing program may want to mark a point with information, such as the start of a path of a certain type. Marked content provides a way to retain this information in the PDF file. A **DP** operator functions the same as the **MP** operator and, in addition, allows a property list dictionary to be associated with a place.

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclass of

[PDEElement](#)

Obtaining

[PDEPlaceCreate](#)

Disposing

[PDERelease](#)

Attributes

Get	Set
PDEPlaceGetDict	PDEPlaceSetDict
PDEPlaceGetMCTag	PDEPlaceSetMCTag



PDEPS

Element representing in-line or XObject pass-through PostScript object. XObject PostScripts are listed in page XObject resources.

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclass of

[PDEElement](#)

Obtaining

[PDEPSCreate](#)

[PDEPSCreateFromCosObj](#)

Attributes

Get	Set
PDEPSGetAttrs	—
PDEPSGetData	PDEPSSetData
PDEPSGetDataStm	PDEPSSetDataStm

Cos conversion

[PDEPSCreateFromCosObj](#)



PDEShading

A **PDEElement** that represents smooth shading.

NOTE: The PDFEdit API is not available in Adobe Reader.

Obtaining

[PDEShadingCreateFromCosObj](#)

Cos conversion

[PDEShadingCreateFromCosObj](#)

[PDEShadingGetCosObj](#)

PDESoftMask

Object for creating and manipulating a soft mask in a PDF file.

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclass of

[PDEElement](#)

Obtaining

[PDESoftMaskCreate](#)

[PDESoftMaskCreateFromCosObj](#)

[PDESoftMaskCreateFromName](#)

Disposing

[PDERelease](#)

Attributes

Get	Set
PDESoftMaskAcquireForm	—
PDESoftMaskGetBackdropColor	PDESoftMaskSetBackdropColor
PDESoftMaskGetCosObj	—
PDESoftMaskGetName	—
PDESoftMaskGetTransferFunction	PDESoftMaskSetTransferFunction
—	PDESoftMaskSetXGroup

Cos conversion

[PDESoftMaskGetCosObj](#)

[PDESoftMaskCreateFromCosObj](#)

PDEText

A **PDEElement** representing text. It is a container for text as show strings or as individual characters. Each sub-element may have different graphics state properties. However, the same clip applies to all sub-elements of a **PDEText**. Also, the **charpath** of a **PDEText** can be used to represent a clip.

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclass of

[PDEElement](#)

Obtaining

[PDETextCreate](#)

Disposing

[PDERelease](#)

Attributes

Get	Set
PDETextGetAdvance	—
PDETextGetAdvanceWidth	—
PDETextGetBBox	—
PDETextGetFont	PDETextRunSetFont
PDETextGetGState	PDETextRunSetGState
PDETextGetItem	PDETextAddItem PDETextRemoveItems
PDETextGetMatrix	—
PDETextGetNumBytes	—
PDETextGetNumRuns	—
PDETextGetQuad	—
PDETextGetRunForChar	—
PDETextGetStrokeMatrix	PDETextRunSetStrokeMatrix
PDETextGetState	PDETextRunSetState
PDETextGetText	—

Get	Set
PDETextGetTextMatrix	PDETextRunSetTextMatrix
PDETextGetTextState	PDETextRunSetTextState
PDETextRunGetCharOffset	—
PDETextRunGetNumChars	—
—	PDETextRunSetMatrix
—	PDETextRunSetState
—	PDETextRunSetMatrix
—	PDETextRunSetStrokeMatrix
PDETextIsAtPoint	—
—	PDETextAdd
—	PDETextIsAtPoint
—	PDETextReplaceChars
—	PDETextSplitRunAt

Declarations

[PDEGraphicStateP](#)
[PDEGraphicStateWasSetFlags](#)
[PDETextFlags](#)
[PDETextRenderMode](#)
[PDETextState](#)
[PDETextStateWasSetFlags](#)



PDETextItem

A **PDEElement** representing a text object.

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclass of

[PDEElement](#)

Obtaining

[PDETextItemCreate](#)

[PDETextItemGetItem](#)

Attributes

Get	Set
PDETextItemCopyText	—
PDETextItemGetFont	PDETextItemSetFont
PDETextItemGetGState	PDETextItemSetGState
PDETextItemGetTextLen	—
PDETextItemGetTextMatrix	PDETextItemSetTextMatrix
PDETextItemGetTextState	PDETextItemSetTextState
—	PDETextItemRemoveChars PDETextItemReplaceChars PDETextItemReplaceText



PDEUnknown

A **PDEElement** representing an unknown element.

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclass of

[PDEElement](#)

Attributes

Get	Set
PDEUnknownGetOpName	—



PDEXGroup

A transparency (**XGroup**) resource.

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclass of

[PDEElement](#)

Obtaining

[PDEXGroupCreate](#)

[PDEXGroupCreateFromCosObj](#)

Disposing

[PDERelease](#)

Attributes

Get	Set
PDEXGroupAcquireColorSpace	—
PDEXGroupGetCosObj	—
PDEXGroupGetIsolated	PDEXGroupSetIsolated
PDEXGroupGetKnockout	PDEXGroupSetKnockout
—	PDEXGroupSetColorSpace

Cos conversion

[PDEXGroupGetCosObj](#)

[PDEXGroupCreateFromCosObj](#)



PDEXObject

A **PDEElement** representing an arbitrary **XObject**.

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclass of

[PDEElement](#)

Obtaining

[PDEXObjectCreate](#)

Disposing

[PDERelease](#)

Cos conversion

[PDEXObjectGetCosObj](#)



PDFSysEncoding

A **PDFElement** that provides system encoding for a PDF file.

NOTE: The PDFEdit API is not available in Adobe Reader.

Subclass of

[PDFElement](#)

Obtaining

[PDFSysEncodingCreateFromBaseName](#)

[PDFSysEncodingCreateFromCMapName](#)

[PDFSysEncodingCreateFromCodePage](#)

[PDFFontGetSysEncoding](#)

Disposing

[PDFRelease](#)

Attributes

Get	Set
PDFSysEncodingGetWMode	—



PDSysFont

A reference to a font installed in the host system. **PDSysFont** methods allow you to list the fonts available in the host system and to find a font in the system that matches a **PDEFont**, if it is present.

NOTE: The PDFEdit API is not available in Adobe Reader.

Obtaining

- [PDEnumSysFonts](#)
- [PDFindSysFont](#)
- [PDFindSysFontEx](#)
- [PDFindSysFontForPDEFont](#)
- [PDEFontGetSysFont](#)

Enumerating

- [PDEnumSysFonts](#)

Attributes

Get	Set
PDSysFontAcquirePlatformData	PDSysFontReleasePlatformData
PDSysFontGetAttrs	—
PDSysFontGetCIDSystemInfo	—
PDSysFontGetCreateFlags	—
PDSysFontGetEncoding	
PDSysFontGetInfo	—
PDSysFontGetName	—
PDSysFontGetType0Widths	—
PDSysFontGetWidths	
PDSysFontGetWidthsEx	
—	PDEmbedSysFontForPDEFont

Declarations

- [PDSysFontMatchFlags](#)



PDSEdit

The creation and manipulation of logical structure in PDF documents.

NOTE: The write functions in the PDSEdit API are not available in Adobe Reader.

PDSAttrObj

Represents PDF logical structure attribute objects, which are dictionaries containing application-specific data that can be attached to **PDSElements**.

NOTE: The write functions in the PDSEdit API are not available in Adobe Reader.

Obtaining

[PDSAttrObjCreate](#)
[PDSAttrObjCreateFromStream](#)
[PDSClassMapGetAttrObj](#)
[PDSElementGetAttrObj](#)

Disposing

[PDSElementRemoveAttrObj](#)
[PDSElementRemoveAllAttrObjs](#)

Attributes

Get	Set
PDSAttrObjGetCosObj	—
PDSAttrObjGetOwner	—

PDSClassMap

Associates class identifiers, which are names, with objects of type **PDSAttrObj**. Structural elements maintain a list of names identifying classes to which they belong. The associated attributes are thus shared by all structural elements belonging to a given class. There is one class map per document, associated with the **PDSTreeRoot**.

NOTE: The write functions in the PDSEdit API are not available in Adobe Reader.

Obtaining

[PDSTreeRootCreateClassMap](#)
[PDSTreeRootGetClassMap](#)

Disposing

[PDSTreeRootRemoveClassMap](#)

Attributes

Get	Set
—	PDSClassMapAddAttrObj PDSClassMapRemoveAttrObj
PDSClassMapGetAttrObj PDSClassMapGetNumAttrObjs	—



PDSElement

Represents PDF structural elements, which are nodes in a tree giving a PDF document's logical structure.

NOTE: The write functions in the PDSEdit API are not available in Adobe Reader.

Obtaining

[PDSElementCreate](#)
[PDSElementGetParent](#)
[PDSElementGetInfo](#)
[PDSElementGetParent](#)
[PDSElementGetElementFromID](#)

Attributes

Get	Set
—	PDSElementAddAttrObj PDSElementRemoveAttrObj PDSElementRemoveAllAttrObjs
—	PDSElementAddClass PDSElementRemoveClass PDSElementRemoveAllClasses
PDSElementGetActualText	PDSElementSetActualText
PDSElementGetAlt	PDSElementSetAlt
PDSElementGetAttrObj PDSElementGetNumAttrObjs	—
PDSElementGetClass PDSElementGetNumClasses	—
PDSElementGetCosObj	—
PDSElementGetFirstPage	—
PDSElementGetID	PDSElementClearID PDSElementSetID



Get	Set
PDSElementGetKid	PDSElementInsertKid
PDSElementGetKidEx	PDSElementInsertMCAsKid
PDSElementGetKidWithMCInfo	PDSElementInsertOBJAsKid
PDSElementGetNumKids	PDSElementInsertStmMCAsKid
PDSElementGetParent	PDSElementReplaceKid
	PDSElementRemoveKid
	PDSElementRemoveKidOBJ
	PDSElementRemoveKidMC
	PDSElementReplaceKidMC
PDSElementGetLanguage	PDSElementSetLanguage
PDSElementGetRevision	PDSElementIncrementRevision
PDSElementGetTitle	PDSElementSetTitle
PDSElementGetType	PDSElementSetType



PDSMC

Represents marked content—portions of the graphic content of a PDF document that may be included in the document's logical structure hierarchy. This type is identical with the PDFEdit layer type **PDEContainer**.

NOTE: The write functions in the PDSEdit API are not available in Adobe Reader.

Attributes

Get	Set
PDSMCGetInfo	—
PDSMCGetParent PDSMCIDGetParent	—
PDSMCGetPDEContainer	—

Declarations

[PDSMCInfo](#)

PDSRoleMap

Represents mappings of structural element types present in a PDF document to standard element types having similar uses. There is one **PDSClassMap** per document, associated with the **PDSTreeRoot**.

NOTE: The write functions in the PDSEdit API are not available in Adobe Reader.

Obtaining

[PDSRoleMapCopy](#)
[PDSTreeRootCreateRoleMap](#)
[PDSTreeRootGetRoleMap](#)

Disposing

[PDSTreeRootRemoveRoleMap](#)

Attributes

Get	Set
PDSRoleMapCopy	—
PDSRoleMapGetDirectMap	—
PDSRoleMapDoesMap	PDSRoleMapMap PDSRoleMapUnMapDst PDSRoleMapUnMapSrc

PDSTreeRoot

The root of the structure tree, which is a central repository for information related to a PDF document's logical structure. There is at most one **PDSTreeRoot** in each document.

NOTE: The write functions in the PDSEdit API are not available in Adobe Reader.

Obtaining

[PDDocCreateStructTreeRoot](#)
[PDDocGetStructTreeRoot](#)

Disposing

[PDDocRemoveStructTreeRoot](#)

Attributes

Get	Set
PDSTreeRootGetClassMap	PDSTreeRootRemoveClassMap
PDSTreeRootGetElementFromID	—
PDSTreeRootGetKid	PDSTreeRootInsertKid PDSTreeRootRemoveKid
PDSTreeRootGetNumKids	—
PDSTreeRootGetRoleMap	PDSTreeRootRemoveRoleMap
—	PDSTreeRootReplaceKid
—	PDSTreeRootReplaceStreamRef



Methods

Methods are organized into sections for each of the Acrobat code layers. Within layers, methods are organized by object. Within each object, methods are listed alphabetically.

NOTE: Method descriptions list exceptions that can be raised by each method. However, this list is not necessarily complete, and in general, any method can raise an exception.

AS Layer Methods	The Acrobat Support (AS) layer provides a variety of utility methods, including platform-independent memory allocation and fixed-point math utilities.
AV Layer Methods	The Acrobat Viewer (AV) layer deals with the viewer's user interface.
Cos Layer Methods	The Cos Object System (Cos) layer provides access to the low-level building blocks of PDF files.
PD Layer Methods	The Portable Document (PD) layer provides access to components of PDF documents.
PDFEdit Methods	The PDFEdit layer provides access to PDF page contents associated with the Contents entry in the page's dictionary. These methods are not available in Adobe Reader.
PDSEdit Methods	The PDSEdit layer provides access to the logical structure of a PDF document. The methods that write are not available in the Adobe Reader.
Macintosh-specific Methods UNIX-specific Methods Windows-specific Methods	These methods apply to and are available for only a particular operating system.



AS Layer Methods

General

ASAtom

ASCab

ASCallback

ASDate

ASExtension

ASFile

ASFileSys

ASPlatformPath

ASStm

ASText

ASTimeSpan

ASUUID

Errors

Fixed Point Math

HFT and HFTServer

Memory Allocation

String Encoding

General

These methods perform general AS-level operations that are not associated with particular object types.

ASGetConfiguration

```
void* ASGetConfiguration (ASAtom key);
```

Description

Gets information about the Acrobat viewer application under which the plug-in is running. Use this method if your plug-ins functionality depends on which Acrobat viewer it is running under.

The method can return a product name, or check whether the current product allows editing. Do not rely on the product name to determine whether the product can edit files, as product names and feature sets may vary; use the **CanEdit** selector to do this.

Parameters

key	The key determines whether the method tests editability, or finds which product configuration is running. Values are: ● CanEdit:—Checks whether editing is allowed in the current environment (regardless of the product name). ● Product:—Checks which Acrobat application is running.

Return Value

The return value’s type depends on the request key. Cast the return value to the type you are expecting, based on the key you pass in:

- For the key **CanEdit**, returns an **ASBool** value, **true** if the current application allows editing, **false** otherwise.
- For the key **Product**, returns a **const char*** value, one of the following strings:

“Reader”	Adobe Reader
“Exchange”	Adobe Acrobat Standard
“Exchange-Pro”	Adobe Acrobat Professional
“Acrobat PDF Library”	Adobe Acrobat PDF Library

Known Exceptions

If an unknown value is passed as **key**, the value **UNDEFINED_CONFIGURATION_SELECTOR** is returned (see `CoreExpT.h`).

**Header File**

CorCalls.h

Example

```
prodName = ASGetConfiguration(  
    ASAtomFromString("Product"))
```

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



ASGetSecs

```
ASUns32 ASGetSecs (void);
```

Description

Returns the number of seconds elapsed since midnight, January 1, 1970, coordinated universal time, up to the current time, using Greenwich mean time (GMT).

Parameters

None

Return Value

See above.

Header File

ASCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



ASAtom

ASAtomExistsForString

```
ASBool ASAtomExistsForString (const char* str, ASAtom* atom);
```

Description

Tests whether an **ASAtom** exists for the specified string.

Parameters

sStr	The string to test.
atom	<i>(Filled by the method, may be NULL)</i> If the ASAtom corresponding to nameStr already exists, it is returned in atom . Pass NULL to simply check whether the ASAtom already exists.

Return Value

true if an **ASAtom** already exists for **str**, **false** otherwise.

Header File

CorCalls.h

Related Methods

[ASAtomFromString](#)

[ASAtomGetCount](#) (Only available with PDF Library SDK)

[ASAtomGetString](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

ASAtomFromString

```
ASAtom ASAtomFromString (const char* str);
```

Description

Gets the **ASAtom** for the specified string. You can also use this method to create an **ASAtom**, since it creates one for the string if one does not already exist.

If an **ASAtom** already exists for **str**, the existing **ASAtom** is returned. Thus **ASAtoms** may be compared for equality of the underlying string.

Because **ASAtoms** cannot be deleted, they are useful for strings that are used many times in an Acrobat viewer session, but are not advisable for strings that have a short lifetime. For the same reason, it is not a good idea to create large numbers of **ASAtoms**.

Parameters

str	The string for which an ASAtom is created.

Return Value

The **ASAtom** corresponding to **str**.

Header File

CorCalls.h

Related Methods

[ASAtomExistsForString](#)

[ASAtomGetCount](#) (Only available with PDF Library SDK)

[ASAtomGetString](#)

Example

```
/* Compare ASAtoms for equality */
PDAnnot annot;

if (PDAnnotGetSubtype(annot) == ASAtomFromString("Link")) {
    ...
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

ASAtomGetString

```
const char* ASAtomGetString (ASAtom atom);
```

Description

Gets the string associated with the specified **ASAtom**.

Parameters

atom	The ASAtom whose string is obtained.

Return Value

The string corresponding to **atom**. Returns an empty string if **atom == ASAtomNull** or **NULL** if the **atom** has not been defined.

Header File

CorCalls.h

Related Methods

[ASAtomExistsForString](#)
[ASAtomFromString](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



ASCab

ASCabCopy

```
void ASCabCopy (ASCab srcCab, ASCab dstCab);
```

Description

For each key-value pair in **srcCab** a copy of the key-value pair will be placed into **dstCab**, possibly overwriting any identically named key-value pair in **dstCab**. If the value being copied is a pointer with an associated “destroyProc”, the pointer and its type string, but not the data it points to, will be copied and an internal reference count incremented.

Parameters

srcCab	The source cabinet.
dstCab	The destination cabinet.

Return Value

None

Known Exceptions

[genErrBadParm](#)
[genErrNoMemory](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabDup](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



ASCabDestroy

```
void ASCabDestroy (ASCab theCab);
```

Description

Destroys the cabinet and all its key-value pairs. This method raises if the cabinet is the value for some key in another cabinet.

Parameters

theCab	The cabinet.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

ASCabDestroyEmpties

```
void ASCabDestroyEmpties (ASCab theCab, ASBool recurse);
```

Description

Finds any empty cabinets in **theCab**, removes their corresponding keys, and destroys them.

Parameters

theCab	The cabinet.
recurse	true to recurse through all sub-cabinets inside theCab ; false to limit enumeration to key-value pairs directly inside theCab .

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabDetachBinary

```
void* ASCabDetachBinary (ASCab theCab, const char* theKey,
    ASTArraySize* numBytes);
```

Description

Retrieves the binary object stored under **theKey** in **theCab** and removes the key from **theCab**.

The client assumes ownership of the object and is responsible for de-allocating any resources associated with it.

Parameters

theCab	The cabinet.
theKey	The key name.
numBytes	<i>(Filled by the method, may be NULL)</i> If non- NULL , contains the size of the object retrieved, in bytes.

Return Value

A pointer to the binary object. Will be **NULL** if **theKey** is not present in **theCab** or if the value stored under **theKey** is not of type **kASTypeBinary**.

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabGetBinary](#)

[ASCabGetBinaryCopy](#)

[ASCabPutBinary](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabDetachCab

[ASCab](#) ASCabDetachCab ([ASCab](#) theCab, const char* theKey);

Description

Retrieves the **ASCab** stored under **theKey** in **theCab** and removes the key from **theCab**. The client assumes ownership of the **ASCab** returned and is responsible for destroying it.

Parameters

theCab	The cabinet.
theKey	The key name.

Return Value

The cabinet. Will be **NULL** if **theKey** is not present in **theCab**, or if the value stored under **theKey** is not of type **kASValueCabinet**.

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabPutCab](#)
[ASCabGetCab](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabDetachPathName

```
void ASCabDetachPathName (ASCab theCab, const char* theKey,
    ASFileSys* fileSys, ASPathName* pathName);
```

Description

Retrieves the **ASPathName** stored under **theKey** in **theCab** and removes the key from **theCab**.

Both **fileSys** and **pathName** will be **NULL** if **theKey** was not found, or there was no valid **ASPathName** stored under the key, or if the **ASPathName** does not point to an existing file.

It is the client's responsibility to release the memory associated with the **ASPathName** using **ASFileSysReleasePath**.

Parameters

theCab	The cabinet.
theKey	The key name.
fileSys	(Filled by the method) The ASFileSys that pathName was opened through.
pathName	(Filled by the method) The pathname.

Return Value

None

Known Exceptions

genErrNoMemory

Any exceptions raised by **ASFileSysPathFromDIPath**.

Header File

ASExtraCalls.h

Related Methods

ASCabPutPathName

ASCabGetPathNameCopy

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabDetachPointer

```
void* ASCabDetachPointer (ASCab theCab, const char* theKey,
const char* expectedType, ASBool* noRefs);
```

Description

Retrieves the pointer stored under **theKey** in **theCab** and removes the key from **theCab**. If **noRefs** is set to **true**, the client assumes ownership of the data referenced by the pointer and is responsible for de-allocating any resources associated with it.

Parameters

theCab	The cabinet.
theKey	The key name.
expectedType	The data type referenced by the pointer. Since ASCabDetachPointer is actually a macro, pass the type as literal name, not a string—for example, PDDoc , not "PDDoc" . Pointers are always “typed” in that they always have associated with them a string indicating the type to which they point.
noRefs	<i>(Filled by the method, may be NULL)</i> If non- NULL , a value of true indicates that there are no other ASCabs that reference this pointer, and a value of false indicates that some ASCab still contains a copy of the pointer.

Return Value

The pointer value stored under **theKey**. Will be **NULL** if **theKey** is not present in **theCab** or the value stored under **theKey** is not of type **kASValuePointer** or the type of the pointer does not match **expectedType**.

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabGetPointer](#)
[ASCabPutPointer](#)

Product

reader, base, pro, pdf



Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



ASCabDetachString

```
char* ASCabDetachString (ASCab theCab, const char* theKey);
```

Description

Retrieves the string stored under **theKey** in **theCab** and removes the key from **theCab**. The client assumes ownership of the string and is responsible for de-allocating any resources associated with it.

Parameters

theCab	The cabinet.
theKey	The key name.

Return Value

The string stored under **theKey**. Will be **NULL** if **theKey** is not present in **theCab**, or if the value stored under **theKey** is not of type **kASValueString**.

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabGetString](#)

[ASCabGetStringCopy](#)

[ASCabPutString](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASCabDetachText

```
ASText ASCabDetachText (ASCab theCab, const char* theKey);
```

Description

Retrieves the **ASText** object stored under **theKey** in **theCab** and removes the key from **theCab**.

The client assumes ownership of the **ASText** object and is responsible for de-allocating it using **ASTextDestroy**.

Parameters

theCab	The cabinet.
theKey	The key name.

Return Value

The **ASText** object stored under **theKey**. Will be **NULL** if **theKey** is not present in **theCab**, or if the value stored under **theKey** is not of type **kASValueText**.

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabGetText](#)

[ASCabPutText](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASCabDup

```
ASCab ASCabDup (ASCab srcCab);
```

Description

Creates a new **ASCab** and populates it with copies of the key-value pairs in **srcCab**. Equivalent to **ASCabCopy** (**srcCab**, **ASCabNew**()).

Parameters

srcCab	The source cabinet.

Return Value

The newly created **ASCab**.

Known Exceptions

[genErrBadParm](#)
[genErrNoMemory](#)

Header File

`ASExtraCalls.h`

Related Methods

[ASCabCopy](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

ASCabEnum

```
void ASCabEnum (ASCab theCab, ASCabEnumProc enumProc,
void* clientData);
```

Description

Enumerates all the keys in the cabinet.

Keys consisting solely of digits are enumerated first, in numeric order (assuming they're not padded with zeros at the front, which will confuse matters). Non-numeric keys are then enumerated in **strcmp** order.

It is safe to add, delete, and modify items in **theCab** during the enumeration. Items that are added during the enumeration will not be enumerated. Modified items that have been enumerated already will not be enumerated again. Delete items that have not yet been enumerated will not be enumerated.

Parameters

theCab	The cabinet.
enumProc	User-supplied callback that is called for each entry found in theCab .
clientData	Pointer to user-supplied data to pass to enumProc each time it is called.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Will **RERAISE** any exceptions thrown by **enumProc**.

Header File

ASExtraCalls.h

Related Methods

[ASConstCabEnum](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASCabEqual

```
ASBool ASCabEqual (ASCab cab1, ASCab cab2);
```

Description

Compares two cabinets and verifies that they have a matching set of keys and all key values are equal (as reported by [ASCabValueEqual](#)).

Parameters

cab1	The first cabinet.
cab2	The second cabinet.

Return Value

true if the cabinets are equal, **false** otherwise.

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabValueEqual](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

ASCabFromEntryList

```
ASCab ASCabFromEntryList (const ASCabEntryRec* entryList);
```

Description

Builds a cabinet based on a constant array of **ASCabDescriptor** records (see **ASCabEntryRec**) . The first entry in each descriptor specifies the name of the key; subsequent fields contain the value. The entry list must end with a descriptor containing **NULL** for the key name. See `ASEExtraExpT.h` for more info.

Parameters

entryList	A constant array of ASCabDescriptor records (see ASCabEntryRec) . Passing NULL generates an empty ASCab .

Return Value

The newly created **ASCab**.

Known Exceptions

[genErrBadParm](#)

Header File

`ASEExtraCalls.h`

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



ASCabGetAtom

```
ASAtom ASCabGetAtom (ASCab theCab, const char* theKey,
ASAtom defValue);
```

Description

Returns the **ASAtom** value stored under **theKey** in **theCab**.

Parameters

theCab	The cabinet.
theKey	The key name.
defValue	The default value.

Return Value

The **ASAtom** value stored under **theKey** if the key is found and the value stored under it is of type **kASValueAtom**; otherwise **defValue** is returned.

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabPutAtom](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabGetBinary

```
const void* ASCabGetBinary (ASConstCab theCab,
const char* theKey, ASArraySize* numBytes);
```

Description

Returns the binary object stored under **theKey** in **theCab**.

Parameters

theCab	The cabinet.
theKey	The key name.
numBytes	<i>(Filled by the method, may be NULL)</i> If non- NULL , contains the size of the object returned, in bytes.

Return Value

The binary object stored under **theKey** if the key is found and the value stored under it is of type **kASValueBinary**; otherwise **NULL** is returned. This object is owned by the **ASCab** and should not be destroyed by the caller.

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

- [ASCabGetBinaryCopy](#)
- [ASCabDetachBinary](#)
- [ASCabPutBinary](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASCabGetBinaryCopy

```
void* ASCabGetBinaryCopy (ASConstCab theCab,
    const char* theKey, ASArraySize* numBytes);
```

Description

Returns a copy of the binary object stored under **theKey** in **theCab**. It is the client's responsibility to release the memory associated with the object using [ASfree](#).

Parameters

theCab	The cabinet.
theKey	The key name.
numBytes	<i>(Filled by the method, may be NULL)</i> If non- NULL , contains the size of the object returned.

Return Value

The binary object stored under **theKey** if the key is found and the value stored under it is of type **kASValueBinary**; otherwise **NULL** is returned.

Known Exceptions

[genErrBadParm](#)
[genErrNoMemory](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabGetBinary](#)
[ASCabDetachBinary](#)
[ASCabPutBinary](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASCabGetBool

```
ASBool ASCabGetBool (ASCab theCab, const char* theKey,  
ASBool defValue);
```

Description

Returns the [ASBool](#) value stored under **theKey** in **theCab**.

Parameters

theCab	The cabinet.
theKey	The key name.
defValue	The default value.

Return Value

The [ASBool](#) value stored under **theKey** if the key is found and the value stored under it is of type [kASValueBool](#); otherwise **defValue** is returned.

Known Exceptions

[genErrBadParm](#)

Header File

`ASExtraCalls.h`

Related Methods

[ASCabPutBool](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

ASCabGetCab

```
ASCab ASCabGetCab (ASCab theCab, const char* theKey);
```

Description

Returns the **ASCab** stored under **theKey** in **theCab**.

Parameters

theCab	The cabinet.
theKey	The key name.

Return Value

The **ASCab** stored under **theKey** if the key is found and the value stored under it is of type **kASValueCabinet**; otherwise **NULL** is returned. This object is owned by **theCab** and should not be destroyed by the client.

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabPutCab](#)

[ASCabDetachCab](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabGetDouble

```
double ASCabGetDouble (ASCab theCab, const char* theKey,
double defValue);
```

Description

Returns the double value stored under **theKey** in **theCab**. If the value stored under **theKey** is of type **kASValueInteger**, this value will be cast to a double and returned to the client.

Parameters

theCab	The cabinet.
theKey	The key name.
defValue	The default value.

Return Value

The double value stored under **theKey** if the key is found and the value stored under it is of type **kASValueDouble** or **kASValueInteger**; otherwise **defValue** is returned.

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabPutDouble](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabGetInt

```
ASInt32 ASCabGetInt (ASCab theCab, const char* theKey,
ASInt32 defValue);
```

Description

Returns the [ASInt32](#) value stored under **theKey** in **theCab**.

Parameters

theCab	The cabinet.
theKey	The key name.
defValue	The default value.

Return Value

The [ASInt32](#) value stored under **theKey** if the key is found and the value stored under it is of type [kASValueInteger](#); otherwise **defValue** is returned.

Known Exceptions

[genErrBadParm](#)

Header File

`ASExtraCalls.h`

Related Methods

[ASCabPutInt](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ASEXTRA_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

ASCabGetPathNameCopy

```
void ASCabGetPathNameCopy (ASCab theCab, const char* theKey,
ASFileSys* fileSys, ASPathName* pathName);
```

Description

Returns a copy of [ASPathName](#) stored under **theKey** in **theCab**. It is the client's responsibility to release the [ASPathName](#) using [ASFileSysReleasePath](#).

Both **fileSys** and **pathName** will be **NULL** if **theKey** was not found, or there was no valid [ASPathName](#) stored under the key, or if the pathname does not point to an existing file.

Parameters

theCab	The cabinet.
theKey	The key name.
fileSys	<i>(Filled by the method)</i> The ASFileSys that pathName was opened through.
pathName	<i>(Filled by the method)</i> The pathname.

Return Value

None

Known Exceptions

[genErrNoMemory](#)

Any exception raised by [ASFileSysPathFromDIPath](#).

Header File

[ASExtraCalls.h](#)

Related Methods

[ASCabPutPathName](#)

[ASCabDetachPathName](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ASEXTRA_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

ASCabGetPointer

```
void* ASCabGetPointer (ASCab theCab, const char* theKey,
const char* expectedType);
```

Description

Returns the pointer value stored under **theKey** in **theCab**.

Parameters

theCab	The cabinet.
theKey	The key name.
expectedType	The data type referenced by the pointer. Since ASCabGetPointer is actually a macro, you should pass the type as literal name, not a string—for example, PDDoc , not "PDDoc" . Pointers are always “typed” in that they always have associated with them a string indicating the type to which they point.

Return Value

The pointer value stored under **theKey** if the key is found and the value stored under **theKey** is of type **kASValuePointer** and the type of the pointer matches **expectedType**; otherwise **NULL** is returned. The object referenced by this pointer is owned by **theCab** and should not be destroyed by the client.

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabDetachPointer](#)
[ASCabPutPointer](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASCabGetPointerDestroyProc

[ASCabPointerDestroyProc](#) ASCabGetPointerDestroyProc
([ASCab](#) theCab, const char* theKey);

Description

Obtains the resource de-allocation callback associated with the pointer stored under **theKey** in **theCab**. When the reference count of the pointer falls to zero, the callback is called to free the resources associated with the object it references.

Parameters

theCab	The cabinet.
theKey	The key name.

Return Value

The callback (if any) associated with the pointer if the key is found and the value stored under it is of type **kASValuePointer**; otherwise **NULL** is returned.

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabDetachPointer](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabGetPointerType

```
const char* ASCabGetPointerType (ASCab theCab,  
const char* theKey);
```

Description

Returns a string representation of the data type referenced by the pointer stored under **theKey** in **theCab**.

Parameters

theCab	The cabinet.
theKey	The key name.

Return Value

The string if the key is found and the value stored under it is of type **kASValuePointer**; otherwise **NULL** is returned.

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabGetString

```
const char* ASCabGetString (ASCab theCab, const char* theKey);
```

Description

Returns the string stored under **theKey** in **theCab**.

Parameters

theCab	The cabinet.
theKey	The key name.

Return Value

The string stored under **theKey** if the key is found and the value stored under it is of type **kASValueString**; otherwise **NULL** is returned. The object referenced by this pointer is owned by **theCab** and should not be destroyed by the client.

Known Exceptions

[genErrBadParm](#)
[genErrNoMemory](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabGetStringCopy](#)
[ASCabDetachString](#)
[ASCabPutString](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabGetStringCopy

```
char* ASCabGetStringCopy (ASCab theCab, const char* theKey);
```

Description

Returns a copy of the string stored under **theKey** in **theCab**.

It is the client's responsibility to release the memory allocated for the string using [ASfree](#).

Parameters

theCab	The cabinet.
theKey	The key name.

Return Value

A copy of the string stored under **theKey** if the key is found and the value stored under it is of type **kASValueString**; otherwise **NULL** is returned.

Known Exceptions

[genErrBadParm](#)
[genErrNoMemory](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabGetString](#)
[ASCabDetachString](#)
[ASCabPutString](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabGetText

```
ASText ASCabGetText (ASCab theCab, const char* theKey);
```

Description

Returns the **ASText** object stored under **theKey** in **theCab**.

Parameters

theCab	The cabinet.
theKey	The key name.

Return Value

The **ASText** object stored under **theKey** if the key is found and the value stored under it is of type **kASValueText**; otherwise **NULL** is returned. This object is owned by **theCab** and should not be destroyed by the client.

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabDetachText](#)

[ASCabPutText](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabGetType

```
ASCabValueType ASCabGetType (ASCab theCab,  
const char* theKey);
```

Description

Returns the type of the value stored under **theKey** in **theCab**.

Parameters

theCab	The cabinet.
theKey	The key name.

Return Value

The type of the value stored under **theKey**, or **kASValueUnknown** if the key is not found.

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabGetUns

```
ASUns32 ASCabGetUns (ASConstCab theCab, const char* theKey,  
ASUns32 defValue);
```

Description

Returns the [ASUns32](#) value stored under **theKey** in **theCab**.

Parameters

theCab	The cabinet.
theKey	The key name.
defValue	The default value.

Return Value

The [ASUns32](#) value stored under **theKey** if the key is found and the value stored under it is of type [kASValueUns](#); otherwise **defValue** is returned.

Known Exceptions

[genErrBadParm](#)

Header File

`ASExtraCalls.h`

Related Methods

[ASCabPutUns](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ASEXTRA_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.



ASCabKnown

```
ASBool ASCabKnown (ASCab theCab, const char* theKey);
```

Description

Returns **true** if **theKey** is present in **theCab**.

Parameters

theCab	The cabinet.
theKey	The key name.

Return Value

See above.

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASCabMakeEmpty

```
void ASCabMakeEmpty (ASCab theCab);
```

Description

Removes all keys from **theCab** and destroys all values they point to.

Parameters

theCab	The cabinet.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



ASCabMunge

```
void ASCabMunge (ASCab theCab, ASCab keyCab,  
ASCabMungeAction action);
```

Description

Munges the keys and the corresponding values in **theCab** based on the keys in **keyCab** and the munge action. Note that **keyCab** is never altered; **theCab** is.

Parameters

theCab	The cabinet to be modified.
keyCab	The cabinet used to modify theCab .
action	The type of action to be taken.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASCabNew

```
ASCab ASCabNew (void);
```

Description

Creates a new, empty cabinet.

Parameters

None

Return Value

The newly created cabinet.

Known Exceptions

[genErrNoMemory](#)

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



ASCabNumEntries

```
ASTArraySize ASCabNumEntries (ASCab theCab);
```

Description

Returns the number of key-value pairs in **theCab**.

Parameters

theCab	The cabinet.

Return Value

See above.

Known Exceptions

genErrBadParm

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASCabPutAtom

```
void ASCabPutAtom (ASCab theCab, const char* theKey,  
ASAtom atomValue);
```

Description

Stores an **ASAtom** value in **theCab** under **theKey**.

Parameters

theCab	The cabinet.
theKey	(May be NULL) The key name.
atomValue	The value to be stored.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabGetAtom](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabPutBinary

```
void ASCabPutBinary (ASCab theCab, const char* theKey,
void* theBlob, ASArraySize blobSize);
```

Description

Stores a binary object in **theCab** under **theKey**. The **ASCab** assumes ownership of the binary object, so the client should not attempt to free the memory associated with it. The binary object must have been allocated using **ASmalloc**.

Parameters

theCab	The cabinet.
theKey	(May be NULL) The key name.
theBlob	(May be NULL) A pointer to the binary object to be stored. If NULL , the value (if any) stored under theKey in theCab is destroyed and theKey removed from theCab .
blobSize	The size of the binary object.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

- [ASCabGetBinary](#)
- [ASCabGetBinaryCopy](#)
- [ASCabDetachBinary](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabPutBool

```
void ASCabPutBool (ASCab theCab, const char* theKey,
ASBool boolValue);
```

Description

Stores an [ASBool](#) value in **theCab** under **theKey**.

Parameters

theCab	The cabinet.
theKey	(May be NULL) The key name.
boolValue	The value to be stored.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabGetBool](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

ASCabPutCab

```
void ASCabPutCab (ASCab theCab, const char* theKey,
ASCab cabVal);
```

Description

Stores an **ASCab** in **theCab** under **theKey**. If the cabinet is already a value for some other **ASCab**, **ASCabPutCab** will raise, that is, any cabinet can be contained by at most one other cabinet.

theCab assumes ownership of the cabinet, so the client must not destroy it.

Parameters

theCab	The cabinet.
theKey	(May be NULL) The key name.
cabVal	(May be NULL) The ASCab to be stored in theCab . If cabVal is NULL then any value under theKey is destroyed and theKey is removed from theCab .

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabGetCab](#)

[ASCabDetachCab](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabPutDouble

```
void ASCabPutDouble (ASCab theCab, const char* theKey,  
double doubleValue);
```

Description

Stores a double value in **theCab** under **theKey**.

Parameters

theCab	The cabinet.
theKey	(May be <i>NULL</i>) The key name.
doubleValue	The value to be stored.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabGetDouble](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

ASCabPutInt

```
void ASCabPutInt (ASCab theCab, const char* theKey,  
ASInt32 intValue);
```

Description

Stores an [ASInt32](#) value in **theCab** under **theKey**.

Parameters

theCab	The cabinet.
theKey	(May be NULL) The key name.
intValue	The value to be stored.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabGetInt](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

ASCabPutNull

```
void ASCabPutNull (ASCab theCab, const char* theKey);
```

Description

Stores a value with a type of `kASValueNull` in `theCab` under `theKey`. Null cabinet entries are used as placeholders or to removed other cabinet entries during an [ASCabMunge](#) operation.

Parameters

<code>theCab</code>	The cabinet.
<code>theKey</code>	The key name.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

`ASExtraCalls.h`

Product

reader, base, pro, pdf

Availability

Available if `PI_ASEXTRA_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

ASCabPutPathName

```
void ASCabPutPathName (ASCab theCab, const char* theKey,
    ASFileSys fileSys, ASPathName path);
```

Description

Stores an **ASPathName** in **theCab** under **theKey**.
theCab assumes ownership of the **ASPathName**, so the client need not call **ASFileSysReleasePath**.

Parameters

theCab	The cabinet.
theKey	(May be NULL) The key name.
fileSys	The ASFileSys from which path was obtained.
path	(May be NULL) The ASPathName to be stored. If NULL , the value (if any) stored under theKey in theCab is destroyed and theKey is removed from theCab .

Return Value

None

Known Exceptions

genErrBadParm

Header File

ASExtraCalls.h

Related Methods

ASCabGetPathNameCopy
ASCabDetachPathName

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASCabPutPointer

```
void ASCabPutPointer (ASCab theCab, const char* theKey,
const char* theType, void* ptrValue,
ASCabPointerDestroyProc destroyProc);
```

Description

Stores a pointer in **theCab** under **theKey**. See **ASCab** description for more information.

Parameters

theCab	The cabinet.
theKey	<i>(May be NULL)</i> The key name.
theType	The data type referenced by the pointer. Since ASCabPutPointer is actually a macro, you should pass the type as literal name, not a string—for example, PDDoc , not " PDDoc ". Pointers are always “typed” in that they always have associated with them a string indicating the type to which they point.
ptrValue	The value to be stored.
destroyProc	<i>(May be NULL)</i> A user-supplied callback which is called when the reference count associated with thePtr is zero.

Return Value

None

Known Exceptions

[genErrBadParm](#)
[genErrNoMemory](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabGetPointer](#)
[ASCabDetachPointer](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabPutString

```
void ASCabPutString (ASCab theCab, const char* theKey,
char* theStr);
```

Description

Stores a string in **theCab** under **theKey**. A string consists of some number of bytes followed by a single null (zero) byte. The string must have been allocated using [ASmallloc](#).

theCab assumes ownership of the string, so the client should not attempt to free the memory associated with it.

Parameters

theCab	The cabinet.
theKey	(May be NULL) The key name.
strValue	(May be NULL) The string to be stored. If NULL , the value (if any) stored under theKey in theCab is destroyed and theKey is removed from theCab .

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabGetString](#)

[ASCabGetStringCopy](#)

[ASCabDetachString](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabPutText

```
void ASCabPutText (ASCab theCab, const char* theKey,
  AText theText);
```

Description

Stores an **AText** object in **theCab** under **theKey**.

theCab assumes ownership of the object, so the client should not attempt to free the memory associated with it.

Parameters

theCab	The cabinet.
theKey	(May be NULL) The key name.
theText	(May be NULL) The object to be stored. If NULL , the value (if any) stored under theKey in theCab is destroyed and theKey is removed from theCab .

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabGetText](#)
[ASCabDetachText](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASCabPutUns

```
void ASCabPutUns (ASConstCab theCab, const char* theKey,  
ASUns32 theUns);
```

Description

Stores the [ASUns32](#) value under **theKey** in **theCab**.

Parameters

theCab	The cabinet.
theKey	The key name.
theUns	The value to be stored.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabGetUns](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASCabReadFromStream

[ASCab](#) ASCabReadFromStream ([ASStm](#) stm);

Description

Reads a previously written cabinet from a stream

Parameters

ASStm	Must be a stream opened through ASFileStmRdOpen , ASMemStmRdOpen , or ASProcStmRdOpenEx .

Return Value

The [ASCab](#), or **NULL** if it could not be constructed.

Header File

ASExtraCalls.h

Related Methods

[ASCabWriteToStream](#)

Example

```
// This code writes a cabinet out to a stream and reads it back in.
ASCab asCab;
ASFile asFile;
ASStm asStm;
... // Code to initialize asFile and create/populate asCab.
// Write the cabinet out to the stream.
asStm = ASFileStmWrOpen (asFile, 0);
ASCabWriteToStream (asCab, asStm);

// Destroy the cabinet and close the stream.
ASCabDestroy (asCab);
ASStmClose (asStm);

// Set the file position back to the start of the file.
ASFileSetPos (asFile, 0);

// Read the cabinet in from the stream.
stm = ASFileStmRdOpen (asFile, 0);
asCab = ASCabReadFromStream (asStm);
... // Eventually close the stream and destroy the cabinet.
```

Product

reader, base, pro, pdf



Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



ASCabRemove

```
void ASCabRemove (ASCab theCab, const char* theKey);
```

Description

Removes **theKey** entry from **theCab**, destroying the associated value.

Parameters

theCab	The cabinet.
theKey	The key name.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

ASCabRename

```
void ASCabRename (ASCab theCab, const char* oldKeyName, const char* newKeyName);
```

Description

Renames a key within **theCab** while preserving the value associated with it. If there is already a key equal to **newKeyName** in **theCab**, its value will be destroyed and replaced with the value of **oldKeyName**.

Any attempt to move the item from one sub-cabinet to another will cause **ASCabRename** to raise an exception (for example, **ASCabRename (theCab, "SubCab1:Key1", "SubCab2:Key1")** will raise). If this routine raises an exception **theCab** will be untouched.

Parameters

theCab	The cabinet.
oldKeyName	The key name to be changed.
newKeyName	The new name.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASCabValueEqual

```
ASBool ASCabValueEqual (ASCab cab1, const char* theKey1,
                        ASCab cab2, const char* theKey2);
```

Description

Compares two cabinet values and returns **true** if and only if they are equal—that is, have the same type and value. Cabinets are compared using [ASCabEqual](#). [ASText](#) values are compared by using [ASTextCmp](#) and testing for a return value of 0 (zero). Strings and binary values must have the same lengths and byte-for-byte contents. Booleans, atoms, doubles, and integers must have equal values. Pointer values must point to the same location in memory (but may have different “destroyProcs” and type strings).

Parameters

cab1	The first cabinet.
theKey1	The key name.
cab2	The second cabinet.
theKey2	The key name.

Return Value

See above.

Known Exceptions

[genErrBadParm](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabEqual](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

ASCabWriteToStream

```
void ASCabWriteToStream (ASCab theCab, ASStm theStm);
```

Description

Writes **theCab** out to a stream. The caller retains ownership of the cabinet. The stream will not be closed or flushed.

Parameters

theCab	The cabinet.
theStm	Must be a stream opened through ASFileStmWrOpen or ASProcStmWrOpen .

Return Value

None

Known Exceptions

[genErrBadParm](#)
[fileErrWrite](#)

Header File

ASExtraCalls.h

Related Methods

[ASCabReadFromStream](#)

Example

```
// The following code snippet illustrates writing a cabinet
// out to a stream and reading it back in.

ASCab asCab;
ASFile asFile;
ASStm asStm;

... // Code to initialize asFile and create/populate asCab.

// Write the cabinet out to the stream.
asStm = ASFileStmWrOpen (asFile, 0);
ASCabWriteToStream (asCab, asStm);

// Destroy the cabinet and close the stream.
ASCabDestroy (asCab);
ASStmClose (asStm);

// Set the file position back to the start of the file.
ASFileSetPos (asFile, 0);

// Read the cabinet in from the stream.
stm = ASFileStmRdOpen (asFile, 0);
asCab = ASCabReadFromStream (asStm);

... // Eventually close the stream and destroy the cabinet.
```

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

ASConstCabEnum

```
void ASConstCabEnum (ASConstCab theCab,
ASConstCabEnumProc enumProc, void* clientData);
```

Description

Enumerates all the keys in the constant cabinet.

Keys consisting solely of digits are enumerated first, in numeric order (assuming they're not padded with zeros at the front, which will confuse matters.). Non-numeric keys are then enumerated in **strcmp** order.

The callback procedure must not add, delete, or modify items in **theCab** during the enumeration.

Parameters

theCab	The cabinet.
enumProc	User-supplied callback that is called for each entry found in theCab . This callback cannot modify the ASConstCab object.
clientData	Pointer to user-supplied data to pass to enumProc each time it is called.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Will **RERAISE** any exceptions thrown by **enumProc**.

Header File

ASExtraCalls.h

Related Methods

[ASCabEnum](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASCallback

ASCallbackCreate

```
ASCallback ACallbackCreate (ASExtension extensionID,
void* proc);
```

Description

Creates a callback that allows the Acrobat viewer to call a function in a plug-in. All plug-in functions that are called by the Acrobat viewer must be converted to callbacks before being passed to the viewer.

Whenever possible, plug-ins should not call **ASCallbackCreate** directly, but should use the macros [ASCallbackCreateProto](#), [ASCallbackCreateNotification](#), and [ASCallbackCreateReplacement](#). These macros (which eventually call **ASCallbackCreate**) have two advantages:

- They allow compilers to perform type checking, eliminating one extremely common source of plug-in bugs.
- They handle **extensionID** automatically.

NOTE: If you call **ASCallbackCreate** directly, you are actually invoking the **ASCallbackCreate** macro not this HFT routine. The **ASCallbackCreate** macro takes only one parameter, the **proc**, and passes that information into this underlying HFT routine as the second argument. The first argument is always set to [gExtensionID](#), which should be the extension ID of your plug-in.

Plug-ins must use **ASCallbackCreate** directly, for example, when calling a Macintosh toolbox routine that expects a **ProcPtr**.

Parameters

extensionID	The gExtensionID extension that calls proc .
proc	The user-supplied procedure for which a callback is created.

Return Value

The newly created callback.

Header File

CorCalls.h

Related Methods

[ASCallbackDestroy](#)

**Related Macros**[ASCallbackCreateNotification](#)[ASCallbackCreateReplacement](#)[ASCallbackCreateProto](#)**Product**

reader, base, pro, pdf

Availability

Available if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

ASCallbackCreateNotification

ASCallback ASCallbackCreateNotification(nsel, proc);

Description

Macro that creates a callback for the specified notification.

Performs compile-time type-checking of the procedure if the **DEBUG** macro is nonzero.

Parameters

nsel	The name of the notification for which the callback is created.
proc	A pointer to the user-supplied procedure for which a callback is created. The declaration of proc must be consistent with the notification with which it will be used (as specified by nsel).

Return Value

Needs to be saved so that it can be passed to **AVAppRegisterNotification** and **AVAppUnregisterNotification**, and destroyed after use with the method **ASCallbackDestroy**.

Header File

AVCalls.h

Related Macros

[AVAppRegisterNotification](#)
[AVAppUnregisterNotification](#)
[ASCallbackDestroy](#)
[ASCallbackCreateReplacement](#)
[ASCallbackCreateProto](#)
[ACCB1](#)
[ACCB2](#)
[DEBUG](#)

Related Methods

[ASCallbackCreate](#)

Example

```
ASCallback myNotification;
myNotification = ASCallbackCreateNotification(
    AVDocDidOpen, myDocDidOpen);
```

ASCallbackCreateReplacement

ASCallback AScallbackCreateReplacement (nse1, proc);

Description

Macro that creates a callback appropriate for use in replacing one of the Acrobat viewer's built-in methods. The method being replaced must be one of the [Replaceable Methods](#).

Performs compile-time type-checking of the procedure if the **DEBUG** macro is nonzero.

Parameters

nse1	The name of the method for which the replacement callback is created.
proc	A pointer to the user-supplied procedure for which a callback is created. The declaration of proc must be consistent with the method being replaced.

Header File

ASCalls.h

Related Macros

[ASCallbackCreateNotification](#)
[ASCallbackCreateProto](#)
[REPLACE](#)
[CALL_REPLACED_PROC](#)
[ACCB1](#)
[ACCB2](#)
[DEBUG](#)

Related Methods

[ASCallbackCreate](#)

Examples

```
ASCallback myAlertCallback;

myAlertCallback = AScallbackCreateReplacement(AVAlertSEL, myAlert);
REPLACE(gAcroViewHFT, AVAlertSEL, myAlertCallback);
```



ASCallbackDestroy

```
void AScallbackDestroy (ASCallback callback);
```

Description

Destroys a callback.

Parameters

callback	The callback to destroy.

Return Value

None

Header File

CorCalls.h

Related Methods

[ASCallbackCreate](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



ASDate

ASDateAddCalendarTimeSpan

```
void ASDateAddCalendarTimeSpan (ASDate date,  
const ASCalendarTimeSpan timeSpan);
```

Description

Adds a calendar time span to a date. Modifies the date by the length of time provided in the [ASCalendarTimeSpan](#) object. Note that there is some ambiguity in a calendar time span; to add an exact time span (for example, 2592000 seconds rather than one month) use [ASDateAddTimeSpan](#).

Parameters

date	The date.
timeSpan	The calendar time span to add.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASDateAddTimeSpan](#)

[ASDateSubtractCalendarTimeSpan](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASDateAddTimeSpan

```
void ASDateAddTimeSpan (ASDate date,  
const ASTimeSpan timeSpan);
```

Description

Adds a time span (an exact number of seconds) to a date. Modifies the date by the length of time provided in the [ASTimeSpan](#) object.

Parameters

date	The date.
timeSpan	The time span to add.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASDateAddCalendarTimeSpan](#)

[ASDateSubtractTimeSpan](#)

[ASTimeSpanAdd](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



ASDateCalendarDiff

```
void ASDateCalendarDiff (const ASDate date1,
const ASDate date2, ASCalendarTimeSpan result);
```

Description

Calculates the difference between 2 [ASDate](#) objects and stores the result in the provided [ASCalendarTimeSpan](#) object. The result is broken down into years, months, and so on, in the highest denomination possible. For example, a difference of 13 months is reported as 1 year and 1 month.

Parameters

date1	The first date.
date2	The second date.
result	The calendar time span structure in which to store the difference.

Return Value

None

Header File

`ASExtraCalls.h`

Related Methods

[ASDateExactDiff](#)
[ASDateCompare](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.



ASDateClear

```
void ASDateClear (ASDate asDate) ;
```

Description

Reinitializes a date object to the newly-allocated state, as returned by [ASDateNew](#).

Parameters

asDate	The date object.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASDateDestroy](#)

[ASDateNew](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASDateCompare

```
ASInt32 ASDateCompare (const ASDate date1,  
const ASDate date2);
```

Description

Tests whether one date is earlier or later than another.

Parameters

date1	The first date.
date2	The second date.

Return Value

1 if **date1**>**date2**, 0 if they are equal, -1 if **date1**<**date2**.

Header File

ASExtraCalls.h

Related Methods

[ASDateExactDiff](#)
[ASTimeSpanCompare](#)
[ASCalendarTimeSpanCompare](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



ASDateCopy

```
void ASDateCopy (const ASDate original, ASDate copy);
```

Description

Copies date and time data from one date object to another.

Parameters

original	The date to be copied.
copy	The date into which the data is copied.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASDateClear](#)

[ASDateDup](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



ASDateDestroy

```
void ASDateDestroy (ASDate date);
```

Description

Releases and destroys a date object.

Parameters

date	The date.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASDateClear](#)

[ASDateDup](#)

[ASDateNew](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



ASDateDup

```
ASDate ASDateDup (const ASDate date);
```

Description

Creates a new date object containing the same data as an existing date object.

Parameters

date	The date to duplicate.

Return Value

The new date object.

Header File

ASExtraCalls.h

Related Methods

[ASDateCopy](#)
[ASDateDestroy](#)
[ASDateNew](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASDateExactDiff

```
void ASDateExactDiff (const ASDate date1, const ASDate date2,
ASTimeSpan result);
```

Description

Calculates the exact difference in seconds between two date objects and stores the result in the provided [ASTimeSpan](#) object. If **date1** is earlier than **date2**, the result is negative.

Parameters

date1	The first date.
date2	The second date.
result	The time span structure in which to store the difference.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

- [ASDateCalendarDiff](#)
- [ASDateCompare](#)
- [ASTimeSpanDiff](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASDateGetLocalTime

```
ATimeRec ASDateGetLocalTime (ASDate date);
```

Description

Creates a time record that represents the local time represented by the date object. The resulting local time might not account for daylight savings time correctly if the date object has been modified by adding or subtracting a time span or calendar time span.

Raises an exception if there is not enough memory.

Parameters

date	The date.

Return Value

The newly created time record.

Header File

ASExtraCalls.h

Related Methods

[ASDateGetTimeString](#)
[ASDateGetUTCTime](#)
[ASDateSetTimeFromRec](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASDateGetTimeString

```
char* ASDateGetTimeString (ASDate date,
ASDateTimeFormat format);
```

Description

Creates a time string from a date object according to a specified format. If time zone information is available in the date object, the string contains local time along with the time zone adjustment, if that is supported by the requested format.

Raises an exception if there is not enough memory.

Parameters

date	The date object.
format	The format of the time string.

Return Value

The time string in the specified format.

Header File

ASExtraCalls.h

Related Methods

[ASDateClear](#)
[ASDateGetLocalTime](#)
[ASDateGetUTCTime](#)
[ASDateSetTimeFromString](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



ASDateGetUTCTime

```
ATimeRec ASDateGetUTCTime (ASDate date);
```

Description

Creates a time record that represents the UTC time represented by the date object. Raises an exception if there is not enough memory.

Parameters

date	The date object.

Return Value

The newly created time record.

Header File

ASExtraCalls.h

Related Methods

[ASDateClear](#)
[ASDateGetLocalTime](#)
[ASDateGetUTCTime](#)
[ASDateSetTimeFromString](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



ASDateNew

```
ASDate ASDateNew (void);
```

Description

Creates a date object. The newly allocated object reflects the epoch time: Jan 1 1970 00:00:00 UTC.

Raises an exception if there is not enough memory for the operation.

Parameters

None

Return Value

The newly created date object.

Header File

ASExtraCalls.h

Related Methods

[ASDateClear](#)
[ASDateCopy](#)
[ASDateDestroy](#)
[ASDateDup](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

ASDateSetLocalTimeOffset

```
void ASDateSetLocalTimeOffset (ASDate date);
```

Description

Sets a date object's local time offset according to the operating system's current time zone information. Different operating systems handle daylight savings differently. This method causes the date object to always use the same DST offset that the operating system is currently using, even if the date object is modified.

Parameters

date	The date.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASDateSetToCurrentLocalTime](#)

[ASDateSetToCurrentUTCTime](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASDateSetTimeFromRec

```
void ASDateSetTimeFromRec (ASDate date,  
    const ASTimeRec* timeRec);
```

Description

Initializes a date object from a time record. Raises an exception if the time structure represents an invalid time, such as January 32nd 1999 or Feb 29th 2000. Assumes that the parameters for day and month in the time record are 1-based.

Parameters

date	The date object.
timeRec	The time record.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASDateClear](#)
[ASDateSetToCurrentLocalTime](#)
[ASDateSetToCurrentUTCTime](#)
[ASDateSetTimeFromString](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASDateSetTimeFromString

```
void ASDateSetTimeFromString (ASDate date,
    const char* timeString, ASDateTimeFormat format);
```

Description

Initializes a date object from a time string. Raises an exception if there is not enough memory, if the format is unrecognized, or if the time string is not formatted according to the supplied format.

Parameters

date	The date object.
timeString	The time string, in the specified format.
format	The format of the time string.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASDateClear](#)
[ASDateSetToCurrentLocalTime](#)
[ASDateSetToCurrentUTCTime](#)
[ASDateSetTimeFromString](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

ASDateSetToCurrentLocalTime

```
void ASDateSetToCurrentLocalTime (ASDate date);
```

Description

Sets a date object to the current local time, using the time zone information from the operating system.

Parameters

date	The date.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASDateClear](#)
[ASDateGetLocalTime](#)
[ASDateSetToCurrentUTCTime](#)
[ASDateSetLocalTimeOffset](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASDateSetToCurrentUTCTime

```
void ASDateSetToCurrentUTCTime (ASDate retVal);
```

Description

Sets a date object to the current UTC time with no time zone information.

Parameters

retVal	The date.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASDateClear](#)
[ASDateGetUTCTime](#)
[ASDateSetToCurrentLocalTime](#)
[ASDateSetLocalTimeOffset](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASDateSubtractCalendarTimeSpan

```
void ASDateSubtractCalendarTimeSpan (ASDate date,
const ASCalendarTimeSpan timeSpan);
```

Description

Subtracts a calendar time span from a date. Modifies the date by the length of time provided in the [ASCalendarTimeSpan](#) object. Note that there is some ambiguity in a calendar time span; to subtract an exact time span (for example, 2592000 seconds rather than one month) use [ASDateSubtractTimeSpan](#).

Parameters

date	The date.
timeSpan	The calendar time span to subtract.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASDateAddCalendarTimeSpan](#)
[ASDateSubtractTimeSpan](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



ASDateSubtractTimeSpan

```
void ASDateSubtractTimeSpan (ASDate date,  
    const ATimeSpan timeSpan);
```

Description

Subtracts a time span (an exact number of seconds) from a date. Modifies the date by the length of time provided in the [ATimeSpan](#) object.

Parameters

date	The date.
timeSpan	The time span to subtract.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASDateAddTimeSpan](#)

[ASDateSubtractCalendarTimeSpan](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASExtension

ASEnumExtensions

```
ASExtension ASEnumExtensions (ASExtensionEnumProc proc,  
void* clientData, ASBool onlyLivingExtensions);
```

Description

Enumerates all **ASExtensions**, that is, valid plug-ins.

Parameters

proc	User-supplied callback to call for each plug-in. Enumeration halts if proc returns false .
clientData	Pointer to user-supplied data to pass to proc each time it is called.
onlyLivingExtensions	If true , ASExtensions that have been unloaded or otherwise deactivated are not enumerated. If false , all ASExtensions are enumerated.

Return Value

If **proc** returned **false**, the last **ASExtension** that was enumerated. **NULL** otherwise.

Header File

CorCalls.h

Related Methods

[ASExtensionGetFileName](#)
[ASExtensionGetRegisteredName](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

ASEExtensionGetFileName

```
ASTArraySize ASEExtensionGetFileName (ASEExtension extension,
char* buffer, ASTArraySize bufSize);
```

Description

Gets the filename of an **ASEExtension**.

Parameters

extension	The ASEExtension whose filename is obtained.
buffer	<i>(Filled by the method, may be NULL)</i> Pointer to a buffer for the filename. Pass NULL to have this method return the length of the filename (excluding a terminating NULL character).
bufSize	Number of bytes in buffer . Ignored if buffer is NULL .

Return Value

The number of characters written into **buffer**, excluding the **NULL** character.

Header File

CorCalls.h

Related Methods

[ASEExtensionGetRegisteredName](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

ASExtensionGetRegisteredName

```
ASAtom ASExtensionGetRegisteredName (ASExtension extension);
```

Description

Gets the registered name associated with a plug-in.

Parameters

extension	The ASExtension whose name is obtained.

Return Value

An **ASAtom** representing the plug-in name, or **NULL** if the name could not be identified.

Header File

CorCalls.h

Related Methods

[ASExtensionGetFileName](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



ASFile

ASFileAcquirePathName

```
ASPathName ASFileAcquirePathName (ASFile file);
```

Description

Gets the pathname for **file** and increments an internal reference count. It is the caller's responsibility to release the **ASPathName** using **ASFileSysReleasePath** when it is no longer needed.

Parameters

file	The file whose pathname is acquired.

Return Value

The **ASPathName** associated with **asFile**. You can use **ASFileSysDIPathFromPath** to convert this to a device-independent pathname.

Header File

ASCalls.h

Related Methods

ASFileSysReleasePath

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

ASFileCanSetEOF

```
ASBool ASFileCanSetEOF (ASFile file, ASInt32 newFileSize);
```

Description

Checks if [ASFileSetEOF](#) can be done for this file with a specified new file size.

Parameters

file	The file in question.
newFileSize	The proposed new file size.

Return Value

true if you can set an EOF for the file using the new file size, **false** if the file does not allow this or does not have the necessary callbacks to perform the operation.

Header File

ASCalls.h

Related Methods

[ASFileSetEOF](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.



ASFileClose

```
AS_ERRCODE ASFileClose (ASFile file);
```

Description

Closes the specified file. After a call to **ASFileClose**, the file handle is no longer valid but may be reused as the result of a subsequent call to **ASFileSysOpenFile**.

Parameters

file	The file to close. The file must have been opened previously using ASFileSysOpenFile .

Return Value

Zero if the operation was successful, otherwise some file system/platform dependent error code is returned.

Header File

ASCalls.h

Related Methods

[ASFileFlush](#)
[ASFileReopen](#)
[ASFileStmRdOpen](#)
[ASFileStmWrOpen](#)
[ASFileSysOpenFile](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



ASFileFlush

```
void ASFileFlush (ASFile file);
```

Description

Flushes any buffered data to a file. This method may raise file system/platform specific exceptions.

Parameters

file	The file whose data is flushed.

Return Value

None

Known Exceptions

[fileErrIO](#)

Header File

ASCalls.h

Related Methods

[ASFileHardFlush](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



ASFileFromMDFile

```
ASBool ASFileFromMDFile (ASMDFile fileID, ASFileSys fileSys,
ASFile* pFile);
```

Description

Gets the **ASFile** associated with the specified **ASMDFile** and **ASFileSys**.

Parameters

fileID	The ASMDFile for which the information is desired.
fileSys	The ASFileSys that fileID was opened through.
pFile	<i>(Filled by the method, may be NULL)</i> The ASFile representing fileID within fileSys .

Return Value

true if **fileID** is determined to be a valid file opened through **fileSys**, **false** otherwise.

Header File

ASCalls.h

Related Methods

[ASFileGetFileSys](#)
[ASFileGetMDFile](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.



ASFileGetEOF

```
ASTFilePos ASFileGetEOF (ASFile file);
```

Description

Gets the current size of a file. Calls [ASFileSysGetEofProc](#).

Parameters

file	The ASFile whose size is obtained.

Return Value

The size of the file.

Known Exceptions

[fileErrIO](#)

Header File

ASCalls.h

Related Methods

[ASFileSetEOF](#)

[ASFileSetPos](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.



ASFileGetFileSys

```
ASFileSys ASFileGetFileSys (ASFile file);
```

Description

Gets the file system that **file** was opened through.

Parameters

file	The open file whose file system is obtained.

Return Value

The file's **ASFileSys**.

Header File

ASCalls.h

Related Methods

[ASFileGetFileSysByName](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

ASFileGetMDFile

```
ASBool ASFileGetMDFile (ASFile file, ASMDFile* pFileID,
ASFileSys* pFileSys);
```

Description

Given an **ASFile**, returns the **fileSys** and the **ASMDFile** identification in that **fileSys**. This call is needed for a file system in a plug-in to be able to call the inner routines in another file system.

Parameters

file	The ASFile for which the information is desired.
pFileID	<i>(Filled by the method, may be NULL)</i> The ASMDFile identifier associated with file .
pFileSys	<i>(Filled by the method, may be NULL)</i> The file system that this file was opened through.

Return Value

true if **file** is an open file, **false** otherwise.

Header File

ASCalls.h

Related Methods

[ASFileFromMDFile](#)
[ASFileGetFileSys](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

ASFileGetOpenMode

ASUns16 ASFileGetOpenMode (**ASFile** file);

Description

Gets the file access mode(s) specified for **file** when it was opened.

Parameters

file	The file in question.

Return Value

Returns a value corresponding to one or more **ASFileMode** used to access/create the file, as shown in the table. The values that can be returned include combinations of the following, OR'd with each other:

Return value from ASFileGetOpenMode
0 (created)
1 (readable)
2 (readable and writable)
8 (sequential access)
16 (local)

Header File

ASCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASFileGetPos

```
ASTFilePos ASFileGetPos (ASFile file);
```

Description

Gets the current seek position in a file. This is the position at which the next read or write will begin.

Parameters

file	The file in which to get the seek position.

Return Value

The current seek position.

Known Exceptions

[fileErrIO](#)

Header File

ASCalls.h

Related Methods

[ASFileSetPos](#)

[ASFileRead](#)

[ASFileWrite](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



ASFileGetURL

```
char* ASFileGetURL (ASFile file);
```

Description

Returns the URL associated with **file**. It is the caller's responsibility to release the memory associated with the returned string using **ASfree**.

Parameters

file	The file in question.

Return Value

A buffer containing the URL or **NULL** if it could not be determined.

Header File

ASCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASFileHardFlush

```
ASErrorCode ASFileHardFlush (ASFile aFile);
```

Description

Causes a hard flush on a file. A “hard flush” means that the file is flushed to the physical destination. For example, if a WebDAV-based file is opened, [ASFileFlush](#) only flushes changes to the local cached version of the file. **ASFileHardFlush** would flush changes all the way to the WebDAV server.

Parameters

aFile	The file that is flushed.

Return Value

Zero if the operation succeeded; -1 if there was an error.

Known Exceptions

[fileErrIO](#)

Header File

ASCalls.h

Related Methods

[ASFileFlush](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

ASFileHasOutstandingMReads

```
ASBool ASFileHasOutstandingMReads (ASFile fN) ;
```

Description

Determines whether there are any outstanding multibyte range requests for a file. A document can have outstanding mreads when it was opened in a browser, Acrobat requested some byte ranges, and the byte ranges have not yet arrived.

Parameters

file	The file in question.

Return Value

true if the file has outstanding mreads, **false** otherwise.

Header File

ASCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



ASFileIsSame

```
ASBool ASFileIsSame (ASFile file, ASPathName path, ASFileSys
fileSys);
```

Description

Performs a comparison between **file** and **path** to determine if they represent the same file. This method will return **false** if **file** was not opened through the **fileSys** file system.

NOTE: Adobe does not guarantee that this method will work on all file systems.

Parameters

file	The file in question.
path	The ASPathName in question.
fileSys	The file system from which path was obtained.

Return Value

false if the comparison fails, **true** otherwise.

Header File

ASCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASFilePushData

```
void ASFilePushData (ASFile file, const char* buffer,  
ASTFilePos offset, ASTArraySize length);
```

Description

Sends data from a file system implementation to an [ASFile](#). The data may be for an multi-read request call, or may be unsolicited.

This method can only be called from within file system implementation. It must not be called by clients of the [ASFile](#), for example, by a caller that acquired the file with [ASFileSysOpenFile](#).

Parameters

file	The file to which data is sent.
buffer	The data being pushed.
offset	Byte offset into file at which the data should be written.
length	The number of bytes held in buffer .

Return Value

None

Known Exceptions

[fileErrGeneral](#)

Header File

ASCalls.h

Related Methods

[ASFileRead](#)
[ASFileWrite](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in [PIRequir.h](#)) is set to **0x00020002** or higher.



ASFileRead

```
ASArraySize ASFileRead (ASFile file, char* buffer,
ASArraySize count);
```

Description

Reads data from a file, beginning at the current seek position.

Parameters

file	The file from which data is read.
buffer	<i>(Filled by the method)</i> A buffer into which data is written. The buffer must be able to hold at least count bytes.
count	The number of bytes to read.

Return Value

The number of bytes actually read from the file.

Known Exceptions

```
fileErrIO
fileErrUserRequestedStop
fileErrBytesNotReady
fileErrIOTimeout
fileErrGeneral
```

Header File

ASCalls.h

Related Methods

```
ASFileSetPos
ASFileWrite
```

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



ASFileReopen

```
ASErrorCode ASFileReopen (ASFile file, ASFileMode mode);
```

Description

Attempts to reopen a file using the specified read/write mode. On some platforms, this may result in the file being closed and then reopened, and thus some error conditions may leave **file** invalid.

Parameters

file	The file to reopen.
mode	An open-mode value as specified for ASFileMode .

Return Value

Zero if the operation was successful, otherwise some file system/platform dependent error code is returned.

Header File

ASCalls.h

Related Methods

[ASFileSysOpenFile](#)
[ASFileClose](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

NOTE: The file mode and return types changed in **0x00060000**.

ASFileSetEOF

```
void ASFileSetEOF (ASFile file, ASInt32 newFileSize);
```

Description

Changes the size of a file. The new size may be larger or smaller than the original size. This method may raise file system/platform specific exceptions.

Parameters

file	The file whose size is changed.
newFileSize	The new size of file .

Return Value

[fileErrIO](#)

Header File

ASCalls.h

Related Methods

[ASFileCanSetEOF](#)

[ASFileGetEOF](#)

[ASFileGetPos](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



ASFileSetMode

```
ASFlagBits ASFileSetMode (ASFile file, ASFlagBits modeValue,
ASFlagBits modeMask);
```

Description

Gets and/or sets the mode flags for a file. Pass 0 for **modeValue** and **modeMask** to simply get the current mode flags.

Parameters

file	The file for which to get and/or set the mode.
modeValue	The mode flag values to get or set, which are described in ASFileMode Flags .
modeMask	Mask for the mode flags to get or set.

Return Value

The previous value of the mode, or 0 if the file system does not support this operation.

NOTE: This operation is primarily intended for slow file systems such as the internet, where there can potentially be an appreciable wait between requesting and retrieving bytes.

Header File

ASCalls.h

Related Methods

[ASFileRead](#)
[ASFileSysOpenFile](#)

Example

```
/* Get the current mode */
ASUns32 currentMode;
currentMode = ASFileSetMode(aFile, 0, 0);
/* Set the mode */
ASFileSetMode(aFile,
    kASFileModeDoNotYieldIfBytesNotReady,
    kASFileModeDoNotYieldIfBytesNotReady);
/* Clear the mode */
ASFileSetMode(aFile, 0,
    kASFileModeDoNotYieldIfBytesNotReady);
```

Product

reader, base, pro, pdf

**Availability**

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.



ASFileSetPos

```
void ASFileSetPos (ASFile file, ASTFilePos pos);
```

Description

Seeks to the specified position in a file. This is the position at which the next read or write will begin.

Parameters

file	The file in which to seek.
pos	The position to seek.

Return Value

None

Known Exceptions

[fileErrIO](#)

Header File

ASCalls.h

Related Methods

[ASFileGetPos](#)

[ASFileRead](#)

[ASFileWrite](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



ASFileWrite

```
ASArraySize ASFileWrite (ASFile file, const char* buffer,  
ASArraySize count);
```

Description

Writes data to a file, beginning at the current seek position.

Parameters

file	The file to which data is written.
buffer	A buffer holding the data that is to be written. The buffer must be able to hold at least count bytes.
count	The number of bytes to write.

Return Value

The number of bytes actually written to the file.

Known Exceptions

[fileErrIO](#)
[fileErrWrite](#)

Header File

ASCalls.h

Related Methods

[ASFileRead](#)
[ASFileSetPos](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



ASFileSys

ASFileGetFileSysByName

```
ASFileSys ASFileGetFileSysByName (ASAtom name);
```

Description

Gets the file system that was registered with the specified name.

Parameters

name	The ASAtom corresponding to the name of the file system to obtain.

Return Value

The file system, otherwise **NULL** if no matching file system was found.

Header File

ASCalls.h

Related Methods

[ASFileGetFileSys](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

ASFileRegisterFileSys

```
ASBool ASFileRegisterFileSys (ASExtension extension,
ASFileSys fileSys);
```

Description

Allows an implementor to provide a file system for use by external clients. An external client can locate the file system using [ASFileGetFileSysByName](#). **fileSys** provides its name via the [ASFileSysGetFileSysNameProc](#) callback. This method returns **false** if a file system with the same name is already registered.

Parameters

extension	The gExtensionID of the plug-in registering the fileSys .
fileSys	The fileSys being registered.

Return Value

true if **fileSys** is successfully registered, **false** otherwise.

Header File

ASCalls.h

Related Methods

[ASFileUnregisterFileSys](#)
[ASFileGetFileSysByName](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

ASFileSysAcquireFileSysPath

```
ASPathName ASFileSysAcquireFileSysPath (ASFileSys oldFileSys,
ASPathName oldPathName, ASFileSys newFileSys);
```

Description

Converts an **ASPathName** from one file system to another. Returns an **ASPathName** acquired through **newFileSys** that refers to an image (possibly cached) of the file in **oldFileSys**. Use this call to get a local file that is an image of a remote file (in a URL file system, for example). This is needed by programs such as the Movie Player, because they can only work from local file-system files. The returned **ASPathName** may be a reference to a cache, so the file should be treated as read-only.

Because of the possibility of cache flushing, you must hold a copy of the remote file's **ASPathName** for the duration of use of the local file.

Do not remove the local file copy, since the **newFileSys** system does not know about the linkage to the remote (**oldFileSys**) file!

The source file does not have to be open.

This call is handled by **oldFileSys** if **oldFileSys** contains the appropriate procedure. Otherwise it is handled by copying the file. The source file is closed at the end of the copy if it was not open prior to the call.

It is the caller's responsibility to release the **ASPathName** using **ASFileSysReleasePath** when it is no longer needed.

Parameters

oldFileSys	(May be NULL) The file system from which oldPathName was obtained. Pass NULL to use the default file system.
oldPathName	The ASPathname in the current file system.
newFileSys	(May be NULL) The file system to which the oldPathName is converted. Pass NULL to use the default file system.

Return Value

The **ASPathName** in **newFileSys** or **NULL** if one can not be made.

Known Exceptions

ERR_NOMEMORY
fileErrIO
fileErrUserRequestedStop
fileErrBytesNotReady
fileErrIOTimeout
fileErrGeneral
fileErrWrite

**Header File**

ASCalls.h

Related Methods

[ASFileSysCreatePathName](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.



ASFileSysAcquireParent

```
ASPathName ASFileSysAcquireParent (ASFileSys fileSys,
ASPathName pathName);
```

Description

Returns the parent folder of the file system object associated with **pathName**. The following rules apply in the default file systems:

- **pathName** may be associated with either a file or a folder.
- the file system object associated with **pathName** need not exist.

It is the caller's responsibility to release the returned **ASPathName**.

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The ASPathName .

Return Value

The **ASPathName** associated with the parent folder. Will return **NULL** if the parent could not be returned.

Known Exceptions

```
genErrNoMemory
fileErrIO
fileErrUserRequestedStop
fileErrBytesNotReady
fileErrIOTimeout
fileErrGeneral
fileErrWrite
```

Header File

ASCalls.h

Related Methods

[ASFileSysCreatePathName](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASFileSysAcquirePlatformPath

```
ASInt32 ASFileSysAcquirePlatformPath (ASFileSys FileSys,
ASPathName PathName, ASAtom platformPathType,
ASPlatformPath *platformPath);
```

Description

Returns a platform-specific file-system representation of the specified path, according to the specified type, wrapped in an allocated **ASPlatformPath** object. Calls **ASFileSysAcquirePlatformPathProc**.

This method creates an equivalent platform-specific type (such as **FSRef** on Mac) from an **ASPathName**. Use **ASFileSysCreatePathName** for the reverse, to create an equivalent **ASPathName** from a platform-specific type.

In previous releases, you could cast an **ASPathName** to an **FSSpec** (for example), but that does not work in the Acrobat 6.0 SDK (because of changes to accommodate long, Unicode filenames on Mac OS X). When developing for Mac OS, use this call to get an **FSSpec** from an **ASPathName** safely on Mac OS X, without casting. However, it is recommended that you transition to using newer types such as **FSRef** to be compatible with OS X filenames, or change to using all **ASFileSys** methods.

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The ASPathName in the file system specified by fileSys .
platformPathType	The platform path type, one of the following ASAtom values: Mac: • FSRefWithCFStringRef • FSSpec • CFURLRef • POSIXPath • FSRef (pathName object must exist) Windows/UNIX: • Cstring
platformPath	(Filled by the method) The new platform path object. Always free this object with ASFileSysReleasePlatformPath when done.

Return Value

0 if the operation was successful, otherwise a nonzero error code.

Header File

ASCalls.h

Related Methods

[ASFileSysReleasePlatformPath](#)

Example

```
/* Demonstrates how to get one particular type of platform path:
   an FSSpec on Mac OS. */
ASPlatformPath platformPath;
```

******* define path, make atom of FSSpec**

```
ASInt32 err = ASFileSysAcquirePlatformPath (NULL, path***, FSSpec***,
      &platformPath);
if (platformPath != NULL {
    FSSpec *fsp = ASPlatformPathGetFSSpecPtr(platformPath);
    /* use FSSpec */
    ASFileSysReleasePlatformPath (NULL, platformPath);
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



ASFileSysCanPerformOpOnItem

```
ASInt32 ASFileSysCanPerformOpOnItem (ASFileSys fileSys,
ASPathName pathName, const char *op);
```

Description

Tests whether a given operation can be performed on a particular file. Calls the **canPerformOpOnItem** procedure registered for the **ASFileSysRec**, which determines whether the operation is one of the filesystem-defined operation strings for which there is a handler.

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The ASPathName of the file.
op	The name of the operation to test. A filesystem-defined string handled by ASFileSysCanPerformOpOnItemProc .

Return Value

Returns **asGenErrNoError** if the operation can be performed on the item, or an error indicating why the operation would fail.

Header File

ASExtraCalls.h

Related Methods

[ASFileSysPerformOpOnItem](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



ASFileSysConvertCabToItemProps

```
ASInt32 ASFileSysConvertCabToItemProps
(const ASFileSysItemPropsRec *props, ASCab theCab);
```

Description

Converts a set of item properties from the **ASCab** format to the **ASFileSysItemPropsRec** format.

Parameters

props	<i>(Filled by the method)</i> The item properties structure.
theCab	Properties describing the object, in cabinet format.

Return Value

0 if no error was encountered; otherwise an error code is returned.

Known Exceptions

[genErrBadParm](#)
[genErrMethodNotImplemented](#)

Header File

ASExtraCalls.h

Related Methods

[ASFileSysConvertItemPropsToCab](#)
[ASFileSysGetItemProps](#)
[ASFileSysGetItemPropsAsCab](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASFileSysConvertItemPropsToCab

```
ASInt32 ASFileSysConvertItemPropsToCab (ASCab theCab,
const ASFileSysItemPropsRec *props);
```

Description

Converts a set of item properties from the **ASFileSysItemPropsRec** format to the **ASCab** format. The **ASCab** has the following potential entries:

Key Name	Type
isThere	ASBool
type	ASInt32
isHidden	ASBool
isReadOnly	ASBool
creationDate	char * (PDF style date string)
modDate	char * (PDF style date string)
fileSizeHigh	ASUns32
fileSizeLow	ASUns32
folderSize	ASInt32
creatorCode	ASUns32
typeCode	ASUns32
versionMajor	ASUns32
versionMinor	ASUns32
isCheckedOut	ASBool
isPublished	ASBool

Parameters

theCab	(Filled by the method) Properties describing the object, in cabinet format.
props	The item properties structure.

Return Value

0 if no error was encountered; otherwise an error code is returned.

**Known Exceptions**

[genErrBadParm](#)
[genErrMethodNotImplemented](#)

Header File

ASExtraCalls.h

Related Methods

[ASFileSysConvertCabToItemProps](#)
[ASFileSysGetItemProps](#)
[ASFileSysGetItemPropsAsCab](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



ASFileSysCopyPath

```
ASPathName ASFileSysCopyPath (ASFileSys fileSys,
ASPathName pathName);
```

Description

Generates and copies the specified [ASPathName](#) (but does not copy the file specified by the pathname). The [ASPathName](#) must have been obtained through the specified file system. This method may be used regardless of whether the file specified by **pathname** is open.

It is the caller's responsibility to release the [ASPathName](#) using [ASFileSysReleasePath](#) when it is no longer needed.

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The ASPathName to copy.

Return Value

A copy of **pathName**.

Header File

ASCalls.h

Related Methods

[ASFileSysReleasePath](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



ASFileSysCreateFolder

```
ASErrorCode ASFileSysCreateFolder (ASFileSys fileSys,
ASPathName path, ASBool recurse);
```

Description

Creates an empty folder at the specified **pathName**.

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
path	The path of the folder to create.
recurse	Recursively create the parent folder if necessary.

Return Value

0 if the operation was successful, otherwise returns a nonzero platform-dependent error code.

Known Exceptions

```
genErrMethodNotImplemented
fileErrFNF
```

Header File

ASCalls.h

Related Methods

```
ASFileSysRemoveFolder
```

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASFileSysCreatePathName

```
ASPathName ASFileSysCreatePathName (const ASFileSys fileSys,
ASAtom pathSpecType, const void* pathSpec,
const void* optionalPath);
```

Description

Creates an **ASPathName** based on the input type and **pathSpec**. Each **fileSys** implementation must publish the input types that it accepts.

It is the caller's responsibility to release the **ASPathName** using **ASFileSysReleasePath** when it is no longer needed.

Developers should consider using the simpler helper macros instead of using the call directly. See the Related Macros section below.

Parameters

fileSys	(May be NULL) The ASFileSys in which you are trying to create an ASPathName . Pass NULL to use the default file system.
pathSpecType	An ASAtom specifying the data type in pathSpec, as follows: ● Cstring: Accepted by the default file system on all platforms. pathSpec is a null-terminated char*. —On the Mac it must be an absolute HFS path separated by colons, as in VolumeName:Folder:file.pdf. —On Windows the path may be absolute, as in C:\folder\file.pdf or relative as in ...\folder\file.pdf. —On Unix the path may be absolute as in /folder/file.pdf or relative as in .../folder/file.pdf. ● FSSpec: Accepted by the default file system on the Mac. pathSpec is a valid FSSpec_Ptr. Discouraged on Mac OS X due to its limitations in handling long Unicode file and folder names. ● FSRef: Accepted by the default file system on the Mac. pathSpec is a valid FSSpec_Ptr (the referenced file or folder must exist). ● FSRefWithCFStringRef: Accepted by the default file system on the Mac. pathSpec is a valid FSSpec_Ptr (the referenced file or folder must exist) and optionalPath is a valid CFStringRef. ● CFURLRef: Accepted by the default file system on the Mac. pathSpec is a valid CFURLRef.

pathSpecType (cont'd)	● POSIXPath: Accepted by the default file system on the Mac. pathSpec is a pointer to a valid C string containing a valid POSIX path (UTF-8 encoding). ● SFReply: Deprecated, do not use. Formerly accepted by the default file system on the Mac. ● DIPath: Accepted by the default file system on Windows and Mac. pathSpec is a device-independent path as documented in the PDF Reference that can contain an absolute or relative path. If a relative path is used, the method evaluates that path against an ASPathName passed in the optionalPath parameter. ● DIPathWithASText: Same as DIPath except that the device-independent path passed in pathSpec is an ASText. ● FolderPathName: Accepted by the default file system on Windows and Mac. pathSpec is an ASPathName that contains the path of a folder. optionalPath is a C string that contains the name of the file. The returned ASPathName contains the result of appending optionalPath to pathSpec. ● FolderPathNameWithASText: Same as FolderPathName except that the name of the file or folder in optionalPath is an ASText.
pathSpec	The file specification from which to create an ASPathName . Relative paths are evaluated from the directory containing the executable (if used with the PDF Library), or the directory containing Acrobat (if used in a plug-in).
optionalPath	See pathSpecType parameter description.

Return Value

The newly created pathname, or **NULL** on failure.

Known Exceptions

[genErrBadParm](#) on Windows if the **pathSpecType** is not recognized.
[genErrMethodNotImplemented](#)

Header File

ASCalls.h

Related Methods

[ASFileSysCopyPath](#)

Related Macros

[ASFileSysCreatePathFromCString](#)
[ASFileSysCreatePathFromDIPath](#)
[ASFileSysCreatePathFromFSSpec](#)
[ASFileSysCreatePathWithFolderName](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

ASFileSysDestroyFolderIterator

```
void ASFileSysDestroyFolderIterator (ASFileSys fileSys,
ASFolderIterator folderIter);
```

Description

Releases the resources associated with **folderIter**.

Parameters

fileSys	(May be NULL) The file system from which the iteration was started. Pass NULL to use the default file system.
folderIter	An ASFolderIterator object returned from a previous call to ASFileSysFirstFolderItem .

Return Value

None

Header File

ASCalls.h

Related Methods

[ASFileSysFirstFolderItem](#)
[ASFileSysNextFolderItem](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASFileSysDisplayASTextFromPath

```
ASErrorCode ASFileSysDisplayASTextFromPath (ASFileSys fileSys,
ASPathName pathName, ASText displayText);
```

A Description

Returns a user-friendly representation of a path as a text object. Calls [ASFileSysDisplayASTextFromPathProc](#).

NOTE: Supersedes [ASFileSysDisplayStringFromPath](#) in Acrobat 6.0.

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The ASPathName in question.
displayText	(Filled by method) The text object containing the display representation of the path.

Return Value

0 if the operation was successful, otherwise returns a nonzero platform-dependent error code.

Header File

ASCalls.h

Related Methods

[ASFileSysDisplayStringFromPath](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASFileSysDisplayStringFromPath

```
char* ASFileSysDisplayStringFromPath (ASFileSys fileSys,
ASPathName pathName);
```

Description

Returns a user-friendly representation of a path. It is the caller's responsibility to release the memory associated with the returned string using [ASfree](#).

NOTE: In Acrobat 6.0, this method can only be used to get host encoding. For any other encoding, use [ASFileSysDisplayASTextFromPath](#).

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The ASPathName in question.

Return Value

A buffer containing the display string, or **NULL** if this operation is not supported by the file system or some error occurred. For example:

```
Windows: "C:\Folder\File"
Mac: "Hard Disk:Folder:File"
UNIX: "/Folder/File"
```

Header File

ASCalls.h

Related Methods

[ASFileSysDisplayASTextFromPath](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASFileSysDIPathFromPath

```
char* ASFileSysDIPathFromPath (ASFileSys fileSys,
ASPathName pathName, ASPathName relativeToThisPath);
```

Description

Converts a filename, specified as an **ASPathName**, to a device-independent pathname. It is the caller's responsibility to free the memory associated with the returned string using **ASfree**.

NOTE: This method can only be used to get host encoding. For any other encoding, use **ASFileSysDIPathFromPathEx** in Acrobat 6.0.

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The ASPathName to convert.
relativeToThisPath	(May be NULL) The pathname relative to which the device-independent pathname is specified. If NULL , the device-independent pathname will be an absolute, not a relative, pathname.

Return Value

Device-independent pathname corresponding to the parameter values supplied, or **NULL** if the operation is not supported by the file system.

See Section 3.10 in the [PDF Reference](#) for a description of the device-independent pathname format. This pathname may not be understood on another platform since drive specifiers may be prepended.

Header File

ASCalls.h

Related Methods

[ASGetDefaultFileSys](#)
[ASFileSysPathFromDIPath](#)

Example

```
char* temp;
temp = ASFileSysDIPathFromPath(
    ASGetDefaultFileSys(), path, NULL);
....
ASfree(temp);
```



Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

ASFileSysDIPathFromPathEx

```
ASErrorCode ASFileSysDIPathFromPathEx (ASFileSys fileSys,
ASPathName pathName, ASPathName relativeToThisPath,
ASText diPathText);
```

Description

Converts a filename, specified as an **ASPathName**, to a device-independent pathname, returned as an **ASText** object. Calls **ASFileSysDIPathFromPathExProc**.

NOTE: Supersedes **ASFileSysDIPathFromPath** in Acrobat 6.0.

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The ASPathName to convert.
relativeToThisPath	(May be NULL) The pathname relative to which the device-independent pathname is specified. If NULL , the device-independent pathname will be an absolute, not a relative, pathname.
diPathText	(Filled by the method) The ASText object to contain the device-independent path. Must be allocated and freed by the client.

Return Value

0 if the operation was successful, otherwise returns a nonzero platform-dependent error code.

Header File

ASCalls.h

Related Methods

```
ASGetDefaultFileSys
ASFileSysPathFromDIPathEx
```

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

ASFileSysFirstFolderItem

```
ASFolderIterator ASFileSysFirstFolderItem (ASFileSys fileSys,
ASPathName folderPath, ASFileSysItemProps itemProps,
ASPathName *itemPath);
```

Description

Creates an iterator which can be used to enumerate all objects inside the specified folder, and also returns the properties of the first item found in the folder. The iteration can be continued by passing the returned [ASFolderIterator](#) to [ASFileSysNextFolderItem](#).

Both **itemProps** and **itemPath** are optional and may be **NULL** if you are not interested in that information.

The client is obligated to eventually free the resources associated with **ASFolderIterator** by calling [ASFileSysDestroyFolderIterator](#).

NOTE: The order in which items are enumerated is implementation-dependent. In particular, note that items will probably not be iterated in alphabetic order.

NOTE: If items are added to or removed from a folder during iteration the results are implementation-dependent.

Parameters

fileSys	(May be NULL) The file system from which folderPath was obtained. Pass NULL to use the default file system.
folderPath	The path associated with the target folder.
itemProps	(Filled by the method, may be NULL) Properties structure describing the first object iterated.
itemPath	(Filled by the method, may be NULL) A newly-allocated ASPathName associated with the first object (which the client must free). Contains an absolute path.

Return Value

A valid **ASFolderIterator** object if **folderPath** contained any files. **NULL** will be returned if the folder is empty or the operation is not supported by the file system.

Known Exceptions

[genErrBadParm](#)

Windows default file system may raise: [fileErrFNF](#), [asFileErrNotADir](#) or [ERR_NOMEMORY](#)

Header File

ASCalls.h

Related Methods

[ASFileSysNextFolderItem](#)

[ASFileSysDestroyFolderIterator](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.



ASFileSysFlushVolume

```
ASErrorCode ASFileSysFlushVolume (ASFileSys fileSys,
ASPathName pathName);
```

Description

Flushes the volume on which the specified file resides. This ensures that any data written to the system for the volume containing **pathName** is flushed out to the physical volume (equivalent to the Macintosh **FlushVol** or to the UNIX **sync**).

Only the Mac OS default file system implements the callback associated with this method. This is a no-op on Windows and Unix.

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The ASPathName the volume information is obtained from.

Return Value

0 if the operation was successful, otherwise returns a nonzero platform-dependent error code.

Header File

ASCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASFileSysGetItemProps

```
ASErrorCode ASFileSysGetItemProps (ASFileSys fileSys,
ASPathName pathName, ASFileSysItemProps props);
```

Description

Populates an **ASFileSysItemProps** record with a full description of the file system object associated with **pathName**. Calls **ASFileSysGetItemPropsProc**.

NOTE: The method clears the memory associated with **itemProps**, so the caller need not. However, the caller must explicitly set the **props->size** field of the **ASFileSysItemProps** structure before calling this method.

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The ASPathName associated with the object.
props	(Filled by the method) Properties structure describing the object. The size field must be set on input.

Return Value

0 if no error was encountered; otherwise an error code is returned. If an error code is returned, **props** will not be filled with valid values. If no file system object is present, an error will not be reported—instead, the **props.isThere** field will be false.

Known Exceptions

```
genErrBadParm
genErrMethodNotImplemented
```

Header File

ASCalls.h

Related Methods

```
ASFileSysConvertCabToItemProps
ASFileSysConvertItemPropsToCab
ASFileSysGetItemPropsAsCab
```

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASFileSysGetItemPropsAsCab

```
ASInt32 ASFileSysGetItemPropsAsCab (ASFileSys fileSys,
ASPathName pathName, ASCab theCab);
```

Description

Gets a full description of the file system object associated with **pathName**, returning the item properties in the **ASCab** format. Calls **ASFileSysGetItemPropsAsCabProc**

If the **ASCab** has no keys on entry, every property known is filled in. If it is not empty, only properties corresponding to keys in the **ASCab** are filled in. Keys that do not map to a property of the object are removed. The **ASCab** has the following potential entries:

Key Name	Type
isThere	ASBool
type	ASInt32
isHidden	ASBool
isReadOnly	ASBool
creationDate	char * (PDF style date string)
modDate	char * (PDF style date string)
fileSizeHigh	ASUns32
fileSizeLow	ASUns32
folderSize	ASInt32
creatorCode	ASUns32
typeCode	ASUns32
versionMajor	ASUns32
versionMinor	ASUns32
isCheckedOut	ASBool
isPublished	ASBool

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The ASPathName associated with the object.
theCab	(Filled by the method) Properties describing the object, in cabinet format.

Return Value

0 if no error was encountered; otherwise an error code is returned. If an error code is returned, **theCab** is not filled with valid values. If the path name does not point to an object on the file system, returns **asFileErrFNF** and a valid **ASCab** with **isThere** set to **false**.

Known Exceptions

[genErrBadParm](#)
[genErrMethodNotImplemented](#)

Header File

ASExtraCalls.h

Related Methods

[ASFileSysConvertCabToItemProps](#)
[ASFileSysConvertItemPropsToCab](#)
[ASFileSysGetItemProps](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASFileSysGetNameFromPath

```
ASErrorCode ASFileSysGetNameFromPath (ASFileSys fileSys,
ASPathName pathName, char* buffer, ASArraySize maxLength);
```

Description

Extracts the filename (including extension) from path.

NOTE: In Acrobat 6.0, this method can only be used to get host encoding. For any other encoding, use [ASFileSysGetNameFromPathAsASText](#).

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The ASPathName associated with the file in question.
buffer	(Filled by the method) Buffer used to store the filename.
maxLength	Maximum number of bytes that buffer can hold.

Return Value

0 if the operation was successful, otherwise returns a nonzero platform-dependent error code. The buffer is returned as a host-encoded C string.

Known Exceptions

[fileErrGeneral](#)

Header File

ASCalls.h

Related Methods

[ASFileSysGetNameFromPathAsASText](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASFileSysGetNameFromPathAsASText

```
ASErrorCode ASFileSysGetNameFromPathAsASText
(ASFileSys fileSys, ASPathName pathName, ASText name);
```

Description

Extracts the filename (including extension) from path as an [ASText](#) object. Supersedes [ASFileSysGetNameFromPath](#) in Acrobat 6.0.

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The ASPathName associated with the file in question.
name	(Filled by the method) The text object containing the filename.

Return Value

0 if the operation was successful, otherwise returns a nonzero platform-dependent error code.

Known Exceptions

[fileErrGeneral](#)

Header File

ASCalls.h

Related Methods

[ASFileSysGetNameFromPath](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASFileSysGetStorageFreeSpace

```
ASUns32 ASFileSysGetStorageFreeSpace (ASFileSys fileSys,  
ASPathName pathName) ;
```

Description

Gets the amount of free space on the volume containing **pathName**.

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The ASPathName in question.

Return Value

The amount of free space, in bytes, 0 otherwise. Because the free space is returned as an **ASUns32** it is limited to 4 GB.

Header File

ASCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASFileSysGetTempPathName

```
ASPathName ASFileSysGetTempPathName (ASFileSys fileSys,  
ASPathName siblingPath);
```

Description

Returns an unique pathname suitable for use in creating temporary files. It is the caller's responsibility to release the returned object using [ASFileSysReleasePath](#).

If siblingPath is non-**NULL**, the returned **ASPathName** is created at the same folder level as this path. Otherwise the standard temporary file location is used.

Parameters

fileSys	(May be NULL) The file system from which siblingPath was obtained. Pass NULL to use the default file system.
siblingPath	(May be NULL) An ASPathName indicating the desired location of the temporary pathname. The returned ASPathName is created at the same folder level as this path.

Return Value

The [ASPathName](#) if the operation was successful, **NULL** otherwise.

Header File

ASCalls.h

Related Methods

[ASFileSysReleasePath](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASFileSysGetTypeAndCreator

```
void ASFileSysGetTypeAndCreator (ASFileSys fileSys,  
ASPathName path, unsigned long *type, unsigned long *creator)
```

Description

Gets the type and creator of the file specified by the path. See [Type/Creator Codes](#).

NOTE: Only meaningful for the Macintosh default file system. Windows and UNIX always return 0 for both type and creator.

Parameters

fileSys	The file system containing the file for which the type and creator are needed.
path	The pathname of the file.
type	<i>(Filled by method)</i> The type of the file.
creator	<i>(Filled by method)</i> The creator of the file.

Return Value

None.

Header File

ASCalls.h

Related Methods

[ASFileSysSetTypeAndCreator](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

ASFileSysNextFolderItem

```
ASBool ASFileSysNextFolderItem (ASFileSys fileSys,
ASFolderIterator folderIter, ASFileSysItemProps itemProps,
ASPathName *itemPath);
```

Description

Continues the iteration process associated with the **ASFolderIterator** object.

Both **itemPath** and **itemProps** are optional and may be **NULL** if you are not interested in that information.

Parameters

fileSys	(May be NULL) The file system with which the iteration was started. Pass NULL to use the default file system.
folderIter	An ASFolderIterator object returned from a previous call to ASFileSysFirstFolderItem .
itemProps	(Filled by the method, may be NULL) Properties structure describing the next object in the iteration.
itemPath	(Filled by the method, may be NULL) A newly-allocated ASPathName associated with the object (which the client must free). Contains an absolute path on Windows and UNIX.

Return Value

true if another object was found, **false** otherwise.

Known Exceptions

```
genErrBadParm
fileErrGeneral
ERR_NOMEMORY
```

Header File

ASCalls.h

Related Methods

```
ASFileSysFirstFolderItem
ASFileSysDestroyFolderIterator
```

Product

reader, base, pro, pdf

**Availability**

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

ASFileSysOpenFile

```
ASErrorCode ASFileSysOpenFile (ASFileSys fileSys,
ASPathName pathName, ASFileMode mode, ASFile* pFile);
```

Description

Attempts to open a file in the specified file system, in the specified read/write/create mode. If the file is already open, the existing file handle is returned. The caller retains ownership of **pathName**.
In Mac OS, when this method creates a file, the file's creator is set to 'CARO' and its type is set to 'PDF ' (with a 'space' after PDF).

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The pathname of the file to open.
mode	An open-mode value as specified for ASFileMode .
pFile	(Filled by the method) The ASFile that was opened.

Return Value

0 if the operation was successful, otherwise returns a nonzero error code. The error is platform- and file-system specific.

Windows	Macintosh
Returns fileErrWrPerm if trying to open read-only file with write permissions.	Returns fileErrFNF if trying to open a file for reading that does not exist.
ErrSysXtnMgr (GetLastError ()) — Platform-specific error (any error condition that CreateFile may use).	ErrSysMDSystem (iErr) — Platform-specific error (any error that FSpCreate , FSpSetFInfo , FSpOpenRF , FSpOpenDF , or SetFPos may use).
Returns fileErrGeneral if the developer passed in an invalid ASPathName .	

Known Exceptions

genErrNoError

Header File

ASCalls.h

**Related Methods**

[ASFileClose](#)
[ASFileReopen](#)
[ASGetDefaultFileSys](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.



ASFileSysPathFromDIPath

```
ASPathName ASFileSysPathFromDIPath (ASFileSys fileSys,
char* diPath, ASPathName relativeToThisPath);
```

Description

Converts a device-independent pathname to an **ASPathName**. This method can only be used for files that already exist (that is, it cannot be used to create a placeholder pathname for files that a plug-in intends to create in the future).

It is the caller's responsibility to release the returned **ASPathName**.

NOTE: In Acrobat 6.0, use **ASFileSysPathFromDIPathEx** instead for anything other than host encoding.

Parameters

fileSys	(May be NULL) The file system that the ASPathName will be created within. Pass NULL to use the default file system.
diPath	The device-independent pathname to convert. See Section 3.10 in the PDF Reference for a description of the device-independent pathname format. This pathname may not be understood on another platform since drive specifiers may be prepended. In Windows, you cannot specify a UNC pathname. You must have a file mounted on the file server. For example, the following path is valid: /f/dirname/file.pdf where f is \\server\people. The following does not work: /server/people/dirname/file.pdf
relativeToThisPath	Pathname relative to which diPath is interpreted. If NULL , diPath is interpreted as an absolute pathname, not a relative pathname.

Return Value

ASPathName corresponding to the parameter values supplied. Returns **NULL** if **diPath** cannot be converted to an **ASPathName** or if the specified file does not already exist.

Known Exceptions

genErrNoMemory

Header File

ASCalls.h

Related Methods

[ASFileSysDIPathFromPath](#)

[ASFileSysReleasePath](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

ASFileSysPathFromDIPathEx

ASPathName ASFileSysPathFromDIPathEx (**ASFileSys** fileSys, **ASText** diPath, **ASPathName** relativeToThisPath);

Description

Converts a device-independent pathname in an **ASText** object to an **ASPathName**. This method can only be used for files that already exist (that is, it cannot be used to create a placeholder pathname for files that a plug-in intends to create in the future).

It is the caller's responsibility to release the returned **ASPathName**.

NOTE: Supersedes **ASFileSysPathFromDIPath** in Acrobat 6.0.

Parameters

fileSys	(May be NULL) The file system that the ASPathName will be created within. Pass NULL to use the default file system.
diPath	The device-independent pathname to convert, as an ASText object. See Section 3.10 in the PDF Reference for a description of the device-independent pathname format. This pathname may not be understood on another platform since drive specifiers may be prepended. In Windows, you cannot specify a UNC pathname. You must have a file mounted on the file server. For example, the following path is valid: /f/dirname/file.pdf where <i>f</i> is \\server\people. The following does not work: /server/people/dirname/file.pdf
relativeToThisPath	Pathname relative to which diPath is interpreted. If NULL , diPath is interpreted as an absolute pathname, not a relative pathname.

Return Value

ASPathName corresponding to the parameter values supplied. Returns **NULL** if **diPath** cannot be converted to an **ASPathName**, for example if the specified file does not already exist.

Known Exceptions

genErrNoMemory

Header File

ASCalls.h

**Related Methods**[ASFileSysDIPathFromPathEx](#)[ASFileSysReleasePath](#)**Product**

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

ASFileSysPerformOpOnItem

```
ASInt32 ASFileSysPerformOpOnItem (ASFileSys fileSys,
ASPathName pathName, const char *op, ASCab params);
```

Description

Performs a specified operation on a particular file, passing specified parameters. Calls the **performOpOnItem** procedure registered for the **ASFileSysRec**.

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The ASPathName of the file.
op	The name of the operation to perform. A filesystem-defined string handled by ASFileSysPerformOpOnItemProc .
params	An ASCab object containing parameters to pass to the operation.

Return Value

Zero if the operation was successful, otherwise returns a nonzero platform-dependent error code.

Header File

ASExtraCalls.h

Related Methods

[ASFileSysCanPerformOpOnItem](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASFileSysReleasePath

```
void ASFileSysReleasePath (ASFileSys fileSys,
    ASPathName pathName) ;
```

Description

Decrements the internal reference count for **pathname** and disposes of the pathname (but not the file itself) if the reference count is zero. This does not result in any file-level operations, and is unaffected by whether there is an open file for this pathname.

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The ASPathName to release.

Return Value

None

Header File

ASCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

ASFileSysReleasePlatformPath

```
void ASFileSysReleasePlatformPath (ASFileSys fileSys,
ASPlatformPath platformPath) ;
```

Description

Releases the specified platform path object. Each call to [ASFileSysAcquirePlatformPath](#) should have a corresponding call to this method.

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
platformPath	The platform path object to release.

Return Value

None

Header File

ASCalls.h

Related Methods

[ASFileSysAcquirePlatformPath](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASFileSysRemoveFile

```
ASErrorCode ASFileSysRemoveFile (ASFileSys fileSys,
ASPathName pathName);
```

Description

Attempts to delete the file referred to by **pathName**.

NOTE: If a file is already open for this **pathName**, the semantics of **ASFileSysRemoveFile** are file system-dependent. Make sure you have closed all **ASFiles** for **pathName** before calling **ASFileSysRemoveFile**.

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
pathName	The file to delete.

Return Value

0 if the operation was successful, otherwise returns a nonzero platform-dependent error code.

Header File

ASCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



ASFileSysRemoveFolder

```
ASErrorCode ASFileSysRemoveFolder (ASFileSys fileSys,
ASPathName pathName) ;
```

Description

Deletes the folder at the specified **pathName** only if it is empty.

Parameters

fileSys	(May be NULL) The file system from which pathName was obtained. Pass NULL to use the default file system.
path	The path of the folder to remove.

Return Value

0 if the operation was successful, otherwise returns a nonzero platform-dependent error code.

Known Exceptions

[genErrMethodNotImplemented](#)

Header File

ASCalls.h

Related Methods

[ASFileSysCreateFolder](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASFileSysSetTypeAndCreator

```
void ASFileSysSetTypeAndCreator (ASFileSys fileSys,
ASPathName path, unsigned long type, unsigned long creator);
```

Description

Sets the type and creator of a file. See [Type/Creator Codes](#).

NOTE: As is the case for [ASFileSysGetTypeAndCreator](#), this method only applies to the Macintosh default file system. Windows and UNIX make this a no-op.

Parameters

fileSys	The file system for which the type and creator are needed.
path	The pathname of the file.
type	The type of the file.
creator	The creator of the file.

Return Value

None.

Header File

ASCalls.h

Related Methods

[ASFileSysGetTypeAndCreator](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



ASFileSysURLFromPath

```
char* ASFileSysURLFromPath (ASFileSys fileSys,
    ASPathName pathName);
```

Description

Returns the URL corresponding to **pathName**. It is the caller's responsibility to free the memory associated with the returned string using **ASfree**.

Parameters

fileSys	The file system from which path was obtained. Pass NULL to use the default file system.
path	The ASPathName in question.

Return Value

A buffer containing the URL, or **NULL** if some error occurred. The URL is in the standard "file://" URL style.

Header File

ASCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASFileUnregisterFileSys

```
ASBool ASFileUnregisterFileSys (ASExtension extension,
ASFileSys fileSys);
```

Description

Allows a **fileSys** to be unregistered. In general, a **fileSys** is only unregistered by the extension that registered it.

Parameters

extension	The gExtensionID of the plug-in un-registering fileSys .
fileSys	The ASFileSys to un-register.

Return Value

true if **fileSys** successfully unregistered, **false** if any there are any open files that were opened through **fileSys**.

Header File

ASCalls.h

Related Methods

[ASFileRegisterFileSys](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

ASGetDefaultFileSys

```
ASFileSys ASGetDefaultFileSys (void);
```

Description

Gets the default/standard file system implementation for a platform.

Parameters

None

Return Value

The platform's default file system.

Header File

ASCalls.h

Related Methods

[ASFileRegisterFileSys](#)
[ASPathFromPlatformPath](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to 0x00020000 or higher.



ASPathFromPlatformPath

```
ASPathName ASPathFromPlatformPath (void* platformPath);
```

Description

NOTE: This method was deprecated in version 5.0. Use **ASFileSysCreatePathName** instead.

Converts a platform-specific pathname to an **ASPathName**. It can create an **ASPathName** from a file path where the file does *not* already exist. It works for Windows UNC pathnames as well.

It is the caller's responsibility to release the returned **ASPathName**.

Parameters

platformPath	Pointer to a platform-specific pathname. In Windows and Unix, it is a null-terminated string containing the full pathname with the appropriate path separators for each platform.

Return Value

The **ASPathName** corresponding to **platformPath**.

Header File

ASCalls.h

Related Methods

ASFileSysReleasePath

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

ASPlatformPath

ASPlatformPathGetCFURLRefRecPtr

[CFURLRefRec_Ptr](#) ASPlatformPathGetCFURLRefRecPtr
([ASPlatformPath](#) path)

Description

Gets a platform path object in the form of a [CFURLRef](#) for Mac OS, if the [ASPlatformPath](#) object was acquired with this type in the `platformPathType` parameter of [ASFileSysAcquirePlatformPath](#).

NOTE: Do not release the returned value, or any member data of an [ASPlatformPath](#) directly; use [ASFileSysReleasePlatformPath](#) when finished with the object.

Parameters

<code>path</code>	The platform path.

Return Value

A pointer to a structure containing a [CFURLRef](#).

Header File

`ASCalls.h`

Related Methods

[ASFileSysAcquirePlatformPath](#)
[ASFileSysReleasePlatformPath](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

ASPlatformPathGetCstringPtr

[Cstring_Ptr](#) ASPlatformPathGetCstringPtr ([ASPlatformPath](#) path)

Description

Gets a platform path object in the form of a C string for Microsoft Windows or UNIX, if the [ASPlatformPath](#) object was acquired with this type in the `platformPathType` parameter of [ASFileSysAcquirePlatformPath](#).

NOTE: Applications should use this a read-only pointer; modifying the returned buffer can corrupt the [ASPlatformPath](#). Do not free the pointer.

NOTE: Do not release the returned value, or any member data of an [ASPlatformPath](#) directly; use [ASFileSysReleasePlatformPath](#) when finished with the object.

Parameters

<code>path</code>	The platform path.

Return Value

A pointer to a C string of a platform-specific path.

Header File

`ASCalls.h`

Related Methods

[ASFileSysAcquirePlatformPath](#)
[ASFileSysReleasePlatformPath](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

ASPlatformPathGetFSRefPtr

FSRef_Ptr ASPlatformPathGetFSRefPtr (**ASPlatformPath** path)

Description

Gets a platform path object in the form of an **FSRef** for Mac OS, if the **ASPlatformPath** object was acquired with this type in the **platformPathType** parameter of **ASFileSysAcquirePlatformPath**.

NOTE: Do not release the returned value, or any member data of an **ASPlatformPath** directly; use **ASFileSysReleasePlatformPath** when finished with the object.

Parameters

path	The platform path.

Return Value

A pointer to an **FSRef**.

Header File

ASCalls.h

Related Methods

ASFileSysAcquirePlatformPath
ASPlatformPathGetFSRefWithCFStringRefRecPtr
ASFileSysReleasePlatformPath

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASPlatformPathGetFSRefWithCFStringRefRecPtr

```
FSRefWithCFStringRefRec_Ptr
ASPlatformPathGetFSRefWithCFStringRefRecPtr
(ASPlatformPath path)
```

Description

Gets a platform path object in the form of an **FSRef** and **CFStringRef** for Mac OS, if the **ASPlatformPath** object was acquired with this type in the **platformPathType** parameter of **ASFileSysAcquirePlatformPath**.

NOTE: Do not release the returned value, or any member data of an **ASPlatformPath** directly; use **ASFileSysReleasePlatformPath** when finished with the object.

Parameters

path	The platform path.

Return Value

A pointer to a structure containing an **FSRef** and a **CFStringRef**.

Header File

ASCalls.h

Related Methods

[ASFileSysAcquirePlatformPath](#)
[ASPlatformPathGetFSRefPtr](#)
[ASFileSysReleasePlatformPath](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



ASPlatformPathGetFSSpecPtr

FSSpec_Ptr ASPlatformPathGetFSSpecPtr (**ASPlatformPath** path)

Description

Gets a platform path object in the form of an **FSSpec** for Mac OS, if the **ASPlatformPath** object was acquired with this type in the **platformPathType** parameter of **ASFileSysAcquirePlatformPath**.

NOTE: Do not release the returned value, or any member data of an **ASPlatformPath** directly; use **ASFileSysReleasePlatformPath** when finished with the object.

Parameters

path	The platform path.

Return Value

A pointer to an **FSSpec**.

Header File

ASCalls.h

Related Methods

[ASFileSysAcquirePlatformPath](#)
[ASPlatformPathGetFSRefPtr](#)
[ASFileSysReleasePlatformPath](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASPlatformPathGetPOSIXPathPtr

`POSIXPath_Ptr` ASPlatformPathGetPOSIXPathPtr
(`ASPlatformPath` path)

Description

Gets a platform path object in the form of a POSIX path C string, if the `ASPlatformPath` object was acquired with this type in the `platformPathType` parameter of `ASFileSysAcquirePlatformPath`.

NOTE: Do not release the returned value, or any member data of an `ASPlatformPath` directly; use `ASFileSysReleasePlatformPath` when finished with the object.

Parameters

<code>path</code>	The platform path.

Return Value

A pointer to a POSIX path (UTF-8 encoding) as a C string.

Header File

`ASCalls.h`

Related Methods

`ASFileSysAcquirePlatformPath`
`ASFileSysReleasePlatformPath`

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.



ASStm

ASFileStmRdOpen

```
ASStm ASFileStmRdOpen (ASFile afile, ASUns16 bufSize);
```

Description

Creates a read-only **ASStm** from a file. The stream is seek-able.

Parameters

afile	The open file to associate with the stream. The file must have been opened previously using ASFileSysOpenFile . Each open file has an unique ASFile . The ASFile value has meaning only to the common ASFile implementation and bears no relationship to platform-specific file objects.
bufSize	Length of data buffer, in bytes. If bufSize = 0, the default buffer size (currently 4kB) will be used. The default is generally reasonable. A larger buffer size should be used only when data in the file will be accessed in chunks larger than the default buffer. Although bufSize is passed as an ASUns16 , it is treated internally as an ASInt16 . As a result, buffer sizes above 32kB are not permitted.

Return Value

The newly created **ASStm**.

Header File

ASCalls.h

Related Methods

[ASFileSysOpenFile](#)
[ASFileStmWrOpen](#)
[ASMemStmRdOpen](#)
[ASProcStmRdOpenEx](#)
[ASStmClose](#)
[ASStmRead](#)
[CosNewStream](#)

Product

reader, base, pro, pdf

**Availability**

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

ASFileStmWrOpen

```
ASStm ASFileStmWrOpen (ASFile aFile, ASUns16 bufSize);
```

Description

Creates a writable **ASStm** from a file. The stream is seek-able.

Parameters

aFile	The open file to associate with the stream. The file must have been opened previously using ASFileSysOpenFile . Each open file has an unique ASFile . The ASFile value has meaning only to the common ASFile implementation and bears no relationship to platform-specific file objects.
bufSize	Length of a data buffer, in bytes. If bufSize = 0, the default buffer size (currently 4kB) is used. The default is generally reasonable. A larger buffer size should be used only when data in the file will be accessed in chunks larger than the default buffer. Although bufSize is passed as an ASUns16 , it is treated internally as an ASInt16 . As a result, buffer sizes above 32 KB are not permitted.

Return Value

The newly created **ASStm**.

Known Exceptions

[genErrNoMemory](#)

Header File

ASCalls.h

Related Methods

[ASProcStmWrOpen](#)
[ASFileStmRdOpen](#)
[ASMemStmRdOpen](#)
[ASProcStmRdOpenEx](#)
[ASStmWrite](#)
[ASStmRead](#)
[ASStmClose](#)
[ASFileSysOpenFile](#)
[CosNewStream](#)



Example

```
/* Create a writeable stream from an ASFile */
ASFile logFile;
ASStm writeLog;

writeLog = ASFileStmWrOpen (logFile, 0);

...

/* Be sure to close the stream (and the file) */
ASStmClose (writeLog);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

ASMemStmRdOpen

[ASStm](#) ASMemStmRdOpen (char* data, [ASUns32](#) len);

Description

Creates a read-only **ASStm** from a memory-resident buffer. The stream is seek-able.

Parameters

data	Buffer containing the data to read into the stream. This data buffer must not be disposed of until the ASStm is closed.
len	Length of data, in bytes.

Return Value

The newly created **ASStm**.

Header File

ASCalls.h

Related Methods

- [ASStmRead](#)
- [ASStmClose](#)
- [CosNewStream](#)
- [ASMemStmRdOpen](#)
- [ASProcStmRdOpenEx](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



ASProcStmRdOpen

```
ASStm ASProcStmRdOpen (ASStmProc readProc, void* clientData);
```

Description

Creates a read-only **ASStm** from an arbitrary data-producing procedure. The stream is not seek-able.

readProc is called when the client of the stream attempts to read data from it.

Parameters

readProc	User-supplied callback that supplies the stream's data.
clientData	Pointer to user-supplied data to pass to readProc each time it is called.

Return Value

The newly created **ASStm**.

Known Exceptions

[genErrNoMemory](#)

Header File

ASCalls.h

Related Methods

[ASStmRead](#)
[ASStmClose](#)
[CosNewStream](#)
[ASFileStmRdOpen](#)
[ASMemStmRdOpen](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



ASProcStmRdOpenEx

ASStm ASProcStmRdOpenEx (**ASProcStmRdExHandler** handler,
void* clientData);

Description

Extends **ASProcStmRdOpen** for Acrobat 6.0. Creates a read-only **ASStm** from an arbitrary data-producing procedure. The stream is not seek-able.

The supplied handlers are called when the client of the stream attempts to read data from it, and when the client closes it.

Parameters

handler	Structure containing user-supplied callbacks that supply the stream's data and destroy the stream.
clientData	Pointer to user-supplied data to pass to the procedures each time they are called.

Return Value

The newly created **ASStm**.

Known Exceptions

genErrNoMemory

Header File

ASCalls.h

Related Methods

ASProcStmRdOpen
ASStmClose
ASStmRead

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASProcStmWrOpen

```
ASStm ASProcStmWrOpen (ASStmProc writeProc,
ASProcStmDestroyProc destroyProc, void* clientData);
```

Description

Creates an **ASStm** from an arbitrary data-producing procedure. The stream is not seek-able.

Parameters

writeProc	User-supplied callback that provides the data for the stream.
destroyProc	User-supplied callback that destroys the specified ASStm . (Generally, this means de-allocating the memory associated with the ASStm .)
clientData	Pointer to user-supplied data to pass to writeProc each time it is called.

Return Value

The newly created **ASStm**.

Known Exceptions

[genErrNoMemory](#)

Header File

ASCalls.h

Related Methods

[ASFileStmRdOpen](#)
[ASFileStmWrOpen](#)
[ASMemStmRdOpen](#)
[ASProcStmRdOpenEx](#)
[ASStmWrite](#)
[ASStmRead](#)
[ASStmClose](#)
[CosNewStream](#)

Example

```
ACCB1 ASInt32 ACCB2 writeProc(char* data, ASInt32 nBytes,
void* clientData);
{
...
}

ACCBPROTO1 void ACCBPROTO2 destroyProc (void* clientData);
{
...
}

ASStm writeLog;

writeProcPtr = ASCallbackCreateProto(ASReportProc, writeProc);
destroyProcPtr = ASCallbackCreateProto(ASProcStmDestroyProc,
destroyProc);
writelog = ASProcStmWrOpen (writeProc, destroyProc, void* clientData);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.



ASStmClose

```
void ASStmClose (ASStm stm) ;
```

Description

Closes the specified stream.

Parameters

stm	The stream to close.

Return Value

None

Header File

ASCalls.h

Related Methods

[ASFileStmRdOpen](#)
[ASFileStmWrOpen](#)
[ASMemStmRdOpen](#)
[ASProcStmRdOpenEx](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.



ASStmFlush

```
ASTCount ASStmFlush (ASStm stm);
```

Description

Flushes any buffered data to the specified stream.

Parameters

stm	The stream to flush.

Return Value

0 if successful, otherwise non-zero.

Header File

ASCalls.h

Related Methods

[ASStmClose](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASStmRead

```
ASTCount ASStmRead (char* buffer, ASTArraySize itemSize,
ASTCount nItems, ASStm stm);
```

Description

Reads data from **stm** into memory.

Parameters

buffer	<i>(Filled by the method)</i> Buffer into which data is written.
itemSize	Number of bytes in an item in the stream. See the description of nItems for further information.
nItems	Number of items to read. The amount of data read into the memory buffer will be itemSize × nItems , unless an EOF is encountered first. The relative values of itemSize and nItems really do not matter; the only thing that matters is their product. It is often convenient to set itemSize to 1, so that nItems is the number of bytes to read.
stm	The stream from which data is read.

Return Value

The number of items (not bytes) read.

Header File

ASCalls.h

Related Methods

[ASStmWrite](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



ASStmWrite

```
ASTCount ASStmWrite (const char* buffer,
ASTArraySize itemSize, ASTCount nItems, ASStm stm);
```

Description

Writes data from a memory buffer into an **ASStm**.

You cannot use this method to change a PDF page contents stream—only a print stream.

Historically, this method was provided to allow plug-ins to write data into the print stream when printing to a PostScript printer (see the [PDDocWillPrintPage](#) notification). However, **ASStm** is a general purpose I/O mechanism in Acrobat, although only limited open and read/write methods are provided in the plug-in API. For instance, not all **ASStm** objects are seek-able.

Parameters

buffer	Buffer from which data is read.
itemSize	Number of bytes in an item in the stream. See the description of nItems for additional information.
nItems	Number of items to write. The amount of data written into the stream will be itemSize × nItems . The relative values of itemSize and nItems really do not matter; the only thing that matters is their product. It is often convenient to set itemSize to 1, so that nItems is the number of bytes to read.
stm	The stream into which data is written.

Return Value

The number of items (not bytes) written.

Header File

ASCalls.h

Related Methods

[ASStmRead](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



ASText

ASTextCaseSensitiveCmp

```
ASInt32 ASTextCaseSensitiveCmp (ASConstText str1,  
ASConstText str2);
```

Description

Compares two **ASConstText** objects, ignoring language and country information. The comparison is case-sensitive.

Parameters

str1	First text object.
str2	Second text object.

Return Value

Returns a negative number if **str1** < **str2**, a positive number if **str1** > **str2**, and 0 if they are equal.

Known Exceptions

Various

Header File

ASExtraCalls.h

Related Methods

[ASTextCmp](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASTextCat

```
void ASTextCat (ASText to, ASText from);
```

Description

Concatenates the **from** text to the end of the **to** text, altering **to** but not **from**. Does not change the language or country of **to**, unless it has no language or country in which case it acquires the language and country of **from**.

Parameters

to	The encoded text to which from is appended.
from	The encoded text to be appended to to .

Return Value

None

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASTextCatMany

```
void ASTextCatMany (ASText to, ...);
```

Description

Concatenates a series of **ASText** objects to the end of the **to** object. Be sure to provide a **NULL** as the last argument to the call.

Parameters

to	The ASText object to which the subsequent ASText arguments are concatenated

Return Value

None

Known Exceptions

Various

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASTextCmp

```
ASInt32 ASTextCmp (ASText str1, AText str2);
```

Description

Compares two **ASText** objects. This routine can be used to sort text objects using the default collating rules of the underlying OS before presenting them to the user. The comparison is always case-insensitive.

Parameters

str1	First text object.
str2	Second text object.

Return Value

Returns a negative number if **str1** < **str2**, a positive number if **str1** > **str2**, and 0 if they are equal.

Known Exceptions

Various

Header File

ASExtraCalls.h

Related Methods

[ASTextCaseSensitiveCmp](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASTextCopy

```
void ASTextCopy (ASText to, ASText from);
```

Description

Copies the text in **from** to **to**, along with the country and language.

Parameters

to	Destination text object.
from	Source text object.

Return Value

None

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in PIRequir.h) is set to **0x00050000** or higher.



ASTextDestroy

```
void ATextDestroy (ASText str);
```

Description

Frees all memory associated with the text object.

Parameters

str	A text object.

Return Value

None

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



ASTextDup

```
ASText ASTextDup (ASText str);
```

Description

Creates a new **ASText** object that contains the same text/country/language as the one passed in.

Parameters

str	A text object.

Return Value

An **ASText** object.

Known Exceptions

genErrBadParm if **str == NULL**.

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASTextEval

```
void ASTextEval (ASText theText, ASCab params);
```

Description

Replaces percent-quoted expressions in the text object with the result of their evaluation, using key/value pairs in the **ASCab**. For example, for a text value containing the string "%**keyone**%%**keytwo**%", the value is replaced with the concatenation of the values of the keys **keyone** and **keytwo** in the **ASCab** passed in.

Parameters

theText	A text object containing percent-quoted expressions to replace.
params	The ASCab containing the key/value pairs to use for text replacement.

Return Value

None.

Known Exceptions

[genErrBadParm](#) if **theText** ==NULL.

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AShaderFilter

```
void AShaderFilter (AShader text, AShaderFilterType filter);
```

Description

Runs the specified filter on a text object, modifying the text as specified.

Parameters

text	A text object. Modified by the method.
filter	The filter to run on the text object.

Return Value

None.

Known Exceptions

[genErrBadParm](#) if `text == NULL` or if an invalid filter is specified.

Header File

AShaderCalls.h

Product

reader, base, pro, pdf

Availability

Available if `PI_ASHADER_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

ASTextFromEncoded

```
ASText ASTextFromEncoded (const char* str,  
ASHostEncoding encoding);
```

Description

Creates a new text object from a null-terminated multi-byte string in the specified host encoding.

Parameters

str	The input string.
encoding	The host encoding

Return Value

An **ASText** object.

Header File

ASExtraCalls.h

Related Methods

[ASTextFromSizedEncoded](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



ASTextFromInt32

```
ASText ASTextFromInt32 (ASInt32 num) ;
```

Description

Create a new string from an [ASInt32](#) by converting the number to its decimal representation without punctuation or leading zeros.

Parameters

num	A number of type ASInt32 ..

Return Value

An **ASText** object.

Header File

`ASExtraCalls.h`

Related Methods

[ASTextFromUns32](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



ASharedFromPDFText

ASharedFromPDFText (const char* str);

Description

Creates a new string from some PDF Text taken out of a PDF file. This is either a UTF-16 string with the **0xFEFF** pre-pended to the front or a **PDFDocEncoding** string. In either case the string is expected to have the appropriate **NULL**-termination. If the **PDFText** is in UTF-16 it may have embedded language and country information; this will cause the **ASharedFromPDFText** object to have its language and country codes set to the values found in the string.

Parameters

str	A string.

Return Value

An **ASharedFromPDFText** object.

Header File

ASharedFromPDFText.h

Related Methods

ASharedFromSizedPDFText

Product

reader, base, pro, pdf

Availability

Available if **PI_ASHARED_FROMPDFTEXT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AScriptFromScriptText

```
AScript AScriptFromScriptText (const char* str,
AScript script);
```

Description

Creates a new string from a null-terminated multi-byte string of the specified script. This is a wrapper around [AScriptFromEncoded](#); the script is converted to a host encoding using [AScriptToHostEncoding](#).

Parameters

str	A string.
script	The specified script

Return Value

An **AScript** object.

Header File

AScriptExtraCalls.h

Related Methods

[AScriptFromSizedScriptText](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASCRIPT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASTextFromSizedEncoded

```
ASText ASTextFromSizedEncoded (const char* str,
ASArraySize len, ASHostEncoding encoding);
```

Description

Creates a new text object from a multi-byte string in the specified host encoding and of the specified length.

Parameters

str	A string.
len	Length in bytes.
encoding	The specified host encoding

Return Value

An **ASText** object.

Known Exceptions

genErrBadParm if len < 0

Header File

ASExtraCalls.h

Related Methods

[ASTextFromEncoded](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASTextFromSizedPDText

```
ASText ASTextFromSizedPDText (const char* str,
ASArraySize length);
```

Description

Creates a new string from some PDF Text taken out of a PDF file. This is either a UTF-16 string with the **0xFEFF** prepended to the front or a **PDFDocEncoding** string. If the **PDText** is in UTF-16 it may have embedded language and country information; this will cause the **ASText** object to have its language and country codes set to the values found in the string. The length parameter specifies the size, in bytes, of the string. The string must not contain embedded null-characters.

Parameters

str	A string.
len	Length in bytes.

Return Value

An **ASText** object.

Header File

ASExtraCalls.h

Related Methods

[ASTextFromPDText](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AScriptFromSizedScriptText

```
AScript AScriptFromSizedScriptText (const char* str,
AScriptSize len, AScript script);
```

Description

Creates a new text object from the specified multi-byte string of the specified script. This is a wrapper around [AScriptFromEncoded](#); the script is converted to a host encoding using [AScriptToHostEncoding](#).

Parameters

str	A string.
len	Length in bytes.
script	The specified script

Return Value

An **AScript** object.

Header File

AScriptCalls.h

Related Methods

[AScriptFromScriptText](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASCRIPT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASTextFromSizedUnicode

```
ASText ASTextFromSizedUnicode (const ASUTF16Val* ucs,
ASUnicodeFormat format, ASArraySize len);
```

Description

Creates a new text object from the specified Unicode string. This string is not expected to have 0xFE 0xFF prepended, or country/language identifiers.

The string cannot contain an embedded null character.

Parameters

ucs	The Unicode string
format	The Unicode format of ucs.
len	The length of ucs in bytes.

Return Value

An ASText object.

Known Exceptions

genErrBadParm if len < 0

Header File

ASExtraCalls.h

Related Methods

ASTextFromUnicode

Product

reader, base, pro, pdf

Availability

Available if PI_ASEXTRA_VERSION (in PIRequir.h) is set to 0x00050000 or higher.



ASTextFromUnicode

```
ASText ASTextFromUnicode (const ASUTF16Val* ucs,
ASUnicodeFormat format);
```

Description

Creates a new string from a null-terminated Unicode string. This string is not expected to have **0xFE 0xFF** prepended, or country/language identifiers.

Parameters

ucs	A Unicode string.
format	The Unicode format used by ucs .

Return Value

An **ASText** object.

Header File

ASExtraCalls.h

Related Methods

[ASTextFromSizedUnicode](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASTextFromUns32

```
ASText ASTextFromUns32 (ASUns32 num) ;
```

Description

Create a new string from an [ASUns32](#) by converting it to a decimal representation without punctuation or leading zeros.

Parameters

<code>num</code>	A value of type ASUns32 ..

Return Value

An **ASText** object.

Header File

`ASExtraCalls.h`

Related Methods

[ASTextFromInt32](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ASEXTRA_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

ASTextGetBestEncoding

```
ASHostEncoding ASTextGetBestEncoding (ASText str,
ASHostEncoding preferredEncoding) ;
```

Description

Returns the best host encoding for representing the text. The best host encoding is the one that is least likely to lose characters during the conversion from Unicode to host. If the string can be represented accurately in multiple encodings (for example, it is low-ASCII text that can be correctly represented in any host encoding), **ASTextGetBestEncoding** returns the preferred encoding based on the **preferredEncoding** parameter.

Parameters

str	An ASText string.
preferredEncoding	The preferred encoding. There is no default.

Return Value

The text encoding.

Known Exceptions

Various

Header File

ASExtraCalls.h

Related Methods

[ASTextGetBestScript](#)

Examples

```
// If you prefer to use the app's language encoding...
ASHostEncoding bestEncoding = ASTextGetBestEncoding(text,
AVAppGetLanguageEncoding());

// If you prefer to use the OS encoding...
ASHostEncoding bestEncoding = ASTextGetBestEncoding(text,
(ASHostEncoding) PDGetHostEncoding());

// If you want to favor Roman encodings...
ASHostEncoding hostRoman = ASScriptToHostEncoding(kASRomanScript);
ASHostEncoding bestEncoding = ASTextGetBestEncoding(text, hostRoman);
```

Product

reader, base, pro, pdf



Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



AScriptGetBestScript

```
AScript AScriptGetBestScript (AScript str,
AScript preferredScript);
```

Description

Returns the best host script for representing the text. Functionality is similar to [AScriptGetBestEncoding](#) with resulting host encoding converted to a script code using [AScriptFromHostEncoding](#).

Parameters

str	An AScript string.
preferredScript	The preferred host script. There is no default.

Return Value

The best host script.

Header File

AScriptExtraCalls.h

Related Methods

[AScriptGetBestEncoding](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASCRIPT_EXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASTextGetCountry

```
ASUns16 ASTextGetCountry (ASText text);
```

Description

Retrieves the country associated with an **ASText** object.

Parameters

text	An ASText object.

Return Value

The country code.

Header File

ASExtraCalls.h

Related Methods

[ASTextSetCountry](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASTextGetEncoded

```
const char* ASTextGetEncoded (ASText str,
    AHostEncoding encoding);
```

Description

Returns a null-terminated string in the given encoding. The memory pointed to by this string is owned by the **ASText** object and may not be valid after additional operations are performed on the object.

Parameters

str	An ASText object.
encoding	The specified host encoding.

Return Value

A pointer to a null-terminated string corresponding to the text in **str**.

Known Exceptions

Various

Header File

ASExtraCalls.h

Related Methods

[ASTextGetEncodedCopy](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASTextGetEncodedCopy

```
char* ASTextGetEncodedCopy (ASText str,
    ASHostEncoding encoding);
```

Description

Returns a copy of a string in a specified encoding.

Parameters

str	An ASText object.
encoding	The specified encoding

Return Value

A copy of the text in **str**. The client owns the resulting information and is responsible for freeing it using **ASfree**.

Known Exceptions

genErrNoMemory if memory could not be allocated for the copy.

Header File

ASExtraCalls.h

Related Methods

[ASTextGetEncoded](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



AS`TextGetLanguage`

[ASUns16](#) AS`TextGetLanguage` ([AS`Text`](#) text);

Description

Retrieves the language code associated with an **AS`Text`** object. For details of language values, see [Language Codes](#).

Parameters

text	An AS<code>Text</code> object.

Return Value

The language code.

Header File

AS`ExtraCalls.h`

Related Methods

[AS`TextSetLanguage`](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

ASTextGetPDTextCopy

```
char* ASTextGetPDTextCopy (ASConstText str,
ASArraySize* len);
```

Description

Returns the text in a form suitable for storage in a PDF file. If the text can be represented using **PDFDocEncoding**, it is; otherwise it is represented in big-endian UTF-16 format with **0xFE 0xFF** prepended to the front and any country/language codes embedded in an escape sequence right after **0xFE 0xFF**.

You can determine if the string is Unicode by inspecting the first two bytes. The Unicode case is used if the string has a language and country code set. The resulting string is null-terminated as appropriate. That is, one null byte for **PDFDocEncoding**, two for UTF-16.

Parameters

str	A string.
len	The length in bytes of the resulting string, not counting the null bytes at the end.

Return Value

A string copy. The client owns the resulting information and is responsible for freeing it with **ASfree**.

Known Exceptions

Various

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AScriptGetScriptText

```
const char* AScriptGetScriptText (AScript str, AScript script);
```

Description

Converts the Unicode string in the **AScript** object to the appropriate script and returns a pointer to the converted text. The memory pointed to is owned by the **AScript** object and must not be altered or destroyed by the client. The memory may also become invalid after subsequent operations are applied to the **AScript** object.

Parameters

str	A string.
script	The writing script.

Return Value

A string.

Known Exceptions

Various

Header File

AScriptExtra.h

Related Methods

[AScriptGetScriptTextCopy](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASCRIPT_EXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AScriptGetScriptTextCopy

```
char* AScriptGetScriptTextCopy (ASConstText str,
ASScript script);
```

Description

Converts the Unicode string in the **AScript** object to the appropriate script and returns a pointer to the converted text. The memory pointed to is owned by the client who is responsible for freeing it using **ASFree**.

Parameters

str	A string.
script	A writing script.

Return Value

A string copy. The client owns the resulting information.

Known Exceptions

genErrNoMemory if memory could not be allocated for the copy.

Header File

ASExtraCalls.h

Related Methods

[AScriptGetEncodedCopy](#)
[AScriptGetScriptText](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASTextGetUnicode

```
const ASUTF16Val* ASTextGetUnicode (ASText str);
```

Description

Returns a pointer to a string in **kUTF16HostEndian** format (see [ASUnicodeFormat](#)). The memory pointed to by this string is owned by the **ASText** object and may not be valid after additional operations are performed on the object.

The Unicode text returned will not have **0xFE 0xFF** prepended or any language or country codes.

Parameters

str	A string.

Return Value

See above.

Header File

ASExtraCalls.h

Related Methods

[ASTextGetUnicodeCopy](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASTextGetUnicodeCopy

```
ASUTF16Val* ASTextGetUnicodeCopy (ASText str,
ASUnicodeFormat format);
```

Description

Returns a pointer to a null-terminated string in the specified Unicode format. The memory pointed to by this string is owned by the client who can modify it at will and is responsible for destroying it using [ASfree](#).

The Unicode text returned will not have **0xFE 0xFF** prepended or any language or country codes.

Parameters

str	A string.
format	Unicode format.

Return Value

A string copy. The client owns the resulting information.

Known Exceptions

[genErrNoMemory](#) if memory could not be allocated for the copy.

Header File

ASExtraCalls.h

Related Methods

[ASTextGetUnicode](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASTextIsEmpty

```
ASBool ASTextIsEmpty (ASText str);
```

Description

Used to determine whether the **ASText** object contains no text—for example, if retrieving Unicode text would yield a 0-length string.

Parameters

str	A string.

Return Value

Returns **true** if the **ASText** object contains no text.

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AShTextMakeEmpty

```
void AShTextMakeEmpty (AShText str);
```

Description

Removes the contents of an **AShText** (turns it into an empty string).

Return Value

None

Header File

AShExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASH_EXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



AShTextNew

AShText AShTextNew (void) ;

Description

Creates a new text object containing no text.

Return Value

An **AShText** object.

Known Exceptions

genErrNoMemory

Header File

AShExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASH_EXTRA_VERSION** (in PIRequir.h) is set to **0x00050000** or higher.

ASTextNormalizeEndOfLine

```
void ASTextNormalizeEndOfLine (ASText text);
```

Description

Replaces all end-of-line characters within the **ASText** object with the correct end-of-line character for the current platform—for example, for Windows, `\r` and `\n` are replaced with `\r\n`.

Parameters

text	An object of type ASText .

Return Value

None

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



AShTextReplace

```
void AShTextReplace (AShText src, AShText toReplace,
                    AShText replacement);
```

Description

Replaces all occurrences of **toReplace** in **src** with the text specified in **replacement**. **AShTextReplace** uses an **AShText** string to indicate the **toReplace** string; **AShTextReplaceASCII** uses a low-ASCII Roman string to indicate the text to replace.

Parameters

src	Source text.
toReplace	Text in source text to replace.
replacement	Text used in replacement.

Return Value

None

Known Exceptions

Various

Header File

AShExtraCalls.h

Related Methods

[AShTextReplaceASCII](#)
[AShTextReplaceBadChars](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASH_EXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AShouldReplaceASCII

```
void AShouldReplaceASCII (AShouldText src, const char* toReplace,
    AShouldText replacement);
```

Description

Replaces all occurrences of **toReplace** in **src** with the text specified in **replacement**. **AShouldReplace** uses an **AShouldText** string to indicate the **toReplace** string; **AShouldReplaceASCII** uses a low-ASCII Roman string to indicate the text to replace. This call is intended for formatting strings for the user interface—for example, replacing a known sequence such as “%1” with other text. Be sure to use only low-ASCII characters, which are safe on all platforms. Avoid using backslash and currency symbols.

Parameters

src	The AShouldText object containing the text.
toReplace	The text to replace.
replacement	The replacement text.

Return Value

None

Known Exceptions

Various

Header File

AShouldExtraCalls.h

Related Methods

- [AShouldReplace](#)
- [AShouldReplaceBadChars](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASHOULD_EXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASTextReplaceBadChars

```
void ASTextReplaceBadChars (ASText str, char *pszBadCharList,
char replaceChar);
```

Description

Replaces all occurrences of characters contained in the list **pszBadCharList** in the text with the specified replacement character.

Parameters

str	The text in which to replace characters.
pszBadCharList	A list of characters to replace, in sorted order with no duplicates.
replaceChar	The character with which to replace any character appearing in the list.

Return Value

None

Known Exceptions

Various

Header File

ASExtraCalls.h

Related Methods

- ASTextReplace
- ASTextReplaceASCII

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASetsCountry

```
void ASetsCountry (ASets text, ASets16 country);
```

Description

ASets objects can have country and language codes associated with them. These can be explicitly set or parsed from the Unicode form of **PDets** strings. This method allows you to set the language codes associated with a piece of text.

Parameters

text	An ASets object.
country	Country code.

Return Value

None

Header File

ASetsCalls.h

Related Methods

[ASetsCountry](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASetTextEncoded

```
void ASetTextEncoded (ASetText str, const char* text,
    AHostEncoding encoding);
```

Description

Replaces the contents of an existing **ASetText** object with a null-terminated multi-byte string in the specified host encoding.

Parameters

str	ASetText object to hold the string.
text	Pointer to text string.
encoding	Type of encoding.

Return Value

None

Known Exceptions

genErrBadParm if **text** = **NULL**

Header File

ASetExtraCalls.h

Related Methods

[ASetTextSizedEncoded](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASETTEXT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASetsLanguage

```
void ASetsLanguage (ASets text, ASets16 language);
```

Description

Sets the language codes associated with a piece of text.

Parameters

text	An ASets object.
language	Language code.

Return Value

None

Header File

ASetsCalls.h

Related Methods

[ASetsGetLanguage](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASETRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASetPDText

```
void ASetPDText (ASet str, const char* text);
```

Description

Alters an existing string from some PDF text taken out of a PDF file. This is either a big-endian UTF-16 string with the **0xFEFF** prepended to the front or a **PDFDocEncoding** string. In either case the string is expected to have the appropriate null termination. If the **PDText** is in UTF-16 it may have embedded language and country information; this will cause the **ASet** object to have its language and country codes set to the values found in the string.

Parameters

str	A string.
text	Text string.

Return Value

None

Header File

ASetCalls.h

Related Methods

[ASetSizedPDText](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASET_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AScriptSetScriptText

```
void AScriptSetScriptText (AScript str, const char* text,
AScript script);
```

Description

Alters an existing string from a null-terminated multi-byte string of the specified script. This is a wrapper around [AScriptFromEncoded](#); the script is converted to a host encoding using [AScriptToHostEncoding](#).

Parameters

str	A string.
text	Pointer to text string.
script	The writing script.

Return Value

None

Header File

AScriptExtraCalls.h

Related Methods

[AScriptSetSizedScriptText](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ASCRIPT_EXTRA_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

ASetSizedEncoded

```
void ASetSizedEncoded (ASet str, const char* text,
    AArraySize len, AHostEncoding encoding);
```

Description

Alters an existing string from a multi-byte string in the specified host encoding and of the specified length. This text does not need to be null-terminated, and no null (zero) bytes should appear in the characters passed in.

Parameters

str	A string.
text	Pointer to text string.
len	Length of text string.
encoding	Host encoding type.

Return Value

None

Known Exceptions

[genErrBadParm](#) if `text = NULL`

Header File

ASExtraCalls.h

Related Methods

[ASetSizedEncoded](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ASEXTRA_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

ASetSizedPDText

```
void ASetSizedPDText (ASet str, const char* text,
    ASArraySize length);
```

Description

Replaces the contents of an existing **ASet** object with PDF text taken out of a PDF file. This is either a big-endian UTF-16 string with the **0xFEFF** prepended to the front or a **PDFDocEncoding** string. In either case the length parameter indicates the number of bytes in the string. The string should not be null-terminated and must not contain any null characters. If the **PDText** is in UTF-16 it may have embedded language and country information; this will cause the **ASet** object to have its language and country codes set to the values found in the string.

Parameters

str	A string.
text	Pointer to a text string.
length	Length of the text string.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASetSetPDText](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



ASetSizedScriptText

```
void ASetSizedScriptText (ASet str, const char* text,
    AArraySize len, AScript script);
```

Description

Replaces the contents of an existing **ASet** object with the specified multi-byte string of the specified script. This is a wrapper around **ASetFromSizedEncoded**; the script is converted to a host encoding using **AScriptToHostEncoding**.

Parameters

str	A string.
text	Pointer to a text string.
len	Length of the text string.
script	The writing script.

Return Value

None

Known Exceptions

genErrBadParm if **text==NULL**

Header File

ASetExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASET_EXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASetSizedUnicode

```
void ASetSizedUnicode (ASet str,
const ASUTF16Val* ucsValue, ASUnicodeFormat format,
ASArraySize len);
```

Description

Replaces the contents of an existing **ASet** object with the specified Unicode string. This string is not expected to have **0xFE 0xFF** prepended or embedded country/language identifiers.

The string cannot contain a null character.

Parameters

str	<i>(Filled by the method)</i> A string.
ucsValue	A Unicode string.
format	The Unicode format.
len	Length of the string in bytes.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASetUnicode](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASTextSetUnicode

```
void ASTextSetUnicode (ASText str, const ASUTF16Val* ucsValue,
ASUnicodeFormat format);
```

Description

Alters an existing string from a null-terminated Unicode string. This string is not expected to have 0xFE 0xFF prepended or embedded country/language identifiers.

Parameters

str	(Filled by the method) A string.
ucsValue	A Unicode string.
format	The Unicode format.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASTextSetSizedUnicode](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ASEXTRA_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.



ASTimeSpan

ASCalendarTimeSpanAddWithBase

```
void ASCalendarTimeSpanAddWithBase
(const ASCalendarTimeSpan timeSpan1,
const ASCalendarTimeSpan timeSpan, const ASDate baseDate,
ASCalendarTimeSpan result);
```

Description

Adds two calendar time spans, storing the result in another calendar time span object. Because the values in a calendar time span are not absolute (for example, a leap year has a different number of days), they are resolved with respect to the base date before the addition is done. The result is broken down into years, months, and so on, in the highest denomination possible. For example, a difference of 13 months is reported as 1 year and 1 month.

Parameters

timeSpan1	The first calendar time span to add.
timeSpan2	The calendar time span to add.
baseDate	The base date, or NULL to use Jan 1 1970 00:00:00 , the epoch time.
result	The calendar time span structure in which to store the result.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASCalendarTimeSpanCompare](#)
[ASTimeSpanAdd](#)
[ASDateAddCalendarTimeSpan](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASCalendarTimeSpanCompare

```
ASInt32 ASCalendarTimeSpanCompare
(const ASCalendarTimeSpan timeSpan1,
const ASCalendarTimeSpan timeSpan2, const ASDate baseDate);
```

Description

Compares two calendar time spans with respect to a base date. Because the values in a calendar time span are not absolute (for example, a leap year has a different number of days), they are resolved with respect to the base date before the comparison is made.

Parameters

timeSpan1	The first calendar time span.
timeSpan2	The second calendar time span.
baseDate	The base date, or NULL to use Jan 1 1970 00:00:00 , the epoch time.

Return Value

1 if **timeSpan1**>**timeSpan2**, 0 if they are equal, -1 if **timeSpan1**<**timeSpan2**.

Header File

ASExtraCalls.h

Related Methods

[ASDateCompare](#)
[ASTimeSpanCompare](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASCalendarTimeSpanDiff

```
void ASCalendarTimeSpanDiff
(const ASCalendarTimeSpan timeSpan1,
const ASCalendarTimeSpan timeSpan2, const ASDate baseDate,
ASCalendarTimeSpan result);
```

Description

Calculates the difference between calendar time span objects and stores the result in the provided [ASCalendarTimeSpan](#) object. If `timeSpan2` is less than `timeSpan1`, the result is negative. Because the values in a calendar time span are not absolute (for example, a leap year has a different number of days), they are resolved with respect to the base date before the addition is done. The result is broken down into years, months, and so on, in the highest denomination possible. For example, a difference of 13 months is reported as 1 year and 1 month.

Parameters

<code>timeSpan1</code>	The first calendar time span.
<code>timeSpan2</code>	The second calendar time span.
<code>baseDate</code>	The base date, or <code>NULL</code> to use <code>Jan 1 1970 00:00:00</code> , the epoch time.
<code>result</code>	The calendar time span object in which to store the difference.

Return Value

None

Header File

`ASExtraCalls.h`

Related Methods

[ASDateCalendarDiff](#)
[ASTimeSpanDiff](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.



ASTimeSpanAdd

```
void ASTimeSpanAdd (const ASTimeSpan timeSpan1,
const ASTimeSpan timeSpan, ASTimeSpan result);
```

Description

Adds two time spans, storing the result (an exact number of seconds) in another time span object.

Parameters

timeSpan1	The first time span to add.
timeSpan2	The second time span to add.
result	The time span object in which to store the result.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASCalendarTimeSpanAddWithBase](#)
[ASDateAddTimeSpan](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASTimeSpanCompare

```
ASInt32 ASTimeSpanCompare (const ASTimeSpan timeSpan1,
const ASTimeSpan timeSpan2, const ASDate baseDate);
```

Description

Compares two time spans to see if they are equal, or if one represents fewer seconds than the other.

Parameters

timeSpan1	The first time span.
timeSpan2	The second time span.
baseDate	The base date, or NULL to use Jan 1 1970 00:00:00 , the epoch time.

Return Value

1 if **timeSpan1**>**timeSpan2**, 0 if they are equal, -1 if **timeSpan1**<**timeSpan2**.

Header File

ASExtraCalls.h

Related Methods

[ASDateCompare](#)
[ASCalendarTimeSpanCompare](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



ASpanCopy

```
void ASpanCopy (const ASpan original,
                ASpan copy);
```

Description

Copies data from one time span object to another.

Parameters

original	The time span to be copied.
copy	The time span into which the data is copied.

Return Value

None

Header File

ASpan.h

Related Methods

[ASpanDup](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



ASpanDestroy

```
void ASpanDestroy (ASpan_t span);
```

Description

Releases and destroys a span object.

Parameters

span	The span.

Return Value

None

Header File

ASpan.h

Related Methods

[ASpanDup](#)

[ASpanNew](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASTimeSpanDiff

```
void ASTimeSpanDiff (const ASTimeSpan timeSpan1,
const ASTimeSpan timeSpan2, ASTimeSpan result);
```

Description

Calculates the exact difference in seconds between time span objects and stores the result in the provided [ASTimeSpan](#) object. If **timeSpan2** is less than **timeSpan1**, the result is negative.

Parameters

timeSpan1	The first time span.
timeSpan2	The second time span.
result	The time span object in which to store the difference.

Return Value

None

Header File

ASExtraCalls.h

Related Methods

[ASDateExactDiff](#)
[ASCalendarTimeSpanDiff](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASpanDup

```
ASpan ASpanDup (const ASpan timeSpan);
```

Description

Creates a new time span object containing the same data as an existing time span object.
Raises an exception if there is not enough memory.

Parameters

timeSpan	The time span to duplicate.

Return Value

The new time span object.

Header File

ASpan.h

Related Methods

[ASpanCopy](#)
[ASpanDestroy](#)
[ASpanNew](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASTimeSpanGetASInt32

```
ASInt32 ASTimeSpanGetASInt32 (ASTimeSpan timeSpan,
ASBool* outOverflow);
```

Description

Gets the number of seconds from a time span object.

Parameters

timeSpan	The time span object.
outOverflow	(Filled by the method) true if the number of seconds was too large to be represented by an ASInt32 value, false otherwise.

Return Value

The number of seconds.

Header File

ASExtraCalls.h

Related Methods

[ASTimeSpanSet](#)
[ASTimeSpanSetFromASInt32](#)
[ASTimeSpanSetFromString](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



ASpanNegate

```
void ASpanNegate (ASpan timeSpan);
```

Description

Negates the time span value of a time span object.

Parameters

timeSpan	The time span.

Return Value

None

Header File

ASpan.h

Related Methods

[ASpanSet](#)
[ASpanSetFromASInt32](#)
[ASpanSetFromString](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASpanNew

[ASpan](#) ASpanNew (void);

Description

Creates a time span object. Raises an exception if there is not enough memory for the operation.

Parameters

None

Return Value

The newly created time span object.

Header File

ASpan.h

Related Methods

[ASpanCopy](#)

[ASpanDup](#)

[ASpanDestroy](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.



ASpanSet

```
void ASpanSet (ASpan timeSpan, ASInt32 highBits,  
ASInt32 lowBits);
```

Description

The internal representation of a time span uses 64-bit signed integers (to avoid the 2038 problem caused by 32-bit representation). This method initializes a time span object to represent a time span of *x* number of seconds, where *x* is the 64-bit signed integer obtained from concatenating **highBits** and **lowBits**.

Parameters

timeSpan	The time span object.
highBits	The most significant word in the desired 64-bit signed integer value.
lowBits	The least significant word in the desired 64-bit signed integer value.

Return Value

None

Header File

ASpanSet.h

Related Methods

[ASpanSetFromASInt32](#)
[ASpanSetFromString](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASpanSetFromASInt32

```
void ASpanSetFromASInt32 (ASpan timeSpan,
ASInt32 numSeconds);
```

Description

Initializes a time span object to represent a time span of a specific number of seconds.

Parameters

timeSpan	The time span object.
numSeconds	The number of seconds.

Return Value

None

Header File

ASpanSet.h

Related Methods

[ASpanGetASInt32](#)
[ASpanSet](#)
[ASpanSetFromString](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASpanSetFromString

```
void ASpanSetFromString (ASpan span,
    const char *numSecondsString);
```

Description

Convert a string to a number of seconds, and initializes a time span object to represent a time span of that number of seconds. This is useful for time spans that are too long to represent with an [ASInt32](#) value.

Parameters

span	The time span object.
numSecondsString	The string containing the number of seconds. The string must consist of an optional minus sign (for negative numbers) followed by decimal digits. No white spaces are allowed anywhere in the string.

Return Value

None

Header File

ASpanSet.h

Related Methods

- [ASpanSet](#)
- [ASpanSetFromASInt32](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.



ASUUID

ASUUIDFromCString

[ASBool](#) ASUUIDFromCString ([ASUUID](#) *dst, const char *str)

Description

Parses a C string, such as one generated by [ASUUIDToCString](#), into a unique identifier (UUID).

Parameters

dst	(Filled by the method) The unique identifier created from the string.
str	A NULL -terminated string from which to generate the unique identifier, in the following form: f81d4fae-7dec-11d0-a765-00a0c91e6bf6

Return Value

true if the UUID is successfully created, **false** otherwise.

Header File

ASCalls.h

Related Methods

[ASUUIDGenFromHash](#)
[ASUUIDGenFromName](#)
[ASUUIDGenUnique](#)
[ASUUIDToCString](#)
[AVAppGetUUID](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

ASUUIDGenFromHash

```
ASBool ASUUIDGenFromHash (ASUUID *dst, ASUns8 hash[16])
```

Description

Generates a unique identifier (UUID) from a hash value.

Parameters

dst	<i>(Filled by the method)</i> The unique identifier created from the hash.
hash	A hash value, such as MD5.

Return Value

true if the UUID is successfully created, **false** otherwise.

Header File

ASCalls.h

Related Methods

[ASUUIDFromCString](#)
[ASUUIDGenFromName](#)
[ASUUIDGenUnique](#)
[ASUUIDToCString](#)
[AVAppGetUUID](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASUUIDGenFromName

```
ASBool ASUUIDGenFromName (ASUUID *dst, const ASUUID *ns,
void *name, ASByteCount bytes)
```

Description

Generates a universal unique identifier (UUID) for a block of data (a *name*) in a context (a *namespace*).

Parameters

dst	(Filled by the method) The unique identifier created from the name.
ns	A namespace or context meaningful to the client.
name	A pointer to an arbitrary block of data to be identified by the UUID.
bytes	The number of bytes in name .

Return Value

true if the UUID is successfully created, **false** otherwise.

Header File

ASCalls.h

Related Methods

[ASUUIDFromCString](#)
[ASUUIDGenFromHash](#)
[ASUUIDGenUnique](#)
[ASUUIDToCString](#)
[AVAppGetUUID](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



ASUUIDGenUnique

ASBool ASUUIDGenUnique (**ASUUID** *dst)

Description

Generates a unique identifier (UUID).

Parameters

dst	(Filled by the method) The unique identifier created from the hash.

Return Value

true if the UUID is successfully created, **false** otherwise.

Header File

ASCalls.h

Related Methods

[ASUUIDFromCString](#)
[ASUUIDGenFromHash](#)
[ASUUIDGenFromName](#)
[ASUUIDToCString](#)
[AVAppGetUUID](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

ASUUIDToCString

```
void ASUUIDToCString (char *dst, const ASUUID *src)
```

Description

Generates a **NULL**-terminated C string from the unique identifier (UUID) for a user or session.

Parameters

dst	(Filled by the method) A NULL -terminated string from which to generate the unique identifier, in the following form: f81d4fae-7dec-11d0-a765-00a0c91e6bf6 The string must be at least the length specified by ASUUIDMaxStringLen .
src	The unique identifier from which to generate the string .

Return Value

None

Header File

ASCalls.h

Related Methods

[ASUUIDFromCString](#)
[ASUUIDGenFromHash](#)
[ASUUIDGenFromName](#)
[ASUUIDGenUnique](#)
[AVAppGetUUID](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

Errors

ASGetErrorString

```
const char* ASGetErrorString (ASErrorCode errorCode,
char* buffer, ASArraySize maxLength);
```

Description

Gets a string describing the specified error/exception .

Parameters

errorCode	The exception whose error string is obtained. This must be a full error code, built with the ErrBuildCode macro or a user-defined exception returned from ASRegisterErrorString . See Errors for a list of predefined exceptions.
buffer	<i>(Filled by the method)</i> Buffer into which the string is written.
maxLength	The number of characters that buffer can hold.

Return Value

A useful pointer to buffer, or **NULL** if the error is not known.

Header File

ASCalls.h

Related Methods

[ASGetExceptionErrorCode](#)
[ASRegisterErrorString](#)
[ASRaise](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

ASGetExceptionErrorCode

[ASErrorCode](#) ASGetExceptionErrorCode (void);

Description

Gets the error code for the exception most recently raised. See [Errors](#) for a list of predefined exceptions.

Parameters

None

Return Value

Exception error code.

Header File

CorCalls.h

Related Methods

[ASRaise](#)
[ASGetErrorString](#)
[ASRegisterErrorString](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

ASPopExceptionFrame

```
void ASPopExceptionFrame (void);
```

Description

Pops an exception frame off the stack.

NOTE: You will probably never call **ASPopExceptionFrame** directly; it is called for you as appropriate from within the [HANDLER](#), [E_RETURN](#) and [E_RTRN_VOID](#) macros.

Parameters

None

Return Value

None

Header File

CorCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

ASPushExceptionFrame

```
void ASPushExceptionFrame (void* asEnviron,
ACRestoreEnvironProc restoreFunc);
```

Description

Pushes an exception frame buffer and a frame-restoration callback onto the stack. The **restoreFunc** should be a function matching the following prototype:

NOTE: You will probably never call **ASPushExceptionFrame** directly; use the **DURING** macro instead.

Parameters

asEnviron	Represents a stack environment that is restored if an exception occurs. On Windows and Macintosh, this is a jmp_buf ; A jmp_buf is an array of integers.
restoreFunc	Should be a function matching the following prototype: ACCB1 void ACCB2 RestorePlugInFrame (void* asEnviron)

Return Value

None

Header File

CorCalls.h

Product

reader, base, pro, pdfi

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



ASRaise

```
void ASRaise (ASErrorCode error);
```

Description

Raises an exception. Plug-ins can raise any exception defined in the **AcroErr.h** header file using the **ErrBuildCode** macro, or can define their own exceptions using **ASRegisterErrorString**. See **Errors** for a list of predefined exceptions.

If the code that calls **ASRaise** gets control as a result of a non-Acrobat event (such as a drag and drop event on some platforms), this method fails since there is no Acrobat viewer code in the stack to handle the exception.

Parameters

error	Error code for the exception to raise. Error codes have three parts: severity, system, and error number. Use ErrBuildCode to build an error code for an existing error.

Return Value

None

Header File

CorCalls.h

Related Methods

[ASGetErrorString](#)
[ASRegisterErrorString](#)

Related Macros

[RERAISE](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

ASRegisterErrorString

```
ASErrorCode ASRegisterErrorString (ASErrSeverity severity,
const char* errorString);
```

Description

Registers a new error and string. The error can be used to raise a plug-in-specific exception using [ASRaise](#). When the exception is raised, its error string can be retrieved using [ASGetErrorString](#) and reported to the user using [AAlertNote](#).

The error system is automatically forced to be **ErrSysXtn**. (See the list of [Error Systems](#).)

The error is automatically assigned an error code that is not used by any other plug-in (in the current implementation, the Acrobat viewer increments a counter each time any plug-in requests an error code, and returns the value of the counter). As a result, plug-ins cannot rely on being assigned the same error code each time the Acrobat viewer is launched.

Parameters

severity	The severity of the error being defined. Must be one of the Severities .
errorString	The string describing the exception. This string is used by ASGetErrorString . errorString is copied by ASRegisterErrorString ; it may be freed by the plug-in after registering the error.

Return Value

The newly created error code. Plug-ins should assign the error code returned by this method to a variable if they wish to use the error code later in the current session.

Header File

ASCalls.h

Related Methods

[ASGetErrorString](#)
[ASRaise](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



Fixed Point Math

ASCStringToFixed

```
ASFixed ASCStringToFixed (const char* s);
```

Description

Converts a Cstring to a fixed point number. Processes the string from left to right only until the first invalid character is located (for example, a-z, A-Z).

Parameters

s	A Cstring to convert.

Return Value

Fixed number corresponding to **s**. Returns 0 if the string does not contain any valid number.

Header File

ASCalls.h

Related Methods

[ASFixedToCString](#)

Related Macros

[ASFixedRoundToInt16](#)
[ASFixedRoundToInt32](#)
[ASFixedTruncToInt16](#)
[ASFixedTruncToInt32](#)
[ASFixedToFloat](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



ASFixedDiv

```
ASFixed ASFixedDiv (ASFixed a, ASFixed b);
```

Description

Divides two fixed numbers.

Parameters

a	The dividend.
b	The divisor.

Return Value

The quotient **a** / **b**.

Header File

ASCalls.h

Related Methods

[ASFixedMul](#)

Related Macros

[Fixed Numbers](#)
[ASFixedRoundToInt16](#)
[ASFixedRoundToInt32](#)
[ASFixedTruncToInt16](#)
[ASFixedTruncToInt32](#)
[ASFixedToFloat](#)
[ASFloatToFixed](#)
[ASInt16ToFixed](#)
[ASInt32ToFixed](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

ASFixedMatrixConcat

```
void ASFixedMatrixConcat (ASFixedMatrixP result,  
    const ASFixedMatrixP m1, const ASFixedMatrixP m2);
```

Description

Multiplies two matrices.

Parameters

result	(Filled by the method) Pointer to matrix m2 × m1 . It is OK for result to point to the same location as either m1 or m2 .
m1	Pointer to the ASFixedMatrix value for the first matrix to multiply.
m2	Pointer to the ASFixedMatrix value for the second matrix to multiply.

Return Value

None

Header File

ASCalls.h

Related Methods

- [ASFixedMatrixInvert](#)
- [ASFixedMatrixTransform](#)
- [ASFixedMatrixTransformRect](#)

Related Macros

- [Fixed Numbers](#)
- [ASFixedRoundToInt16](#)
- [ASFixedRoundToInt32](#)
- [ASFixedTruncToInt16](#)
- [ASFixedTruncToInt32](#)
- [ASFixedToFloat](#)
- [ASFloatToFixed](#)
- [ASInt16ToFixed](#)
- [ASInt32ToFixed](#)

Product

reader, base, pro, pdf

**Availability**

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



ASFixedMatrixInvert

```
void ASFixedMatrixInvert (ASFixedMatrixP result,
const ASFixedMatrixP m);
```

Description

Inverts a matrix.

If a matrix is nearly singular (that is, has a determinant nearly zero), inverting and re-inverting the matrix may not yield the original matrix.

Parameters

result	(Filled by the method) Pointer to m -1. It is OK for <i>result</i> to point to the same location as m .
m	Pointer to the ASFixedMatrix to invert.

Return Value

None

Header File

ASCalls.h

Related Methods

[ASFixedMatrixConcat](#)
[ASFixedMatrixTransform](#)
[ASFixedMatrixTransformRect](#)

Related Macros

[ASFixedRoundToInt16](#)
[ASFixedRoundToInt32](#)
[ASFixedTruncToInt16](#)
[ASFixedTruncToInt32](#)
[ASFixedToFloat](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

ASFixedMatrixTransform

```
void ASFixedMatrixTransform (ASFixedPointP result,
                             const ASFixedMatrixP m, const ASFixedPointP p);
```

Description

Transforms the point **p** through the matrix **m**, puts result in **result.p** and **result** can point to the same place.

Parameters

result	<i>(Filled by the method)</i> Pointer to the ASFixedPoint containing the result of transforming p through m . It is OK for result to point to the same location as m .
m	Pointer to the ASFixedMatrix through which p is transformed.
p	Pointer to the ASFixedPoint representing the point to transform through m .

Return Value

None

Header File

ASCalls.h

Related Methods

[ASFixedMatrixTransformRect](#)
[ASFixedMatrixConcat](#)
[ASFixedMatrixInvert](#)

Related Macros

[Fixed Numbers](#)
[ASFixedRoundToInt16](#)
[ASFixedRoundToInt32](#)
[ASFixedTruncToInt16](#)
[ASFixedTruncToInt32](#)
[ASFixedToFloat](#)
[ASFloatToFixed](#)
[ASInt16ToFixed](#)
[ASInt32ToFixed](#)

Product

reader, base, pro, pdf

**Availability**

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to 0x00020000 or higher.

ASFixedMatrixTransformRect

```
void ASFixedMatrixTransformRect (ASFixedRectP result,
                                const ASFixedMatrixP m, const ASFixedRectP rectIn);
```

Description

Transforms a rectangle through a matrix.

Parameters

result	(Filled by the method) Pointer to the ASFixedRect containing the smallest bounding box for the transformed rectangle. It is OK for result to point to the same location as m result will always have bottom < top and left < right .
m	Pointer to the ASFixedMatrix containing the matrix through which r is transformed.
rectIn	Pointer to the ASFixedRect containing the rectangle to transform through m .

Return Value

None

Header File

ASCalls.h

Related Methods

- [ASFixedMatrixTransform](#)
- [ASFixedMatrixConcat](#)
- [ASFixedMatrixInvert](#)

Related Macros

- [Fixed Numbers](#)
- [ASFixedRoundToInt16](#)
- [ASFixedRoundToInt32](#)
- [ASFixedTruncToInt16](#)
- [ASFixedTruncToInt32](#)
- [ASFixedToFloat](#)
- [ASFloatToFixed](#)
- [ASInt16ToFixed](#)
- [ASInt32ToFixed](#)

Product

reader, base, pro, pdf

**Availability**

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



ASFixedMul

```
ASFixed ASFixedMul (ASFixed a, ASFixed b);
```

Description

Multiplies two fixed numbers.

Parameters

a, b	The two numbers to multiply.

Return Value

The product of **a** and **b**.

Header File

ASCalls.h

Related Methods

[ASFixedDiv](#)

Related Macros

[Fixed Numbers](#)
[ASFixedRoundToInt16](#)
[ASFixedRoundToInt32](#)
[ASFixedTruncToInt16](#)
[ASFixedTruncToInt32](#)
[ASFixedToFloat](#)
[ASFloatToFixed](#)
[ASInt16ToFixed](#)
[ASInt32ToFixed](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

ASFixedToCString

```
void ASFixedToCString (ASFixed f, char* s,
os_size_t maxLength, ASInt16 precision);
```

Description

Converts a fixed number to a CString .

Parameters

f	The fixed number to convert.
s	<i>(Filled by the method)</i> The string corresponding to f .
maxLength	The maximum number of characters that s can contain.
precision	The number of digits to retain in the converted number. NOTE: The precision for Mac OS numbers is valid to 9 significant digits.

Return Value

None

Header File

ASCalls.h

Related Methods

[ASCStringToFixed](#)

Related Macros

[Fixed Numbers](#)
[ASFloatToFixed](#)
[ASInt16ToFixed](#)
[ASInt32ToFixed](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



HFT and HFTServer

ASExtensionMgrGetHFT

```
HFT ASExtensionMgrGetHFT (ASAtom name, ASUns32 version);
```

Description

Gets the specified version of the Host Function Table (**HFT**) that has the specified name. If you want to get one of the Acrobat viewer's built-in **HFT**s, use the predefined global variables for the **HFT Values** instead of this method.

Parameters

name	The name of the HFT to obtain.
version	The version number of the HFT to obtain.

Return Value

The specified **HFT**, or **NULL** if the **HFT** does not exist.

Header File

CorCalls.h

Related Methods

[HFTReplaceEntry](#)
[HFTReplaceEntryEx](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



HFTDestroy

```
void HFTDestroy (HFT hft);
```

Description

Destroys an existing **HFT** by freeing all the **HFT**'s memory. Call this method only if you are absolutely sure that neither your plug-in nor any other plug-in will use the **HFT** again. Because this is usually impossible to know, plug-ins should not destroy **HFTs**. It is even dangerous to destroy an **HFT** at unload time, because the order in which plug-ins are unloaded is not specified.

Parameters

hft	The HFT to destroy.

Return Value

None

Header File

ASCalls.h

Related Methods

[HFTNew](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



HFTGetReplacedEntry

```
HFTEntry HFTGetReplacedEntry (HFT hft, Selector sel,
HFTEntry replacer);
```

Description

Gets the **HFTEntry** that was replaced by the specified **HFTEntry** in the specified entry. Plug-ins should generally not use this method directly, but use the [CALL_REPLACED_PROC](#) macro instead.

It is necessary to specify both a selector (the index of an entry in the **HFT**'s table of callback pointers) and an **HFTEntry** (a callback pointer) because a method may be replaced several times, and the various replacement methods are kept in a linked list. The selector determines which linked list is examined, and the **HFTEntry** determines the entry in the linked list to return.

Parameters

hft	The HFT in which a replaced entry is retrieved. See HFTReplaceEntry for more information.
sel	The selector whose previous value is obtained. See HFTReplaceEntry for more information.
replacer	The HFTEntry for which the previous value is obtained.

Return Value

The entry present prior to being replaced by **replacer**. Returns **NULL** if the entry has not been replaced.

Header File

ASCalls.h

Related Methods

[ASExtensionMgrGetHFT](#)

Related Macros

[CALL_REPLACED_PROC](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

HFTGetVersion

```
ASVersion HFTGetVersion (HFT hft);
```

Description

Returns the version of the **HFT**, if available.

Parameters

hft	The HFT whose version is obtained.

Return Value

The version number if the HFT is valid and the version is available, **HFT_ERROR_NO_VERSION** otherwise.

Header File

ASCalls.h

Related Methods

[HFTNewEx](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



HFTIsValid

```
ASBool HFTIsValid (HFT hft);
```

Description

Tests whether an **HFT** is valid.

Parameters

hft	The HFT to test.

Return Value

true if **hft** is valid, **false** otherwise.

Header File

ASCalls.h

Example

```
result = HFTIsValid(gMyPluginHFT);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



HFTNew

```
HFT HFTNew (HFTServer hftServer, ASInt32 numSelectors);
```

Description

Creates a new **HFT** by calling the specified **HFT** server's **HFTServerProvideHFTProc**.

Parameters

hftServer	The HFT server for the HFT being created. The HFT server must have been created previously using HFTServerNew .
numSelectors	The number of entries in the new HFT . This determines the number of methods that the HFT can contain; each method occupies one entry.

Return Value

The newly created **HFT**.

Header File

ASCalls.h

Related Methods

[ASExtensionMgrGetHFT](#)
[HFTDestroy](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



HFTNewEx

```
HFT HFTNewEx (HFTServer hftServer, HFTData data);
```

Description

Extends **HFTNew** with version information in Acrobat 6. Creates a new **HFT** by calling the specified **HFT** server's **HFTServerProvideHFTProc**.

Parameters

hftServer	The HFT server for the HFT being created. The HFT server must have been created previously using HFTServerNew .
data	The data to pass to the server, which includes: • The number of entries in the new HFT, which determines the number of methods that the HFT can contain. Each method occupies one entry. • The HFT version.

Return Value

The newly created **HFT**.

Header File

ASCalls.h

Related Methods

HFTNew
ASExtensionMgrGetHFT
HFTDestroy

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

HFTReplaceEntry

```
void HFTReplaceEntry (HFT hft, Selector sel,
HFTEntry newEntry, ASFlagBits flags);
```

Description

Replaces the specified entry in the specified **HFT**. This allows a plug-in to override and replace certain methods in Acrobat's API. See [Replaceable Methods](#) for a list of replaceable methods. This method can be used from anywhere in the plug-in, but it makes the most sense for most plug-ins to replace methods in the **importReplaceAndRegisterCallback** procedure. Plug-ins register their HFTs in the export callback, but the code to populate the function table is only executed when the first client requests the HFT.

Plug-ins can use the **REPLACE** macro instead of calling **HFTReplaceEntry** directly.

All plug-ins, and Acrobat itself, share a single copy of each **HFT**. As a result, when a plug-in replaces the implementation of a method, all other plug-ins and Acrobat also use the new implementation of that method. In addition, once a method's implementation has been replaced, there is no way to remove the new implementation without restarting Acrobat.

NOTE: The **CALL_REPLACED_PROC** macro is available to call the previous HFT entry function that was replaced.

Parameters

hft	The HFT in which a method is replaced. Use ASExtensionMgrGetHFT to get the HFT , given its name. For the HFTs built into the Acrobat viewer, global variables containing the HFTs have been defined, so you can skip calling ASExtensionMgrGetHFT for these HFTs . See HFT Values for a list of these global variables.
sel	The entry in the HFT to replace, derived from the method's name by appending SEL . For example, to replace AVAlert , sel must be AVAlertSEL .
newEntry	The function to replace the current one. The function pointer must be converted to an HFTEntry using the ASCallbackCreateReplacement macro.
flags	The new entry's properties. Currently, only HFTEntryReplaceable is defined.

Return Value

None

**Known Exceptions**

[xmErrCannotReplaceSelector](#)

Header File

ASCalls.h

Related Methods

[ASExtensionMgrGetHFT](#)

[HFTGetReplacedEntry](#)

Related Macros

[CALL_REPLACED_PROC](#)

[REPLACE](#)

[ASCallbackCreateReplacement](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

HFTReplaceEntryEx

```
void HFTReplaceEntryEx (HFT hft, Selector sel,
HFTEntry newEntry, ASExtension extension, ASFlagBits flags);
```

Description

New version of [HFTReplaceEntry](#). Adds the **extension** argument.

Plug-ins can use the [REPLACE](#) macro instead of calling **HFTReplaceEntryEx** directly.

NOTE: The [CALL_REPLACED_PROC](#) macro is available to call the previous HFT entry function that was replaced.

Parameters

hft	The HFT in which a method is replaced. Use ASExtensionMgrGetHFT to get the HFT , given its name. For the HFTs built into the Acrobat viewer, global variables containing the HFTs have been defined, so you can skip calling ASExtensionMgrGetHFT for these HFTs . See HFT Values for a list of these global variables.
sel	The entry in the HFT to replace, derived from the method's name by appending SEL . For example, to replace AAlert , sel must be AAlertSEL .
newEntry	The function to replace the current one. The function pointer must be converted to an HFTEntry using the ASCallbackCreateReplacement macro.
extension	Plug-ins should pass in gExtensionID for this parameter (see the code for the Acrobat 5.0 version of the REPLACE macro). This parameter is stored by Acrobat so that any entries that were replaced by a plug-in can be unreplaced in the event that the plug-in unloads.
flags	The new entry's properties. Currently, only HFTEntryReplaceable is defined.

Return Value

None

Known Exceptions

[xmErrCannotReplaceSelector](#)

Header File

ASCalls.h

**Related Methods**

[ASExtensionMgrGetHFT](#)
[HFTGetReplacedEntry](#)
[HFTReplaceEntry](#)
[HFTUnreplaceEntry](#)

Related Macros

[CALL_REPLACED_PROC](#)
[REPLACE](#)
[ASCallbackCreateReplacement](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.



HFTServerDestroy

```
void HFTServerDestroy (HFTServer hftServer);
```

Description

Destroys an **HFT** server. Call this method only if the **HFT** will not be used again.

Parameters

hftServer	The HFT server to destroy.

Return Value

None

Header File

ASCalls.h

Related Methods

[HFTServerNew](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

HFTServerNew

```
HFTServer HFTServerNew (const char* name,
HFTServerProvideHFTPProc serverProc,
HFTServerDestroyProc destroyProc, void* clientData);
```

Description

Creates a new Host Function Table (**HFT**) server. An **HFT** server is responsible for creating an instance of an **HFT** with the specified version number, and destroying the **HFT**.

Parameters

name	The new HFT server's name.
serverProc	(Required) User-supplied callback that provides an HFT when given a version number. This procedure is called by ASExtensionMgrGetHFT when another plug-in imports the HFT.
destroyProc	(Optional) User-supplied callback that destroys the specified HFT (generally, this means de-allocating the memory associated with the HFT). This procedure is called by HFTDestroy .
clientData	Pointer to user-supplied data to pass to the HFT server.

Return Value

The newly created **HFT** server.

Header File

ASCalls.h

Related Methods

[HFTServerDestroy](#)
[HFTNew](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

HFTUnreplaceEntry

```
void HFTUnreplaceEntry (HFT hft, Selector sel,
HFTEntry oldEntry, ASExtension extension);
```

Description

Removes the **oldEntry** item from **hft** at **sel** if the **extension** fields match. Allows HFT replacements to be undone in cases such as with the DigSig plug-in, which replaces a method that Acrobat could use *after* DigSig unloads..

Parameters

hft	The HFT in which a method is un-replaced. Use ASExtensionMgrGetHFT to get the HFT , given its name. For the HFTs built into the Acrobat viewer, global variables containing the HFTs have been defined, so you can skip calling ASExtensionMgrGetHFT for these HFTs . See HFT Values for a list of these global variables.
sel	The entry in the HFT to un-replace, derived from the method's name by appending SEL . For example, to replace AValert , sel must be AValertSEL .
oldEntry	The old function to be replaced. The function pointer must be converted to an HFTEntry using the ASCallbackCreateReplacement macro.
extension	Object of type ASExtension .

Header File

ASCalls.h

Related Methods

[ASExtensionMgrGetHFT](#)
[HFTGetReplacedEntry](#)
[HFTReplaceEntry](#)
[HFTReplaceEntryEx](#)

Related Macros

[REPLACE](#)
[ASCallbackCreateReplacement](#)

Product

reader, base, pro, pdf

**Availability**

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



Memory Allocation

ASfree

```
void ASfree (void* ptr);
```

Description

Frees the specified memory block.

Parameters

ptr	The block of memory to free.

Return Value

None

Header File

ASCalls.h

Related Methods

[ASmalloc](#)

[ASrealloc](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



ASmalloc

```
void* ASmalloc (os_size_t nBytes);
```

Description

Allocates and returns a pointer to a memory block containing the specified number of bytes.

Parameters

nBytes	The number of bytes for which space is allocated.

Return Value

Pointer to the allocated memory. Returns **NULL** on failure.

Header File

ASCalls.h

Related Methods

[ASrealloc](#)
[ASfree](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



ASrealloc

```
void* ASrealloc (void* ptr, os_size_t newNBytes);
```

Description

If possible, extends the given block and simply returns **ptr**. Otherwise, allocates a new block of **newNBytes** bytes, copies the contents at the old pointer into the new block, frees the old pointer, and returns the pointer to the new block. If a new block cannot be allocated, the call fails and **ptr** is not freed. Reallocating a block to a smaller size will never fail.

Parameters

ptr	The existing memory block.
newNBytes	The number of bytes the memory block must be able to hold.

Return Value

Pointer to memory block.

Header File

ASCalls.h

Related Methods

[ASmalloc](#)

[ASfree](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



String Encoding

ASHostMBLen

```
ASInt32 ASHostMBLen (ASHostEncoding encoding, ASUns8 byte);
```

Description

Tells you whether the given **byte** is a lead byte of a multi-byte character and how many tail bytes follow.

When parsing a string in a host encoding you must keep in mind that the string could be in a variable length multi-byte encoding. In such an encoding (for example, **Shift-JIS**) the number of bytes required to represent a character varies on a character-by-character basis. To parse such a string you must start at the beginning and, for each byte, ask whether that byte represents a character or is the first byte of a multi-byte character. If the byte is a "lead byte" for a multi-byte character you must also compute how many bytes will follow the lead byte to make up the entire character. Currently the API provides a call (**PDHostMBLen**) that performs these computations but only if the encoding in question is the OS encoding (as returned by **PDGetHostEncoding**). **ASHostMBLen** allows you to ask this question for any byte in any host encoding.

See below for an example of how to parse a multi-byte string.

Parameters

encoding	Host encoding type.
byte	The first byte of a multi-byte character.

Return Value

The number of additional bytes required to form the character. For example, if the encoding is a double-byte encoding the return value will be **1** for a two-byte character and **0** for a one-byte character. For Roman encodings the return value will always be **0**.

NOTE: **ASHostMBLen** cannot confirm that required number of trailing bytes actually follow the first byte. If you are parsing a multi-byte string make sure your code will stop at the first null (zero) byte even if it appears immediately after the lead byte of a multi-byte character.

Header File

ASCalls.h

Related Methods

[PDGetHostEncoding](#)
[PDHostMBLen](#)

Example

```

ASUns8 myString[12] = "Hello World";
ASUns8 *theString = myString;

ASHostEncoding encoding = 0; //some encoding

while(*theString) // All multi-byte encodings are null terminated
{
    ASUns8 thisByte = *theString++;
    ASUns32 thisChar = thisByte;
    ASInt32 trailingBytes = ASHostMBLen(encoding, thisByte);
    while (trailingBytes--)
    {
        thisByte = *theString++;
        if(thisByte == 0)
            ; //error
        else thisChar = (thisChar << 8) | thisByte;
    }
    //At this point you have an entire character in
    //thisChar; it will then be replaced by the next character
    //until you reach the end of the string
}

```

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROSUPPORT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASIsValidUTF8

```
ASBool ASIsValidUTF8 (const ASUns8* cIn, ASCount cInLen);
```

Description

Tests whether or not the bytes in the string conform to the Unicode UTF-8 encoding form. The method does not test whether the string is **null**-terminated.

Parameters

cIn	The string.
cInLen	The length of the string in bytes, not including the null byte at the end.

Return Value

true if the bytes in the string conform to the Unicode UTF-8 encoding form, **false** otherwise.

Header File

ASExtraCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AScriptFromHostEncoding

```
AScript AScriptFromHostEncoding (ASHostEncoding osScript);
```

Description

Converts from a host encoding type to an **AScript** value. On Windows, the host encoding is a CHARSET id. On Mac OS, the host encoding is a script code.

Parameters

osScript	The host encoding type.

Return Value

The new **AScript** value.

Header File

ASExtraCalls.h

Related Methods

[AScriptToHostEncoding](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

ASScriptToHostEncoding

[ASHostEncoding](#) ASScriptToHostEncoding ([ASScript](#) asScript);

Description

Converts from an **ASScript** code to a host encoding type. On Windows, the host encoding is a CHARSET id. On Mac OS, the host encoding is a script code.

Parameters

asScript	The script value.

Return Value

The new host encoding type.

Header File

ASExtraCalls.h

Related Methods

[ASScriptFromHostEncoding](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



AV Layer Methods

General

AVActionHandler

AVAlert

AVAnnotHandler

AVApp

AVCommand

AVConversion

AVCrypt

AVDoc

AVGrafSelect

AVMenu

AVMenubar

AVMenuItem

AVPageView

AVSweetPea

AVSys

AVTool

AVToolBar

AVToolButton

AVUndo

AVWindow

General

AVAcquireSpecialFilePathName

```
ASInt32 AVAcquireSpecialFilePathName
(AVSpecialCategory category, AVSpecialFolder folder,
const char* file, ASFileSys* fileSys, ASPathName* pathName);
```

Description

Obtains the pathname for a file in a special folder. It is the caller's responsibility to release the **ASPathName**. This method may be used even if the associated file does not exist.

NOTE: In Acrobat 6.0, this method is superseded by
[AVAcquireSpecialFilePathNameWithASText](#).

Parameters

category	The folder category. See AVSpecialCategory . Only certain combinations of category/folder are allowed—see AVSpecialError .
folder	The folder in which the file is located. See AVSpecialFolder . Only certain combinations of category/folder are allowed—see AVSpecialError .
file	The name of file (including extension) that you want to access.
fileSys	<i>(Filled by the method)</i> The file system through which pathName was obtained.
pathName	<i>(Filled by the method)</i> ASPathName associated with the file.

Return Value

One of the **AVSpecialError** status codes. The **fileSys** and **pathName** variables will only be valid if the method returns **kAVSEOkay** or **kAVSEDoesntExist**. **kAVSEDoesntExist** is returned if the **ASPathName** was created but the associated file doesn't exist.

Header File

AVCalls.h

Related Methods

[AVAcquireSpecialFilePathNameWithASText](#)
[AVAcquireSpecialFolderPathName](#)

**Product**

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVAcquireSpecialFilePathNameWithASText

```
ASErrorCode AVAcquireSpecialFilePathNameWithASText
(AVSpecialCategory cat, AVSpecialFolder fld,
const ASText file, ASFileSys* fileSys, ASPathName* pathName);
```

Description

Obtains the pathname for a file in a special folder. It is the caller's responsibility to release the **ASPathName**. This method may be used even if the associated file does not exist.

This method supersedes **AVAcquireSpecialFilePathName** in Acrobat 6.0.

Parameters

cat	The folder category. See AVSpecialCategory . Only certain combinations of category/folder are allowed—see AVSpecialError .
fld	The folder in which the file is located. See AVSpecialFolder . Only certain combinations of category/folder are allowed—see AVSpecialError .
file	A text object containing the name of the file (including extension) that you want to access.
fileSys	<i>(Filled by the method)</i> The file system through which pathName was obtained.
pathName	<i>(Filled by the method)</i> ASPathName associated with the file.

Return Value

One of the **AVSpecialError** status codes. The **fileSys** and **pathName** variables will only be valid if the method returns **kAVSEOkay** or **kAVSEDoesntExist**. **kAVSEDoesntExist** is returned if the **ASPathName** was created but the associated file doesn't exist.

Header File

AVCalls.h

Related Methods

AVAcquireSpecialFilePathName
AVAcquireSpecialFolderPathName

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVAcquireSpecialFolderPathName

```

ASErrorCode AVAcquireSpecialFolderPathName
(AVSpecialCategory category, AVSpecialFolder folder,
ASBool bCreate, ASFileSys *fileSys, ASPathName *pathName);

```

Description

Obtains the pathname of a special folder. This method cannot be used to obtain the **ASPathName** of a folder that does not exist. It is the caller's responsibility to release the **ASPathName**.

Parameters

category	The folder category. See AVSpecialCategory . Only certain combinations of category/folder are allowed—see AVSpecialError .
folder	The folder. See AVSpecialFolder . Only certain combinations of category/folder are allowed—see AVSpecialError .
bCreate	Create folder if it does not exist.
fileSys	<i>(Filled by the method)</i> The file system through which pathName was obtained.
pathName	<i>(Filled by the method)</i> ASPathName associated with the folder.

Return Value

One of the **AVSpecialError** status codes. Returns **kAVSEOkay** if the method executed without error. The **fileSys** and **pathName** variables will only be valid if the method returns **kAVSEOkay**. If **bCreate** is false and the folder does not exist, **kAVSEError** is returned on Windows while **kAVSEDoesntExist** is returned on the Macintosh.

Header File

AVCalls.h

Related Methods

[AVAcquireSpecialFilePathName](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



AVDestInfoDestroy

```
void AVDestInfoDestroy (AVDestInfo destInfo);
```

Description

Releases the memory associated with a destination.

Parameters

destInfo	The destination info object to destroy.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewToDestInfo](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020003** or higher.

AVExtensionAcquireInfo

```
AVExtensionInfo AVExtensionAcquireInfo(AVArraySize index);
```

Description

Fills an [AVExtensionInfoRec](#) structure with some information about a client. It is the caller's responsibility to release the memory associated with the contents of the returned structure by calling [AVExtensionReleaseInfo](#).

NOTE: This API is not supported on UNIX platforms.

Parameters

index	The index of the client to retrieve information for. The minimum value for index is zero; the maximum is the return value of AVExtensionGetNumPlugIns - 1.

Return Value

The structure containing information about the client.

Header File

AVCalls.h

Related Methods

[AVExtensionReleaseInfo](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



AVExtensionGetNumPlugIns

[AVArraySize](#) AVExtensionGetNumPlugIns(void);

Description

Returns the number of clients loaded by Acrobat.

NOTE: This method is not supported on UNIX platforms.

Parameters

None

Return Value

The number of clients.

Header File

AVCalls.h

Related Methods

[AVExtensionAcquireInfo](#)

[AVExtensionReleaseInfo](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVExtensionReleaseInfo

```
void AVExtensionReleaseInfo(AVExtensionInfo extensionInfo);
```

Description

Releases the memory associated with the contents of **extensionInfo**.

NOTE: This API is not supported on UNIX platforms.

Parameters

extensionInfo	The AVExtensionInfoRec structure to release.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVExtensionAcquireInfo](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



AVIdentityGetText

```
ASText AVIdentityGetText (AVIdentity id, AText text);
```

Description

Gets the value of a particular aspect of the active user's identity. Valid keys are login name, name, corporation and email.

Parameters

id	The AVIdentity key to retrieve.
text	<i>(Filled by the method)</i> ASText object to hold the text associated with the id key.

Return Value

A useful handle to the **text** parameter.

Header File

AVCalls.h

Related Methods

[AVIdentitySetText](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVIdentitySetText

```
void AVIdentitySetText (AVIdentity id, AText text);
```

Description

Sets the value of a particular aspect of the active user's identity.

For obvious reasons, it is not possible to modify the **kAVILoginName** value through this method.

Parameters

id	The AVIdentity key to set.
text	The new value for the key.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVIdentityGetText](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVRectHandleHitTest

```
ASInt16 AVRectHandleHitTest (AVDevRect* rect,
ASBool bMidpoints, AVDevCoord x, AVDevCoord y);
```

Description

Tests to see if the given point lies on any of the control handles of **rect**.

Parameters

rect	The rectangle to test.
bMidpoints	Pass true if interested in the midpoint handles. If false and a midpoint handle is hit, kAVDragRect is returned.
x	The x-coordinate of the point.
y	The y-coordinate of the point.

Return Value

If the point is not within the rectangle, **-1** is returned. If the point is within the rectangle, but a not over a handle, **kAVDragRect** is returned. If the point is within a handle region, one of the **AVRectHandleType** constants is returned.

Header File

AVCalls.h

Related Methods

[AVPageViewSnapRect](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVUtilGetBaseNameAndExtensionByPathName

```
ASBool AVUtilGetBaseNameAndExtensionByPathName
(ASFileSys fileSys, ASPathName pathName,
AVTArraySize numAddExt, char** addExt, char** baseName,
char** baseExt);
```

Description

Parses a path name to obtain the base filename and extension for a particular file. The function enumerates all registered “convertToPDF” and “convertFromPDF” handlers to find a matching extension for the file passed in. This function allocates memory for the filename and extension. It is the caller’s responsibility to free the memory allocated by the method.

Parameters

fileSys	The file system from which path was obtained. Pass NULL to use the default file system.
pathName	The path containing the filename.
numAddExt	The number of additional extensions to search through.
addExt	An array of null-terminated strings with extensions to search through.
baseName	<i>(Allocated and filled by the method)</i> A buffer containing the filename. Can be NULL if you do not want the base filename.
baseExt	<i>(Allocated and filled by the method)</i> A buffer containing the file extension. Can be NULL if you do not want the base file extension.

Return Value

true if the file info was successfully extracted from **path**, **false** otherwise.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVUtilGetBaseNameAndExtensionByString

```
ASBool AVUtilGetBaseNameAndExtensionByString (char* fileName,
AVTArraySize numAddExt, char** addExt, char** baseName,
char** baseExt);
```

Description

Parses a file name string to obtain the base filename path and extension for a particular file. The function enumerates all registered “convertToPDF” and “convertFromPDF” handlers to find a matching extension for the file passed in. This function allocates memory for the filename and extension. It is the caller’s responsibility to free the memory allocated by the method.

NOTE: `fileName` is modified by the implementation.

Parameters

fileName	The file name string to parse.
numAddExt	The number of additional extensions to search through.
addExt	An array of null-terminated strings with extensions to search through.
baseName	<i>(Allocated and filled by the method)</i> A buffer containing the filename path; that is, the path without the extension, such as <code>c:\folder\file</code> . Can be NULL if you don't want the base filename path.
baseExt	<i>(Allocated and filled by the method)</i> A buffer containing the file extension. Can be NULL if you don't want the base file extension.

Return Value

true if the file info was successfully extracted from `fileName`, **false** otherwise.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVUtilGetBaseNameAndExtensionEx

```
ASBool AVUtilGetBaseNameAndExtensionEx
(const ASFileSys fileSys, const ASPathName pathName,
const ASText fileName, ASInt32 numAddExt,
const char *const *addExt, ASText baseName, char **baseExt);
```

Description

Parses a path name to obtain the base filename and extension for a particular file. The function enumerates all registered “convertToPDF” and “convertFromPDF” handlers to find a matching extension for the file passed in. This function allocates memory for the filename and extension. It is the caller’s responsibility to free the memory allocated by the method.

NOTE: This method extends [AVUtilGetBaseNameAndExtensionByPathName](#) in Acrobat 6.0 to use constant types and a text-object file name.

Parameters

fileSys	The file system from which path was obtained. Pass NULL to use the default file system.
pathName	The path containing the filename.
fileName	The filename as a constant text object.
numAddExt	The number of additional extensions to search through.
addExt	An array of null-terminated strings with extensions to search through.
baseName	<i>(Allocated and filled by the method)</i> A buffer containing the filename. Can be NULL if you do not want the base filename.
baseExt	<i>(Allocated and filled by the method)</i> A buffer containing the file extension. Can be NULL if you do not want the base file extension.

Return Value

true if the file info was successfully extracted from **path**, **false** otherwise.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVActionHandler

AVActionHandlerGetProcs

```
AVActionHandlerProcs AVActionHandlerGetProcs  
(AVActionHandler handler) ;
```

Description

Gets a structure containing pointers to the action handler's procedures.

This method is useful when you want to subclass an already-registered action handler. Acrobat releases the previously-registered action handler when you call [AVAppRegisterActionHandler](#), so you must copy the previously-registered action handler to a new action handler structure, which you use in the call to [AVAppRegisterActionHandler](#).

Parameters

handler	The action handler whose procedures are obtained.

Return Value

Pointer to a structure that contains the action handler's callbacks.

Header File

AVCalls.h

Related Methods

[AVAppRegisterActionHandler](#)

Example

```

/* Replace the DoProperties callback of the Thread action handler */
AVActionHandlerProcs actHandler;
AVActionHandlerProcsRec newThreadHandler;
#define Thread_K ASAtomFromString("Thread")

actHandler = AVActionHandlerGetProcs(
    AVAppGetActionHandlerByType(Thread_K));
memcpy(&newThreadHandler, (void*)actHandler,
    sizeof(AVActionHandlerProcsRec));

/* Set latest size (can change from once Acrobat version to another) */
newThreadHandler.size = sizeof(AVActionHandlerProcsRec);
newThreadHandler.DoProperties = ASCallbackCreateProto(
    AVActionDoPropertiesProc, &myDoProperties);
AVAppRegisterActionHandler(
    &newThreadHandler, NULL,
    (char*)ASAtomGetString(Thread_K), userName);

```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVActionHandlerGetType

```
ASAtom AVActionHandlerGetType (AVActionHandler handler);
```

Description

Gets the **ASAtom** specifying what type of actions an action handler services. This is the same as the name used when the action handler was registered using [AVAppRegisterActionHandler](#).

Parameters

handler	The action handler whose type is obtained.

Return Value

The **ASAtom** specifying what action types **handler** services.

Header File

AVCalls.h

Related Methods

[AVAppRegisterActionHandler](#)
[AVActionHandlerGetProcs](#)

Example

```
if (AVActionHandlerGetType (myActionHandler)
    == ASAtomFromString ("GoToR")) {
    /* do something */
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVActionHandlerGetUIName

```
const char* AVActionHandlerGetUIName
(AVActionHandler handler);
```

Description

Gets the string that was passed as the user friendly when the action handler was registered using [AVAppRegisterActionHandler](#).

Parameters

handler	The action handler whose user interface name is obtained.

Return Value

The user interface name of **handler**.

Header File

AVCalls.h

Related Methods

[AVAppRegisterActionHandler](#)

Example

```
if (strcmp (AVActionHandlerGetUIName(theHandler), "BalloonAction")){
    /* do something */
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVAlert

AVAlert

```
ASInt32 AVAlert (ASInt32 iconType, const char* msg,
const char* button1, const char* button2, const char* button3,
ASBool beep) ;
```

Description

Displays an alert containing the specified message, icon, and one to three buttons with the specified titles.

You can replace this method with your own version, using [HFTReplaceEntry](#).

NOTE: It is a limitation of the current implementation that if the button titles do not match either of the following sets, the size of the dialog is fixed and may not display all of the text in `msg`:

- Yes [, No [, Cancel]]
- OK [, Cancel]

NOTE: All the implementations of the AVAlert methods call [AVAlert](#), which is a replaceable method. If [AVAlert](#) is replaced, all of the AVAlert methods are also affected.

Parameters

<code>iconType</code>	The icon to display. Must be one of the AVAlert Icons . Macintosh users: These constants are defined as per the standard Macintosh user interface guidelines.
<code>msg</code>	The message to display.
<code>button1</code> , <code>button2</code> , <code>button3</code>	Titles for up to three buttons. • Use <code>NULL</code> to suppress a button’s display. • At least <code>button1</code> must be non-<code>NULL</code>. • <code>button3</code> is not displayed if <code>button2</code> is <code>NULL</code>.
<code>beep</code>	Pass <code>true</code> to perform a system beep when the alert is shown.

Return Value

The button number (1, 2, or 3) on which the user clicked.

Header File

`AVCalls.h`

**Related Methods**

[AAlertGetPref](#)
[AAlertConfirm](#)
[AAlertNote](#)
[AVDocAlert](#)

Example

```
ASInt32 choice = AAlert(ALERT_CAUTION, "Reverting will discard all  
changes. Are you sure?", "OK", "Cancel", NULL, false);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AAlertConfirm

ASBool AAlertConfirm (const char* msg);

Description

Displays a dialog box containing the **ALERT_CAUTION** icon, the specified message and **OK** and **Cancel** buttons. The method also performs a system beep. See [AAlert](#) for more information.

Parameters

msg	The message to display.

Return Value

true if the user clicks "OK", **false** if the user clicks "Cancel".

Header File

AVCalls.h

Related Methods

[AAlert](#)
[AAlertNote](#)
[AVDocAlertConfirm](#)

Example

```
if(AAlertConfirm("Are you sure you want to delete all Bookmarks?")){
    PDBookmarkDestroy(PDDocGetBookmarkRoot(pdDoc));
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AAlertGetPref

ASInt32 AAlertGetPref (char* name);

Description

Retrieves the value stored under **name** in the **AAlert** preference store. The **AAlert** preference store is intended to be used by clients to store user preferences regarding whether to display an alert prior to execution of a particular operation. The store is persistent across Acrobat sessions. This routine would typically be used when implementing a dialog that contains a check box saying, "Do not show me this dialog again."

Parameters

name	Name of the entry to retrieve. Limited to alphanumeric characters only.

Return Value

The value stored under the **name**, or 0 if the key was not found.

Header File

AVCalls.h

Related Methods

[AAlertSetPref](#)
[AAlertResetPrefs](#)

Example

```
ASInt32 nAnswer = AAlertGetPref("UniqueDialogID");
if (nAnswer != 0) {
    nAnswer = AAlertWithParams();
    AAlertSetPref("UniqueDialogID", nAnswer);
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in PIRquir.h) is set to 0x00050000 or higher.



AAlertNote

```
void AAlertNote (const char* msg);
```

Description

Displays a dialog box containing the **ALERT_NOTE** icon, the specified message and an **OK** button. The method also performs a system beep.

NOTE: The implementation of **AAlertNote** calls **AAlert**, which is a replaceable method. If **AAlert** is replaced, **AAlertNote** is also affected.

Parameters

msg	The message to display.

Return Value

None

Header File

AVCalls.h

Related Methods

[AAlert](#)
[AAlertConfirm](#)

Example

```
DURING

HANDLER
    sprintf(buf, "Exception raised: %d, %s",
        ERRORCODE, ASGetErrorString(ERRORCODE));
    AAlertNote(buf);
END_HANDLER
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AAlertResetPrefs

```
void AAlertResetPrefs (void);
```

Description

Resets the entire AAlert preference store. Specific preference entries can be cleared by passing a value of 0 to [AAlertSetPref](#).

Parameters

None

Return Value

None

Header File

AVCalls.h

Related Methods

[AAlertGetPref](#)

[AAlertSetPref](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AAlertSetPref

```
void AAlertSetPref (char* name, ASInt32 value);
```

Description

Stores a value under **name** in the **AAlert** preference store. The **AAlert** preference store is intended to be used by clients to store user preferences regarding whether an alert is displayed prior to execution of a particular operation. The store is persistent across Acrobat sessions. This routine would typically be used when implementing a dialog that contains a check box saying, "Do not show me this dialog again."

Parameters

name	The name of the entry. Limited to alphanumeric characters only.
value	The value to store; pass 0 (zero) to clear this specific entry.

Return Value

None

Header File

AVCalls.h

Related Methods

[AAlertGetPref](#)
[AAlertResetPrefs](#)

Example

```
ASInt32 nAnswer = AAlertGetPref("UniqueDialogID");
if (nAnswer != 0) {
    nAnswer = AAlertWithParams();
    AAlertSetPref("UniqueDialogID", nAnswer);
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



AVAlertWithParams

```
ASInt32 AVAlertWithParams (AVAlertParams params);
```

Description

Displays an alert dialog with a feature set as described by the supplied **AVAlertParams**.

Parameters

params	A description of the alert feature set.

Return Value

The button number (1,2, or 3) on which the user clicked.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



AVAnnotHandler

AVAnnotHandlerDeleteInfo

```
void AVAnnotHandlerDeleteInfo (AVAnnotHandler annotHandler,
                               AVAnnotHandlerInfo info);
```

Description

Delete the **AVAnnotHandlerInfo** associated with an annotation handler.

Parameters

annotHandler	The annotation handler.
info	The AVAnnotHandlerInfo associated with the annotation handler. info contains the user interface name of annotation type and the platform-dependent bitmap used as the annotation icon.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAnnotHandlerGetInfo](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION**(in `PIRequir.h`) is set to **0x00040000** or higher.

AVAnnotHandlerGetInfo

```
AVAnnotHandlerInfo AVAnnotHandlerGetInfo
(AVAnnotHandler annotHandler) ;
```

Description

Gets the information structure associated with an annotation handler.

Parameters

annotHandler	The annotation handler for which the information structure is needed.

Return Value

The information structure associated with **annotHandler**, or **NULL** if the handler does not support this operation.

Header File

AVCalls.h

Related Methods

[AVAnnotHandlerDeleteInfo](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

AVApp

AVAppAutoShowHowToPanel

```
void AVAppAutoShowHowToPanel (AVDoc avdoc, ASAtom panelName)
```

Description

Opens the HowTo panel and fills it with the specified panel's content, if that panel's AutoShow attribute is **true**.

The Acrobat application does not automatically show any HowTo panel; it is up to a client to determine the conditions under which to show the panel automatically. Clients should allow the user to disable the auto-show behavior; use [AVAppSetHowToPanelAutoShow](#) to set the AutoShow attribute of a panel to **false**.

Parameters

avdoc	The document for which to show the HowTo panel, or NULL to display the application-wide HowTo panel in the main application window. You must specify the document if it is external to the application.
panelName	The name of the HowTo panel to show.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppGetHowToPanelAutoShow](#)
[AVAppSetHowToPanelAutoShow](#)
[AVAppSetHowToPanelAutoShowText](#)
[AVAppSetHowToPanelComputeVisibleProc](#)
[AVAppRegisterHowToPanel](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVAppBeginFullScreen

```
ASBool AVAppBeginFullScreen (PDColorValue color);
```

Description

Begins full-screen mode. In full-screen mode, all window borders, the menubar, and the toolbar are hidden. All regions of the screen outside of the window boundary are painted with the specified color.

AVAppBeginFullScreen is ignored if the application is already in full-screen mode, or if there are no currently open documents.

Parameters

color	(May be NULL) The color to use for painting all regions of the screen outside of the window boundary. Pass NULL to default to the color specified by the application preference avpFullScreenColor .

Return Value

true if the application enters full-screen mode, **false** if it is already in full-screen mode or the user selects **Cancel** from the dialog describing how to exit full-screen mode.

Header File

AVCalls.h

Related Methods

[AVAppEndFullScreen](#)
[AVAppDoingFullScreen](#)

Example

```
PDColorValue color;
if (!AVAppDoingFullScreen())
    AVAppBeginFullScreen(color);
else
    AVAppEndFullScreen();
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVAppBeginModal

```
void AVAppBeginModal (AVWindow window);
```

Description

Prepares the Acrobat viewer to display a modal window. For example, disables floating windows in Windows, where they are not automatically disabled. When you are done with the modal window, call [AVAppEndModal](#). Calling **AVAppBeginModal** does not make your window modal, it only informs the Acrobat viewer that you intend to display a modal window now.

Windows users: The parent of a modal window must be the application's window. If you display a second modal window from within a modal window, you must *not* call **AVAppBeginModal** a second time. Instead, make the first modal window the parent of the second.

Macintosh users: This method is a no-op.

UNIX users: This method is a no-op.

Parameters

window	The modal window to display.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppEndModal](#)
[AVAppModalWindowIsOpen](#)
[WinAppGetModalParent](#)

Example

```
AVWindow win = AVWindowNew();
if(!AVAppModalWindowIsOpen()){
    AVAppBeginModal(win);/* disable floating windows */
    AVAppShowWindow(win);
    AVAppEndModal(win);
}
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVAppCanQuit

```
ASBool AVAppCanQuit (void);
```

Description

The Acrobat viewer calls this method to obtain permission (programmatically, not using the user interface) when it wants to quit. To use this method, replace it with your own version, using [HFTReplaceEntry](#).

If you replace this method in UNIX, your replacement implementation must not have any user interface component (for example, dialog boxes). At the time your replacement is called, the Acrobat viewer owns the pointer and will not give it to your dialog, so the screen will freeze.

Parameters

None

Return Value

Returns **true** if Acrobat can quit, **false** if it may not. The default version of this routine always returns **true**.

Header File

AVCalls.h

Related Methods

[HFTReplaceEntry](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVAppChooseFolderDialog

```
ASBool AVAppChooseFolderDialog
(AVOpenSaveDialogParams dialogParams, ASFileSys* outFileSys,
ASPathName *outASPathName) ;
```

Description

Displays a platform dependent folder selection dialog. **fileSys** and **pathName** will be **NULL** if the user cancelled the operation.

It is the caller's responsibility to release the returned **ASPathName**.

Parameters

dialogParams	Standard dialog parameters for the Open/Save/ChooseFolder dialogs.
outFileSys	<i>(Filled by the method, may be NULL)</i> The file system through which pathName was obtained. May be NULL if kAVOpenSaveAllowForeignFileSystems is not passed in dialogParams .
outASPathName	<i>(Filled by the method)</i> The ASPathName associated with the folder chosen by the user.

Return Value

Returns **true** if the user confirmed the selection, **false** if user clicked cancel or some error occurred.

Header File

AVCalls.h

Related Methods

```
AVAppOpenDialog
AVAppSaveDialog
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVAppCreateIconBundle6

```
AVIconBundle6 AVAppCreateIconBundle6
(AVIconDataFormat eDataFormat, AVIconDataRec *dataRecs,
ASCount numRecs)
```

Description

Creates an icon bundle object from an array of icon data records.

Parameters

eDataFormat	The data format for the icons in the new icon bundle.
dataRecs	An array of icon data records for the new icon bundle.
numRecs	The number of records in dataRecs.

Return Value

The new icon bundle object.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVAppDoingFullScreen

ASBool AVAppDoingFullScreen (void);

Description

Tests whether the application is running in full-screen mode.

Parameters

None

Return Value

true if application is currently in full-screen mode, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVAppBeginFullScreen](#)

[AVAppEndFullScreen](#)

Example

```
PDColorValue color;
if (!AVAppDoingFullScreen())
    AVAppBeginFullScreen(color);
else
    AVAppEndFullScreen();
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppEndFullScreen

```
void AVAppEndFullScreen (void);
```

Description

Ends full-screen mode. Does nothing if the application is not running in full-screen mode.

Parameters

None

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppBeginFullScreen](#)

[AVAppDoingFullScreen](#)

Example

```
PDColorValue color;
if (!AVAppDoingFullScreen())
    AVAppBeginFullScreen(color);
else
    AVAppEndFullScreen();
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVAppEndModal

```
void AVAppEndModal (void);
```

Description

(Windows only) Informs the Acrobat viewer that a modal window is no longer being displayed.

Parameters

None

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppBeginModal](#)
[AVAppModalWindowIsOpen](#)
[WinAppGetModalParent](#)

Example

```
AVWindow win = AVWindowNew();  
if(!AVAppModalWindowIsOpen()) {  
    AVAppBeginModal(win); /* disable floating windows */  
    AVAppShowWindow(win);  
    AVAppEndModal(win);  
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppEnumActionHandlers

```
void AVAppEnumActionHandlers (AVActionEnumProc enumProc,
void* clientData);
```

Description

Enumerates all registered action handlers, calling the user-supplied procedure for each.

Parameters

enumProc	User-supplied procedure to call once for each action handler.
clientData	Pointer to user-supplied data to pass to enumProc each time it is called.

Return Value

None

Known Exceptions

Raises an exception if and only if **enumProc** raises an exception.

Header File

AVCalls.h

Related Methods

[AVActionHandlerGetProcs](#)
[AVAppRegisterActionHandler](#)

Example

```
#define Thread_K ASAtomFromString("Thread")
static ACCB1 ASBool ACCB2
myActionEnumerator(ASAtom type, char* userName, void* clientData)
{
    if(type == Thread_K){
        actHandler = AVActionHandlerGetProcs(
            AVAppGetActionHandlerByType(Thread_K));
        origDoProperties = actHandler->DoProperties;
        return false; /* leave once we've got what we need */
    }
    return true;
}
AVAppEnumActionHandlers(
    ASCallbackCreateProto(AVActionEnumProc,
        &myActionEnumerator), NULL);
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppEnumAnnotHandlers

```
void AVAppEnumAnnotHandlers (AVAnnotHandlerEnumProc enumProc,
void* clientData);
```

Description

Enumerates all registered annotation handlers, calling the user-supplied procedure for each.

Parameters

enumProc	User-supplied callback to call for each annotation handler.
clientData	Pointer to user-supplied data to pass to enumProc each time it is called.

Return Value

None

Known Exceptions

Raises an exception if and only if **enumProc** raises an exception.

Header File

AVCalls.h

Related Methods

[AVAppGetAnnotHandlerByName](#)
[AVAppRegisterAnnotHandler](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppEnumDocs

```
void AVAppEnumDocs (AVDocEnumProc enumProc, void* clientData);
```

Description

Enumerates all [AVDocs](#) currently open in the viewer, calling the user-supplied procedure for each.

NOTE: Note that **enumProc** must not close any documents.

Parameters

enumProc	User-supplied callback to call once for each open AVDoc .
clientData	Pointer to user-supplied data to pass to enumProc each time it is called.

Return Value

None

Known Exceptions

Raises an exception if and only if **enumProc** raises an exception.

Header File

AVCalls.h

Related Methods

[PDEnumDocs](#)

Example

```
ACCB1 ASBool ACCB2 myAppEnumProc (AVDoc doc,
    void* clientData) {
    if (PDDocGetNumPages (AVDocGetPDDoc (doc))
        > 100)
        AVAlertNote ("Long document");
    return true;
}
...
AVAppEnumDocs (ASCallbackCreateProto (AVDocEnumProc, &myAppEnumProc),
    NULL);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVAppEnumTools

```
void AVAppEnumTools (AVToolEnumProc enumProc,
void* clientData);
```

Description

Enumerates all registered tools, calling the user-supplied procedure for each.

Parameters

enumProc	User-supplied callback to call for each tool.
clientData	Pointer to user-supplied data to pass to enumProc each time it is called.

Return Value

None

Known Exceptions

Raises an exception if and only if **enumProc** raises an exception.

Header File

AVCalls.h

Related Methods

[AVAppGetToolByName](#)

[AVAppRegisterTool](#)

Example

```
ACCB1 ASBool ACCB2 myToolEnumerator(AVTool tool, void* clientData)
{
    return false;
}
AVAppEnumTools (ASCallbackCreateProto (AVToolEnumProc, &myToolEnumerator),
NULL);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVAppEnumTransHandlers

```
void AVAppEnumTransHandlers (AVTransHandlerEnumProc enumProc,
void* clientData);
```

Description

Enumerates all registered transition handlers, calling the user-supplied procedure for each.

Parameters

enumProc	User-supplied procedure to call once for each transition handler.
clientData	User-supplied data passed to enumProc each time it is called.

Return Value

None

Known Exceptions

Raises an exception if and only if **enumProc** raises an exception.

Header File

AVCalls.h

Related Methods

[AVAppGetTransHandlerByType](#)
[AVAppRegisterTransHandler](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

AVAppFindCommandHandlerByName

```
AVCommandHandler AVAppFindCommandHandlerByName (ASAtom name);
```

Description

Gets the **AVCommandHandler** that was registered under the given name.

Parameters

name	The name of the handler to obtain.

Return Value

The **AVCommandHandler** if it was found, **NULL** otherwise.

Header File

AVCalls.h

Related Methods

[AVAppRegisterCommandHandler](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVAppFindGlobalCommandByName

```
AVCommand AVAppFindGlobalCommandByName (ASAtom name);
```

Description

Returns the **AVCommand** that was registered under the given name.

Parameters

name	The name of the AVCommand to obtain.

Return Value

The **AVCommand** if one was found, **NULL** otherwise.

Header File

AVCalls.h

Related Methods

[AVAppRegisterGlobalCommand](#)
[AVAppUnregisterGlobalCommand](#)
[AVCommandNew](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVAppGetActionHandlerByType

[AVActionHandler](#) AVAppGetActionHandlerByType ([ASAtom](#) type) ;

Description

Gets the action handler that services the specified action type .

Parameters

type	The action type whose handler is obtained.

Return Value

The handler that services actions of the specified type. Returns **NULL** if no such handler is registered.

Header File

AVCalls.h

Related Methods

[AVAppRegisterActionHandler](#)
[AVActionHandlerGetType](#)
[AVActionHandlerGetUIName](#)

Example

```
#define Thread_K ASAtomFromString("Thread")
actHandler = AVActionHandlerGetProcs(
    AVAppGetActionHandlerByType(Thread_K) );
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVAppGetActiveDoc

AVDoc AVAppGetActiveDoc (void);

Description

Gets the frontmost document window. In UNIX, gets the **AVDoc** being viewed within the window that got the last user event (Key or Button event). This method is often useful for menu or tool enable procs. The frontmost document may not be active—for example, the clipboard window may be active and over it.

This method returns documents that are being viewed within the host application. In some cases it can return a document open in an external window (for example, a web browser), but you cannot count on this.

The safest approach is to avoid calling **AVAppGetActiveDoc** if at all possible. You will usually get good results if **AVAppGetActiveDoc** is called only from within menu item or toolbar button callbacks (**AVExecuteProc**, **AVComputeEnabledProc**, and **AVComputeMarkedProc**), but in general it is best to use some bit of context already provided to you to figure out which **AVDoc** the user is interacting with. Use the following guidelines:

- If you have a **PDDoc** in hand (or can get one from some other PD object) call **AVDocFromPDDoc** to find the **AVDoc** open on the **PDDoc**.
- If you have an **AVPageView** in hand, call **AVPageViewGetAVDoc** to retrieve the **AVDoc**.
- If you have an **AVPanel** in hand, call **AVPanelGetAVDoc** to retrieve the **AVDoc** that the panel points to. You probably should also provide a DocChanged callback in your panel handler so you'll be notified when your panel's document changes.
- If you have an **AVCommand** in hand and are absolutely sure you want an **AVDoc**, use **AVCommandGetPDDoc** and pass the result to **AVDocFromPDDoc**.

It's also a good idea to localize the determination of the active doc. Occasionally client code calls **AVAppGetActiveDoc** often and at various levels. It's more robust to determine the active document once and then pass it along as an argument to your other routines.

Also, make absolutely sure you want an **AVDoc** and not a **PDDoc** in the first place. If your code can work on a **PDDoc** (for example, it does not require any UI) then it should.

NOTE: **AVAppGetActiveDoc** is often used to enable a menu item in a client (**if AVAppGetActiveDoc() != NULL**).

Parameters

None

Return Value

The frontmost document window, or **NULL** if no documents are open. **NULL** is also returned when a document is open but its invisible flag is set (see **AVDocOpenParams**) and may be returned while a document is being opened.



Notifications

None. You can get an **AVDoc** as it opens by registering for the [AVDocDidOpen](#), [AVDocWillOpenFromFile](#), or [AVDocWillOpenFromPDDoc](#) notifications.

Header File

AVCalls.h

Related Methods

[AVAppEnumDocs](#)

[AVAppGetNumDocs](#)

Example

```
/* Can be used as menu item enable proc */
static ACCB1 ASBool ACCB2
    DocExistEnable(void* data){
    return (AVAppGetActiveDoc() != NULL);
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppGetActiveTool

[AVTool](#) AVAppGetActiveTool (void);

Description

Gets the active tool for the application.

NOTE: It is recommended that you use [AVDocGetActiveTool](#) instead, preferably specifying a particular document.

Parameters

None

Return Value

The active tool.

Header File

AVCalls.h

Related Methods

[AVAppSetActiveTool](#)
[AVAppGetLastActiveTool](#)
[AVAppGetDefaultTool](#)
[AVDocGetActiveTool](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppGetAnnotHandlerByName

[AVAnnotHandler](#) AVAppGetAnnotHandlerByName ([ASAtom](#) name) ;

Description

Gets the annotation handler that handles the specified annotation type.

Parameters

name	Name of the requested annotation handler.

Return Value

The annotation handler that services annotations of type **name**. If no annotation handler is registered for that type, the viewer's default annotation handler is returned. To determine whether the returned annotation handler is the default annotation handler, check the return value from the handler's **GetType** for the type **Unknown**.

Header File

AVCalls.h

Related Methods

[AVAppEnumAnnotHandlers](#)
[AVAppGetAnnotHandlerByName](#)

Example

```
AVAnnotHandler theHandler = AVAppGetAnnotHandlerByName(asaType);
if (ASAtomFromString("unknown") == theHandler->GetType(theHandler))
    AAlertNote("This is the viewer's default annothandler");
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVAppGetCancelProc

CancelProc AVAppGetCancelProc (void** procClientDataP);

Description

Gets the default application cancel procedure. The procedure returns true if the users has recently entered the keystroke described below.

A cancel procedure is often passed to methods that take a long time to run. The method will call the procedure at frequent intervals, and if the return value is **true**, the method cancels its operation.

The keystroke that the application recognizes as a cancel command is platform-dependent.

- In Mac OS, an operation is canceled by a Command-period combination.
- In Windows, an operation is canceled by the Escape key.

Parameters

cancelProcClientDataP	(Allocated and filled by the method) Data needed by an CancelProc . This value must be passed as the clientData whenever the procedure is called or passed to a method which may call it.

Return Value

The default **CancelProc**, or **NULL** if one is not available.

Header File

AVCalls.h

Example

```
CancelProc cm;
void* cmData;
cm = AVAppGetCancelProc(&cmData);
PDDocInsertPages(AVDocGetPDDoc(doc), afterPage, pdDoc, 0, PDAllPages,
NULL, pm, pmData,
cm, cmData);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVAppGetDefaultTool

[AVTool](#) AVAppGetDefaultTool (void);

Description

Gets the default tool. Use this method, together with [AVAppSetActiveTool](#), to restore the default tool any time you want. The default tool is the hand tool.

Parameters

None

Return Value

The default tool. Returns **NULL** if there is no default tool.

Header File

AVCalls.h

Related Methods

[AVAppSetActiveTool](#)
[AVAppGetActiveTool](#)
[AVAppGetLastActiveTool](#)

Example

```
AVAppSetActiveTool (AVAppGetDefaultTool (), true);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppGetDocProgressMonitor

```
ASProgressMonitor AVAppGetDocProgressMonitor  
(void** progMonClientData);
```

Description

Gets the standard application progress monitor, which puts combined status window/progress bar in the message pane of the frontmost document window. This progress monitor can subsequently be passed to any API method requiring a progress monitor.

If you want to display and control a progress monitor in your own client, you can simply invoke the appropriate callbacks in the progress monitor data structure this method returns.

(New in Acrobat 5.0) New **setText** value in **ASProgressMonitor** allows adding text to a monitor.

Parameters

progMonClientData	(Allocated and filled by the method, may not be NULL) Private data used by the progress monitor. This value must be passed as the clientData whenever a callback in the structure is called, or the monitor is passed to a method that takes a progress monitor as a parameter.

Return Value

The standard application progress monitor, or **NULL** if no documents are open.

Header File

AVCalls.h

Example

```

static ProgressMonitor progMon;
static void* monClientData;
progMon = AVAppGetDocProgressMonitor(&monClientData);
if (progMon) {
    if (progMon->beginOperation)
        progMon->beginOperation(monClientData);
    if (progMon->setDuration)
        progMon->setDuration(len, monClientData);
    if (progMon->setCurrValue)
        progMon->setCurrValue(0, monClientData);
}
/* do long operation */
for(i=0;i<numBookmarks;i++){
    /* do the work */
    if (progMon->setCurrValue)
        progMon->setCurrValue(i, monClientData);
}
if (progMon->beginOperation)
    progMon->beginOperation(monClientData);

```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVAppGetHowToPanelAutoShow

ASBool AVAppGetHowToPanelAutoShow(**ASAtom** panelName) ;

Description

Get the current auto-show state of a HowTo panel. The auto-show state is **true** by default. When the auto-show state is **true** for a specified panel, that panel is shown whenever the client calls **AVAppAutoShowHowToPanel** on it.

Parameters

panelName	The internal language-independent name of the HowTo panel whose auto-show state is obtained.

Return Value

true if the current auto-show state is ON, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVAppAutoShowHowToPanel](#)
[AVAppSetHowToPanelAutoShow](#)
[AVAppSetHowToPanelAutoShowText](#)
[AVAppSetHowToPanelComputeVisibleProc](#)
[AVAppRegisterHowToPanel](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVAppGetLanguage

```
void AVAppGetLanguage (char* buffer);
```

Description

Gets the language in which the application's user interface is running. See the list of [Language Codes](#) for the possible return values.

NOTE: Superseded in Acrobat 6.0 by [AVAppGetLanguageWithParams](#).

Parameters

buffer	<i>(Filled by the method)</i> The language code. buffer must be able to contain four bytes.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppGetLanguageWithParams](#)

[AVAppGetVersion](#)

Example

```
char lang[4];
AVAppGetLanguage(lang);
if(strcmp("ENU", lang)){
    /* handle non-English language in special way */
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppGetLanguageEncoding

[ASHostEncoding](#) AVAppGetLanguageEncoding (void);

Description

Returns the **ASHostEncoding** corresponding to Acrobat's current locale setting. For example, if the user is using the English version of Acrobat on a Japanese system, **AVAppGetLanguageEncoding** returns a Roman encoding but [PDGetHostEncoding](#) returns a Japanese encoding.

Parameters

None

Return Value

The encoding constant for the application.

Header File

AVCalls.h

Related Methods

[PDGetHostEncoding](#)
[PDGetPDFDocEncoding](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



AVAppGetLanguageWithParams

```
ASBool AVAppGetLanguageWithParams
(AVAppLanguageParams params);
```

Description

Retrieves the language in which the application's user interface is running, in the format specified by the **kLangFormat** member of the supplied parameter structure.

NOTE: Supersedes [AVAppGetLanguage](#) in Acrobat 6.0.

Parameters

params	(Filled by the method) On input, contains the desired language format. On return, contains the language string in the given format.

Return Value

true if the language string was retrieved, otherwise **false**.

Header File

AVCalls.h

Related Methods

[AVAppGetLanguage](#)
[AVAppGetVersion](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVAppGetLastActiveTool

```
AVTool AVAppGetLastActiveTool (void);
```

Description

Gets the tool that was active before the current tool became active.

Parameters

None

Return Value

The last active tool. If only one tool has ever been active, it is returned.

Header File

AVCalls.h

Related Methods

[AVAppGetActiveTool](#)
[AVAppGetDefaultTool](#)

Example

```
if (!toolPersistent)
    AVAppSetActiveTool(
        AVAppGetLastActiveTool(), true);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVAppGetMenubar

[AVMenubar](#) AVAppGetMenubar (void);

Description

Gets Acrobat's menubar.

Parameters

None

Return Value

The menubar.

Header File

AVCalls.h

Related Methods

[AVMenuGetParentMenubar](#)

Example

```
AVMenubar bar = AVAppGetMenubar();
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVAppGetName

```
ASAtom AVAppGetName (void);
```

Description

Gets the **ASAtom** corresponding to the application's name—that is, the name of the file containing the Acrobat viewer application. The user might have changed this, so do not use it to determine what the application is; use [ASGetConfiguration](#) instead.

Parameters

None

Return Value

An **ASAtom** representing the Acrobat viewer's name.

Header File

AVCalls.h

Related Methods

[ASGetConfiguration](#)

Example

```
char buf[200];
const char* appName;
ASAtom appNameAtom;

appNameAtom = AVAppGetName();
if (appNameAtom != ASAtomNull) {
    appName = ASAtomGetString(appNameAtom);
    sprintf(buf, "Application name is %s",
            appName);
    AAlertNote(buf);
}
else /* This should >never< happen */
    AAlertNote("Application name NULL!");
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVAppGetNumDocs

[AVTArraySize](#) AVAppGetNumDocs (void);

Description

Gets the number of open document views.

Parameters

None

Return Value

The number of open [AVDocs](#).

Header File

AVCalls.h

Related Methods

[AVAppGetActiveDoc](#)

Example

```
/* Might be used as a menu item's enable
   proc */
static ACCB1 ASBool ACCB2 DocExistEnable(void* data)
{
    if (AVAppGetNumDocs() <= 0)
        return false;
    else
        return true;
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppGetPreference

```
void* AVAppGetPreference (AVPrefsType preference);
```

Description

Gets the value of the specified built-in application preference.

Parameters

preference	The preference value to get. See the preference descriptions in AVPrefsType .

Return Value

NULL if preference was not recognized. Otherwise, clients must cast the return value to the appropriate type, depending on the preference requested. See the preference descriptions in [AVPrefsType](#).

Header File

AVCalls.h

Related Methods

[AVAppSetPreference](#)

Example

```
if (AVAppGetPreference (avpSkipWarnings) ||
    AAlertConfirm ("Really?")) {
    AVAppSetPreference (avpShowToolBar,
        (void* )true);
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppGetReportProc

[ASReportProc](#) AVAppGetReportProc
(void** asReportProcClientDataP)

Description

Retrieves a standard report proc that can be used to report errors and warnings to the user. See [ASReportProc](#) for more details.

Parameters

asReportProcClientDataP	Pointer to client data passed to the report proc.

Return Value

The report callback.

Header File

AVCalls.h

Related Methods

[AVCommandGetReportProc](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



AVAppGetToolBar

[AVToolBar](#) AVAppGetToolBar (void);

Description

Returns Acrobat's Editing toolbar.

NOTE: Developers should use [AVToolBarNew](#) to create their own toolbars. The Editing toolbar is disabled in external document views (see [Toolbar and Toolbar Button Names](#)).

Parameters

None

Return Value

The toolbar.

Header File

AVCalls.h

Related Methods

[AVAppGetToolBarByName](#)
[AVToolBarNew](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppGetToolBarByName

AVToolBar AVAppGetToolBarByName (const char* name);

Description

Returns the toolbar created with the specified name. Refer to [Toolbar and Toolbar Button Names](#) for a list of the standard named toolbars.

NOTE: It is recommended that you position toolbar buttons by finding a related button on the main toolbar (the one returned by [AVAppGetToolBar](#)) and placing your button next to the existing button. Acrobat will automatically place your toolbar button on the correct named toolbar so it appears next to the existing button.

Parameters

name	The name of the toolbar.

Return Value

The **AVToolBar**, or **NULL** if no **AVToolBar** was found with the specified name.

Header File

AVCalls.h

Related Methods

[AVAppGetToolBar](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



AVAppGetToolByName

```
AVTool AVAppGetToolByName (ASAtom name);
```

Description

Returns the **AVTool** that was registered under the specified name.

Parameters

name	The ASAtom corresponding to the tool's name. See Tool Names for a list of the names of the built-in tools.

Return Value

The tool that was registered under **name**, or **NULL** if no match was found.

Header File

AVCalls.h

Related Methods

[AVAppGetActiveTool](#)

Example

```
AVTool noteTool = AVAppGetToolByName (
    ASAtomFromString("Note"));
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVAppGetTransHandlerByType

[AVTransHandler](#) AVAppGetTransHandlerByType ([ASAtom](#) name) ;

Description

Gets the transition handler registered for the specified transition type.

Parameters

name	Type of transition handler, which may be one of the types provided in the Acrobat viewer or a new type registered by a plug-in.

Return Value

The transition handler for the type, or **NULL** if no transition handler is registered for that type.

Header File

AVCalls.h

Related Methods

[AVAppRegisterTransHandler](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.



AVAppGetUUID

```
ASBool AVAppGetUUID (AVAppUUIDType type, ASUUID *dst);
```

Description

Gets a unique identifier (UUID) for the current user or the current session. The UUID can be used with P2P, Web interactions, and so on.

Parameters

type	<i>(Filled by the method)</i> The UUID type value, which specifies whether it identifies the current user or the current session.
dst	<i>(Filled by the method)</i> Pointer to the UUID object.

Return Value

true if the UUID is successfully retrieved, **false** otherwise.

Header File

AVCalls.h

Related Methods

[ASUUIDFromCString](#)
[ASUUIDGenFromHash](#)
[ASUUIDGenFromName](#)
[ASUUIDGenUnique](#)
[ASUUIDToCString](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVAppGetVersion

```
void AVAppGetVersion (AVTVersionNumPart* major,  
AVTVersionNumPart* minor);
```

Description

Gets the major and minor version numbers of the Acrobat viewer.

To correctly accommodate cases such as 4.05 and 4.5, the minor version is split into two 8 bit numbers for example:

4.05 - minor version would be 0x0005

4.5 - minor version would be 0x0500

Parameters

major	(Filled by the method) Pointer to the major version number.
minor	(Filled by the method) Pointer to the minor version numbers.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppGetLanguage](#)

Example

```
AVTVersionNumPart major, minor;  
AVAppGetVersion(&major, &minor);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppHandlePlatformEvent

[ASBool](#) AVAppHandlePlatformEvent (void* platformEvent);

Description

Handles a platform-specific event.

Parameters

platformEvent	A pointer to a platform-specific event structure.

Return Value

true if the event was handled, **false** otherwise.

Known Exceptions

May raise exceptions, depending on the event.

Header File

AVCalls.h

Related Methods

[AVAppHandleAppleEvent](#)

[AVWindowHandlePlatformEvent](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.



AVAppHelpSearch

```
ASBool AVAppHelpSearch (const char *fileName,
                        const char *searchText);
```

Description

Opens the specified PDF file (if it is not already open) and searches for a specified string. If the help file is not found, opens a dialog box asking if the user wants to search for the help file. If the help file is found, the method opens it with the navigation panel on the left showing the search panel, which displays the result of the search.

Parameters

fileName	The name of the PDF file in which to search. If it is a simple file name, it is assumed to reside in the help contents folder. Pass NULL for the default Acrobat help file.
searchText	The UTF8-encoded text for which to search.

Return Value

true if the help file was successfully opened, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVAppHelpShowContents](#)
[AVAppHelpShowIndex](#)
[AVAppOpenHelpFileWithParams](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVAppHelpShowContents

```
ASBool AVAppHelpShowContents (const char *fileName,
const char *contentTag);
```

Description

Opens the specified PDF help file and goes to the location of the specified content. If the help file is not found, opens a dialog box asking if the user wants to search for the help file.

Parameters

fileName	The name of the PDF help file to open. If it is a simple file name, it is assumed to reside in the help contents folder. Pass NULL for the default Acrobat help file.
contentTag	A named destination defined in the PDF help file.

Return Value

true if the PDF was opened successfully, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVAppHelpSearch](#)
[AVAppHelpShowIndex](#)
[AVAppOpenHelpFileWithParams](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVAppHelpShowIndex

ASBool AVAppHelpShowIndex (const char *fileName);

Description

Opens the specified PDF help file and displays the index. If the help file is not found, opens a dialog box asking if the user wants to search for the help file.

Parameters

fileName	The name of the PDF help file to open. If it is a simple file name, it is assumed to reside in the help contents folder. Pass NULL for the default Acrobat help file.

Return Value

true if the PDF was opened successfully, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVAppHelpSearch](#)

[AVAppHelpShowContents](#)

[AVAppOpenHelpFileWithParams](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVAppIsIdle

```
ASBool AVAppIsIdle (void);
```

Description

Returns **true** if the application is in a state in which it is safe to destroy a document that uses the multi-read protocol. The multi-read protocol is implemented on some platforms using a “yield” mechanism, which involves a transfer of information between two applications. This function returns **true** if there is no task executing in the application that could invoke such a transfer. Attempting to close a document during such a transfer can cause a deadlock event or a crash.

When a multi-read protocol document is closed, the client must register an **AVAppIdle** task. During its idle proc, it must call **AVAppHandlePlatformEvent**. If the call returns **true**, it is safe to call **AVDocClose**. If it returns **false**, the client must retry at its next idle proc call. **AVAppHandlePlatformEvent** always returns **false** if invoked from an **AVExecuteProc**.

This method does *not* protect against clients that invoke the API outside of **AVAppIdle**, an **AVExecuteProc**, or other explicit API methods. For example, if a Windows client uses a Windows SetTimer event to call an API method, **AVAppHandlePlatformEvent** may erroneously return **true**. Therefore, clients should always use the API methods.

Parameters

None

Return Value

true if it is safe to call **AVDocClose**, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVDocClose](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.



AVAppModalWindowsOpen

ASBool AVAppModalWindowIsOpen (void);

Description

A client should use this method to determine whether a modal window is open. There is a large (and ill-defined) group of actions that are illegal while a modal window is open, although these actions are not programmatically prevented by the Acrobat viewer. While a modal dialog is open, a client must not open documents, change pages, change views, close documents, change tools, or do anything that might disrupt the user or Acrobat viewer.

Parameters

None

Return Value

true if a modal window is open, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVAppBeginModal](#)

[AVAppEndModal](#)

[WinAppGetModalParent](#)

Example

```
AVWindow win = AVWindowNew();
if (!AVAppModalWindowIsOpen()) {
    AVAppBeginModal(win); /* disable floating windows */
    AVAppShowWindow(win);
    AVAppEndModal(win);
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppOpenDialog

```
ASBool AVAppOpenDialog (AVOpenSaveDialogParams dialogParams,
ASFileSys* outFileSys, ASPathName** outASPathNames,
AVArraySize* outNumPathNames,
AVFilterIndex* ioChosenFilterIndex);
```

Description

Displays a platform-dependent file open dialog. It is the caller's responsibility to release the returned **ASPathNames** and the associated memory.

Parameters

dialogParams	Standard parameters for Open/Save/ChooseFolder dialogs.
outFileSys	<i>(Filled by the method)</i> The file system through which the contents of pathNames were opened. May be NULL if kAVOpenSaveAllowForeignFileSystems is not passed in dialogParams .
outPathNames	<i>(Allocated and filled by the method)</i> The ASPathName (s) associated with the file(s) selected by the user. The caller must free the list of filled-in ASPathNames using the supplied ASFileSys .
outNumPathNames	<i>(Filled by the method, may be NULL)</i> The number of ASPathNames in pathNames array. May be NULL if kAVOpenSaveAllowMultiple is not set.
ioChosenFilterIndex	<i>(Filled by the method, may be NULL)</i> The index of the filter chosen by the user to select the file(s). May be NULL if caller does not care which filter was chosen, or -1 for "All Files".

Return Value

Returns **true** if user clicked **Action** to confirm the selection, **false** if user clicked **Cancel** or some error occurred.

Header File

AVCalls.h

Related Methods

[AVAppChooseFolderDialog](#)
[AVAppSaveDialog](#)

**Product**

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

NOTE: The numeric parameter types changed in **0x00060000**.



AVAppOpenHelpFile

```
ASBool AVAppOpenHelpFile (const char* fileName,
ASBool doSearch);
```

Description

Opens a specified help file from the default installation “help” directory. If the help file is not found, optionally opens a dialog box asking if the user wants to search for the help file.

NOTE: Deprecated in Acrobat 6.0. It has been superseded by [AVAppOpenHelpFileWithParams](#).

Parameters

fileName	Help filename. This is not a fully-qualified pathname, but the name of an individual help file, such as “AcroHelp.pdf”.
doSearch	If true and the help file is not found, displays a dialog box asking if the user wants to search for the help file. If false , returns false if the help file is not found.

Return Value

true if the help file is found, **false** otherwise.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

AVAppOpenHelpFileWithParams

```
ASBool AVAppOpenHelpFileWithParams (const char* fileName,
ASBool doSearch, AVDocOpenParams params);
```

Description

Opens a specified help PDF file, using the specified parameters to control the window's size, location, and visibility. If the help file is not found, optionally opens a dialog box asking if the user wants to search for the help file.

NOTE: Supersedes [AVAppOpenHelpFile](#), which is deprecated in Acrobat 6.0.

Parameters

fileName	Help PDF file name. If it is a simple file name, it is assumed to reside in the help contents folder.
doSearch	If true and the help file is not found, displays a dialog box asking if the user wants to search for the help file. If false , returns false if the help file is not found.
params	Open parameters for the document.

Return Value

true if the help file is found, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVAppHelpSearch](#)
[AVAppHelpShowContents](#)
[AVAppHelpShowIndex](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVAppRegisterActionHandler

```
void AVAppRegisterActionHandler
(AVActionHandlerProcs actionHandler, void* actionHandlerObj,
char* pdfName, char* userName);
```

Description

Registers an action handler within Acrobat.

Parameters

actionHandler	A structure containing the callbacks for the action handler to register. This structure must <i>not</i> be freed after calling AVAppRegisterActionHandler .
actionHandlerObj	Pointer to user-supplied data to pass to the action handler's methods when they are invoked.
pdfName	The action type serviced by this handler, as it appears in the PDF file. Storage for this string may be released after the call. This string must not contain any white space characters (for example, spaces).
userName	The name of this action type as it should appear in the Acrobat viewer's user interface. Storage for this string may be released after the call.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVActionHandlerGetProcs](#)

[AVAppEnumActionHandlers](#)

[AVAppGetActionHandlerByType](#)

Example

```
AVActionHandlerProcs actHandler;
char userName[48];

actHandler = ASMalloc(sizeof(AVActionHandlerProcsRec));
if(!actHandler) return;
actHandler->size = sizeof( AVActionHandlerProcsRec);
/* vital for future compatibility */
actHandler->Perform = ASCallbackCreateProto(
    AVActionPerformProc, &myPerform);
...
AVAppRegisterActionHandler(actHandler, NULL, "MultiAction",
    "Multiple Destination Link");
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppRegisterAnnotHandler

```
void AVAppRegisterAnnotHandler (AVAnnotHandler handler);
```

Description

Registers a handler for an annotation subtype, replacing any previous handler that had been registered for that subtype. The annotation handler is not registered if its [AVAnnotHandlerGetTypeProc](#) returns **NULL**.

Parameters

handler	Pointer to a structure containing the annotation handler's callbacks. This structure must <i>not</i> be freed after this call, but must be retained.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppEnumAnnotHandlers](#)
[AVAppGetAnnotHandlerByName](#)
[AVPageViewIsAnnotAtPoint](#)

Example

```
static AVAnnotHandlerRec myAH;

memset(&myAH, 0, sizeof(myAH));
myAH.size = sizeof(myAH); /* Vital for future compatibility */
myAH.flags = ANNOT_CLIP_TEXT_SELECTION;
myAH.GetType = ASCallbackCreateProto
    (AVAnnotHandlerGetTypeProc, &myGetType);
myAH.Draw = ASCallbackCreateProto
    (AVAnnotHandlerDrawProc, &myDraw);
myAH.GetAnnotViewBBox = ASCallbackCreateProto
    (AVAnnotHandlerGetAnnotViewBBoxProc, &myGetViewBBox);
myAH.AdjustCursor = ASCallbackCreateProto
    (AVAnnotHandlerAdjustCursorProc, &myAdjustCursor);
...
AVAppRegisterAnnotHandler (&myAH);
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVAppRegisterCommandHandler

```
void AVAppRegisterCommandHandler (ASAtom name,
AVCommandHandler handler);
```

Description

Registers an **AVCommandHandler** as the implementor of an **AVCommand** with the specified name. If there are any existing commands registered under **name**, the new command replaces them.

A single handler can implement and register multiple commands.

Parameters

name	The command name.
handler	The handler to register.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppFindCommandHandlerByName](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVAppRegisterForContextMenuAddition

```
void AVAppRegisterForContextMenuAddition (ASAtom menuName,
AVContextMenuAdditionProc proc, void *clientData)
```

Description

Registers a user-supplied procedure to call after a context menu has been created but before it is shown to the user. The callback can add menu items to or remove menu items from the menu.

Parameters

menuName	The name of the context menu to modify. One of: ● Page:The standard context menu for an AVPageView. ● Select:The context menu for selected text. For both menus, the menu data is an AVDoc pointer.
proc	User-supplied procedure to call.
clientData	Pointer to user-supplied data to pass to the procedure each time it is called.

Return Value

None

Known Exceptions

genErrNoMemory

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if PI_ACROVIEW_VERSION (in PIRequir.h) is set to 0x00060000 or higher.

AVAppRegisterForPageViewAdjustCursor

```
void AVAppRegisterForPageViewAdjustCursor
(AVPageViewCursorProc cursorProc, void* data);
```

Description

Registers a user-supplied procedure to call each time the cursor is moved. If more than one procedure is registered, the procedures are called in the order that they were registered.

To un-register, you *must* use the same callback that was used to register; you cannot use a newly created callback. To accomplish this, call [ASCallbackCreateProto](#) once before registering and use the value returned from this call both to register and un-register; do not call [ASCallbackCreateProto](#) a second time when un-registering.

Parameters

cursorProc	User-supplied callback to call each time the cursor is moved.
data	Pointer to user-supplied data to pass to cursorProc each time it is called.

Return Value

None

Known Exceptions

[genErrNoMemory](#)

Header File

AVCalls.h

Related Methods

[AVAppUnregisterForPageViewAdjustCursor](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

NOTE: Numeric types in the callback have changed in Acrobat 6.0.

AVAppRegisterForPageViewClicks

```
void AVAppRegisterForPageViewClicks
(AVPageViewClickProc clickProc, void* data);
```

Description

Registers a user-supplied procedure to call each time a mouse click occurs. This is useful if a client wishes to handle mouse clicks, and the client is better implemented as something other than an annotation or a tool. If more than one routine is registered to receive mouse clicks, the most recently registered routine is called first.

To un-register, you *must* use the same callback that was used to register; you cannot use a newly created callback. To accomplish this, call [ASCallbackCreateProto](#) once before registering and use the value returned from this call both to register and un-register; do not call [ASCallbackCreateProto](#) a second time when un-registering.

Parameters

clickProc	User-supplied callback to call each time a mouse click occurs. If the user-supplied callback returns true , it indicates that the procedure handled the mouse click, and it should not be passed along to the Acrobat viewer or other clients that registered to receive mouse clicks. If it returns false , the mouse click is passed to the Acrobat viewer or other clients that registered to receive mouse clicks.
data	Pointer to user-supplied data to pass to clickProc each time it is called.

Return Value

None

Known Exceptions

[genErrNoMemory](#)

Header File

AVCalls.h

Related Methods

[AVAppRegisterForPageViewRightClicks](#)
[AVAppUnregisterForPageViewClicks](#)

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

NOTE: Numeric types in the callback have changed in Acrobat 6.0.

AVAppRegisterForPageViewDrawing

```
void AVAppRegisterForPageViewDrawing (AVPageViewDrawProc proc,
void* data);
```

Description

Registers a user-supplied procedure to call each time a window is drawn, after the page's contents and annotations have been drawn. This allows clients to draw whatever they wish to on top of pages. The procedure is called each time the page view changes (scrolls, zooms, goes to a different page).

To unregister, you *must* use the same callback that was used to register; you cannot use a newly created callback. To do this, call [ASCallbackCreateProto](#) once before registering and use the value returned from this call both to register and unregister; do not call [ASCallbackCreateProto](#) a second time when unregistering.

You can register only one procedure, but you can register it more than once, with a different client data pointer in each case. If more than one procedure-data combination is registered, they are called in a last-in, last-out order. In versions prior to 6.0, you should not do this, as the call to [AVAppUnregisterForPageViewDrawing](#) unregisters only one of the registered combinations, and does not guarantee which one it is. However, in version 6.0 or later, you can pass the same **proc** and **clientData** pointer to [AVAppUnregisterForPageViewDrawingEx](#) to unregister that particular combination.

NOTE: As of version 5.0, Acrobat renders PDF pages offscreen before copying them to the display memory. Windows developers who call [AVPageViewAcquireMachinePort](#) on the [AVPageView](#) passed to the drawing callback should note that the HWND points to the screen but the HDC points to an offscreen bitmap. All drawing should be performed using the returned HDC. Developers should not create their own HDC based on the returned HWND because their content will be lost when Acrobat copies the offscreen buffer to the display.

Parameters

proc	User-supplied callback to call each time a window is drawn.
data	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Known Exceptions

[genErrNoMemory](#)

**Header File**

AVCalls.h

Related Methods

[AVAppUnregisterForPageViewDrawing](#)

[AVAppUnregisterForPageViewDrawingEx](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

NOTE: Numeric types in the callback have changed in Acrobat 6.0.

AVAppRegisterForPageViewKeyDown

```
void AVAppRegisterForPageViewKeyDown
(AVPageViewKeyDownProc proc, void* data);
```

Description

Registers a user-supplied procedure to call each time a key is pressed in an **AVPageView**. To un-register, you *must* use the same callback that was used to register; you cannot use a newly created callback. To accomplish this, call [ASCallbackCreateProto](#) once before registering and use the value returned from this call both to register and un-register; do not call [ASCallbackCreateProto](#) a second time when un-registering.

Parameters

proc	User-supplied callback to call each time a key is pressed.
data	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Known Exceptions

[genErrNoMemory](#)

Header File

AVCalls.h

Related Methods

[AVAppUnregisterForPageViewKeyDown](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00040005** or higher.

AVAppRegisterForPageViewRightClicks

```
void AVAppRegisterForPageViewRightClicks
(AVPageViewClickProc proc, void* data);
```

Description

Registers a user-supplied procedure to call each time a user clicks the right mouse button in an **AVPageView**. On MacOS, the procedure is called when the user holds down the Ctrl key and clicks the mouse button.

To unregister, you *must* use the same callback that was used to register; you cannot use a newly created callback. To accomplish this, call [ASCallbackCreateProto](#) once before registering and use the value returned from this call both to register and unregister; do not call [ASCallbackCreateProto](#) a second time when un-registering.

Parameters

proc	User-supplied callback to call each time the right mouse button is clicked.
data	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Known Exceptions

[genErrNoMemory](#)

Header File

AVCalls.h

Related Methods

[AVAppUnregisterForPageViewClicks](#)
[AVAppUnregisterForPageViewRightClicks](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVAppRegisterFromPDFHandler

```
void AVAppRegisterFromPDFHandler
(AVConversionFromPDFHandler conversionHandler) ;
```

Description

Registers an **AVConversionFromPDFHandler** to export from PDF to other file formats. When a “FromPDF” converter is registered, the converter is automatically added to the list of supported file formats in the **SaveAs** dialog. In addition, the converter is displayed in the list of “FromPDF” converters for Batch framework.

The handler is only required to implement the **convert** callback. All others are optional. If a handler fails to implement the **convert** callback, the handler will not be registered.

Parameters

conversionHandler	The handler.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppRegisterToPDFHandler](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVAppRegisterGlobalCommand

```
void AVAppRegisterGlobalCommand (AVCommand cmd) ;
```

Description

Registers an **AVCommand** in the global command list. The application assumes ownership of the resources associated with **cmd**. As such, the client must not call [AVCommandDestroy](#).

Parameters

cmd	The AVCommand to register.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppFindGlobalCommandByName](#)
[AVAppUnregisterGlobalCommand](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVAppRegisterHowToPanel

```
void AVAppRegisterHowToPanel (ASAtom panelName,
const char *panelURL, AVIcon panelIcon,
ASConstText panelTitle, ASBool showInHomepage,
ASPathName urlDirectory, ASUns32 sortKey);
```

Description

Registers a HowTo panel with the application.

Parameters

panelName	The internal language-independent name of the HowTo panel to register.
panelURL	The file name for the XML/HTML content, specified as a diPath (see PDFFileSpecGetDIPath). The path is relative to urlDirectory if supplied; otherwise it is relative to the application's localized HowTo folder.
panelIcon	The icon to show in the upper-left corner of the panel.
panelTitle	The text to display in the panel's title. The viewer make a copy of this text.
showInHomepage	true to show the title on the HowTo panel's home page, false otherwise. To hide or show the title dynamically (based on the contents of the current document, for example) use a compute-visible callback. See AVAppSetHowToPanelComputeVisibleProc .
urlDirectory	The directory where the panelURL and other XML/HTML pages are located, specified as a diPath , or NULL for the default location (the application's localized HowTo folder).
sortKey	A key by which to order items on the HowTo panel's home page.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppAutoShowHowToPanel](#)
[AVAppGetHowToPanelAutoShow](#)

AVAppSetHowToPanelAutoShow
AVAppSetHowToPanelAutoShowText
AVAppSetHowToPanelComputeVisibleProc

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVAppRegisterIdleProc

```
void AVAppRegisterIdleProc (AVIDleProc idleProc,
void* clientData, ASUns32 period);
```

Description

Registers a user-supplied procedure to call “regularly” when the Acrobat viewer is otherwise idle. If more than one idle procedure is registered, they are all called in a round robin order. The registered idle procs may be called when the Acrobat viewer is not the frontmost application. In addition, in Mac OS, the registered idle procs receive idle events any time a movable modal dialog or modal **AVWindow** is displayed, but not a system-modal one. Use **AVAppModalWindowIsOpen** if you wish to determine if a modal window is open.

To un-register, you *must* use the same callback that was used to register; you cannot use a newly created callback. To accomplish this, call **ASCallbackCreateProto** once before registering and use the value returned from this call both to register and un-register; do not call **ASCallbackCreateProto** a second time when un-registering.

Parameters

idleProc	User-supplied callback to call at idle time.
clientData	Pointer to user-supplied data to pass to idleProc each time it is called.
period	Minimum time between calls to idleProc . idleProc will not be called any more frequently than period , but it may be called less frequently. period is specified in ticks (one tick is 1/60 of a second).

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppUnregisterIdleProc](#)
[AVAppModalWindowIsOpen](#)

**Example**

```
ACCB1 void ACCB2 myIdleProc(void* data)
{
    /* do some stuff */
}
AVAppRegisterIdleProc(
    ASCallbackCreateProto(AVIdleProc,
        &myIdleProc), NULL, 15);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppRegisterNotification

```
void AVAppRegisterNotification (NSElector nsel,
    ASExtension owner, void* proc, void* clientData);
```

Description

Registers a user-supplied procedure to call when the specified event occurs.

Many notifications appear in **Will/Did** pairs, for example [AVDocWillPerformAction](#) and [AVDocDidPerformAction](#). It is possible that an operation may fail after the **Will** notification and before the **Did** notification. When this occurs, the **Did** notification is still broadcast, but the **err** parameter in the **Did** notification is nonzero, and represents the error that occurred. When **err** is nonzero, the other parameters are not necessarily valid. Always check **err** in a **Did** notification before using the other parameters.

When calling [AVAppUnregisterNotification](#) to un-register for a notification, you *must* pass the **proc**, **clientData**, and **owner** that were used when the notification was registered using [AVAppRegisterNotification](#). You *must* use the *same* callback that was used to register; you cannot use a newly created callback. To accomplish this, call [ASCallbackCreateNotification](#) once before registering, and use the value returned from this call both to register and un-register; do not call [ASCallbackCreateNotification](#) a second time when un-registering. You will then need to destroy the pointer to the callback using the [ASCallbackDestroy](#) method.

Parameters

nsel	The notification type. Must be one of the notification selectors (see Notifications). The notification selector is the name of the notification with the characters NSEL appended. For example, the selector for AVDocDidOpen is AVDocDidOpenNSEL .
owner	The gExtensionID of the client registering the notification.
proc	User-supplied callback to call when the notification occurs. Its declaration depends on the notification type (see Notifications). Remember to use ASCallbackCreateNotification to convert proc to a callback before passing it to AVAppRegisterNotification .
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppUnregisterNotification](#)
[ASCallbackCreateNotification](#)
[ASCallbackDestroy](#)

Example

```
ASCallback myCallback;

myCallback = ASCallbackCreateNotification(
    AVDocDidOpen, &CalcAVDocDidOpen);
AVAppRegisterNotification(AVDocDidOpenNSEL,
    gExtensionID, myCallback, NULL);
...
/* Unregister for the notification */
AVAppUnregisterNotification(
    AVDocDidOpenNSEL, gExtensionID,
    myCallback, NULL);

/* Destroy Pointer*/
ASCallbackDestroy(myCallback);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppRegisterTool

```
void AVAppRegisterTool (AVTool tool);
```

Description

Registers the specified tool with the Acrobat viewer. The tool is not registered if its [GetTypeProcType](#) callback returns **NULL**.

Parameters

tool	Structure containing the tool's callbacks. This structure must <i>not</i> be freed after calling AVAppRegisterTool , but must be retained.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppEnumTools](#)

[AVAppSetActiveTool](#)

Example

```
static AVToolRec myTool;
{
    memset(&myTool, 0, sizeof(AVToolRec));
    myTool.size = sizeof(AVToolRec);

    myTool.Activate = ASCallbackCreateProto(
        ActivateProcType, &myToolActivate);
    myTool.GetType = ASCallbackCreateProto(
        GetTypeProcType, &myToolGetType);
    myTool.AdjustCursor =
        ASCallbackCreateProto(AdjustCursorProcType,
            &myToolAdjustCursor);
    myTool.DoClick = ASCallbackCreateProto(
        DoClickProcType, &myToolDoClick);
    myTool.ComputeEnabled =
        myToolIsEnabledCallback; /* already callbackCreated */
    myTool.computeEnabledData = NULL;
    AVAppRegisterTool(&myTool);
}
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppRegisterToolBarPosition

```
void AVAppRegisterToolBarPosition (const char* name,
    ASBool external, AVToolBarPosition position);
```

Description

Sets the position of the toolbar. A toolbar can have separate positional attributes for internal and external views. The position is set when the user first opens Acrobat; after that, the user's preferences dictate where the toolbar is located.

Parameters

name	The name of the toolbar. Refer to Toolbar and Toolbar Button Names for a list of the standard named toolbars.
external	If true , the preferences are associated with the external view.
position	The structure defining the toolbar's attributes.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVToolBarNew](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

NOTE: Numeric types in the data structure have changed in **0x00060000**.

AVAppRegisterToPDFHandler

```
void AVAppRegisterToPDFHandler  
(AVConversionToPDFHandler conversionHandler);
```

Description

Registers an **AVConversionToPDFHandler** to import other file formats. When a “ToPDF” converter is registered, the converter is automatically added to the list of supported file formats in the **Open** dialog. In addition, the converter is displayed in the list of “ToPDF” converters for Batch framework.

Parameters

conversionHandler	The handler.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppRegisterFromPDFHandler](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVAppRegisterTransHandler

```
void AVAppRegisterTransHandler (AVTransHandler avth);
```

Description

Registers a transition handler within Acrobat. If **avth** has not implemented the **GetType** callback, it will not be registered.

Parameters

avth	An AVTransHandler structure containing pointers to the transition handler's functions. This structure must <i>not</i> be freed after calling AVAppRegisterTransHandler .

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppEnumTransHandlers](#)
[AVAppGetTransHandlerByType](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

AVAppSaveDialog

```
ASBool AVAppSaveDialog (AVOpenSaveDialogParams dialogParams,
ASFileSys* outFileSys, ASPathName* outPathName,
AVFilterIndex* ioChosenFilterIndex);
```

Description

Displays a platform-dependent file save dialog. It is the caller’s responsibility to release the returned **ASPathName**.

Parameters

dialogParams	Standard dialog parameters for Open/Save/ChooseFolder dialogs.
outFileSys	<i>(Filled by the method, may be NULL)</i> The file system through which pathName was obtained. May be NULL if kAVOpenSaveAllowForeignFileSystems is not passed in dialogParams .
outPathName	<i>(Filled by the method)</i> The ASPathName associated with the file selected by the user. The caller must free the filled in ASPathName using the supplied ASFileSys .
ioChosenFilterIndex	<i>(Filled by the method, may be NULL)</i> The index of the filter chosen by the user to select the file. May be NULL if caller does not care which filter was chosen, -1 for “All Files”.

Return Value

Returns **true** if user confirmed the selection, **false** if user clicked cancel or some error occurred.

Header File

```
AVCalls.h
```

Related Methods

```
AVAppOpenDialog
AVAppChooseFolderDialog
```

Product

```
reader, base, pro
```

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVAppSetActiveTool

```
void AVAppSetActiveTool (AVTool tool, ASBool persistent);
```

Description

Sets the active tool. Does nothing if the specified tool is not currently enabled. Whether a tool is enabled or not is determined by the [AVComputeEnabledProc](#) callback in the [AVTool](#) structure. If this callback is **NULL**, the tool is always enabled.

NOTE: It is recommended that you use [AVDocSetActiveTool](#) rather than [AVAppSetActiveTool](#), preferably specifying a particular document.

Parameters

tool	The tool to set as the active tool.
persistent	A flag that indicates a preference as to whether the tool stays active after it is used. true is a hint that the tool should, if possible, stay active for an arbitrary number of "operations" (whatever that happens to be) rather than doing a "one shot" operation and restoring the prior active tool. Persistence is not enforced by the Acrobat viewer. It is up to a one-shot tool to restore the previously active tool (determined using AVAppGetLastActiveTool), or to restore the default tool (determined using AVAppGetDefaultTool) if there was no previously active tool.

Return Value

None

Header File

AVCalls.h

Related Methods

- [AVAppGetActiveTool](#)
- [AVAppGetLastActiveTool](#)
- [AVAppGetDefaultTool](#)
- [AVDocSetActiveTool](#)

Example

```
if (!toolPersistent)
    AVAppSetActiveTool(
        AVAppGetLastActiveTool(), true);
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppSetHowToPanelAutoShow

```
void AVAppSetHowToPanelAutoShow(ASAtom panelName,  
ASBool autoShow);
```

Description

Set the current auto-show state of a HowTo panel. The auto-show state is **true** by default. When the auto-show state is **true**, the panel is shown whenever when the client calls [AVAppAutoShowHowToPanel](#). Set it to **false** to disable the client’s auto-show behavior for the panel. See also [AVAppSetHowToPanelAutoShowText](#).

Parameters

panelName	The internal language-independent name of the HowTo panel whose auto-show state is set.
autoShow	The new auto-show state, true or false .

Return Value

None

Header File

AVCalls.h

Related Methods

- [AVAppAutoShowHowToPanel](#)
- [AVAppGetHowToPanelAutoShow](#)
- [AVAppSetHowToPanelAutoShowText](#)
- [AVAppSetHowToPanelComputeVisibleProc](#)
- [AVAppRegisterHowToPanel](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVAppSetHowToPanelAutoShowText

```
void AVAppSetHowToPanelAutoShowText (ASAtom panelName,
ASConstText checkBoxText) ;
```

Description

When the auto-show state of a HowTo panel is **true**, the panel is shown whenever when the client calls [AVAppAutoShowHowToPanel](#). To give users the option of suppressing the auto-show behavior, use this method to register some text, which appears in a checkbox at the bottom of the panel. If the user clears the checkbox, use [AVAppSetHowToPanelAutoShow](#) to suppress the auto-show behavior for the panel.

Parameters

panelName	The internal language-independent name of the HowTo panel whose auto-show suppression text is set.
checkBoxText	The text to display in the auto-show suppression check box at the bottom of the HowTo panel. The view copies this text. The string should be of the form "Show XXXX when XXX". For example, "Show home page at startup".

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppAutoShowHowToPanel](#)
[AVAppGetHowToPanelAutoShow](#)
[AVAppSetHowToPanelAutoShow](#)
[AVAppSetHowToPanelComputeVisibleProc](#)
[AVAppRegisterHowToPanel](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVAppSetHowToPanelComputeVisibleProc

```
void AVAppSetHowToPanelComputeVisibleProc (ASAtom panelName,
AVComputeVisibleProc p, void* data);
```

Description

Registers a new HowTo panel callback with the application. This callback is executed at various times to determine whether the panel's title will appear in the HowTo home page. The title appears if and only if **showInHomePage** was set to **true** in the call to [AVAppRegisterHowToPanel](#) and this callback returns **true**.

If this callback is not registered, the panel title appears if **showInHomePage** was set to **true** in the call to [AVAppRegisterHowToPanel](#).

Register this callback if you want the title to appear only under certain conditions—for example, when the active document has some special characteristics or attributes.

Parameters

panelName	The internal, language-independent name of the HowTo panel.
p	The procedure to call to determine whether the panel should appear on the how-to home page.
data	A pointer to data to pass to the procedure.

Return Value

None

Header File

AVExpT.h

Related Methods

[AVAppAutoShowHowToPanel](#)
[AVAppGetHowToPanelAutoShow](#)
[AVAppSetHowToPanelAutoShow](#)
[AVAppSetHowToPanelAutoShowText](#)
[AVAppRegisterHowToPanel](#)

AVAppSetPreference

```
void AVAppSetPreference (AVPrefsType preference, void* value);
```

Description

Sets the value of the specified built-in application preference. The preference values are automatically saved to disk when a new value is set.

Parameters

preference	The preference value to set. See AVPrefsType for a list of preference descriptions.
value	The new value for preference . The type of this value is dependent on the preference being set. See AVPrefsType for more information.

Return Value

None

Notifications

[AVAppOldPrefDidChange](#)

Header File

AVCalls.h

Related Methods

[AVAppGetPreference](#)

Example

```
if (AVAppGetPreference(avpSkipWarnings) || AAlertConfirm("Really?")) {
    AVAppSetPreference(avpShowToolBar, (void *)true);
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppUnregisterForPageViewAdjustCursor

```
void AVAppUnregisterForPageViewAdjustCursor
(AVPageViewCursorProc cursorProc);
```

Description

Un-registers a user-supplied adjust cursor procedure.

To un-register, you *must* use the same callback that was used to register; you cannot use a newly created callback. To accomplish this, call [ASCallbackCreateProto](#) once before registering and use the value returned from this call both to register and un-register; do not call [ASCallbackCreateProto](#) a second time when un-registering.

Parameters

cursorProc	The original callback.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppRegisterForPageViewAdjustCursor](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppUnregisterForPageViewClicks

```
void AVAppUnregisterForPageViewClicks
(AVPageViewClickProc clickProc);
```

Description

Un-registers a user-supplied page view click procedure.

To un-register, you *must* use the same callback that was used to register; you cannot use a newly created callback. To accomplish this, call [ASCallbackCreateProto](#) once before registering and use the value returned from this call both to register and un-register; do not call [ASCallbackCreateProto](#) a second time when un-registering.

Parameters

clickProc	The original callback.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppRegisterForPageViewClicks](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVAppUnregisterForPageViewDrawing

```
void AVAppUnregisterForPageViewDrawing
(AVPageViewDrawProc proc);
```

Description

Un-registers a user-supplied page view drawing procedure.

To un register, you *must* use the same callback that was used to register; you cannot use a newly created callback. To accomplish this, call [ASCallbackCreateProto](#) once before registering and use the value returned from this call both to register and un-register; do not call [ASCallbackCreateProto](#) a second time when un-registering.

NOTE: Superseded by [AVAppUnregisterForPageViewDrawingEx](#) in Acrobat 6.0.

Parameters

proc	The original callback.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppRegisterForPageViewDrawing](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppUnregisterForPageViewDrawingEx

```
void AVAppUnregisterForPageViewDrawingEx (void* proc, void
*clientData);
```

Description

Un-registers a user-supplied page view drawing procedure.

NOTE: Supersedes [AVAppUnregisterForPageViewDrawing](#) in Acrobat 6.0.

To unregister, you *must* use the same callback that was used to register; you cannot use a newly created callback. To accomplish this, call [ASCallbackCreateProto](#) once before registering and use the value returned from this call both to register and unregister; do not call [ASCallbackCreateProto](#) a second time when unregistering.

You can register only one procedure; however, in version 6.0, you can register it more than once, with a different client data pointer in each case. You can then pass the same **proc** and **clientData** pointer to this method to unregister that particular combination.

Parameters

proc	The original callback.
clientData	A pointer to data to pass to the callback.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppRegisterForPageViewDrawing](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVAppUnregisterForPageViewKeyDown

```
void AVAppUnregisterForPageViewKeyDown
(AVPageViewKeyDownProc proc) ;
```

Description

Un-registers a user-supplied page view key down procedure.

To un-register, you *must* use the same callback that was used to register; you cannot use a newly created callback. To accomplish this, call [ASCallbackCreateProto](#) once before registering and use the value returned from this call both to register and un-register; do not call [ASCallbackCreateProto](#) a second time when un-registering.

Parameters

proc	The original callback.

Return Value

None

Known Exceptions

[genErrNoMemory](#)

Header File

AVCalls.h

Related Methods

[AVAppRegisterForPageViewKeyDown](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040005** or higher.

AVAppUnregisterForPageViewRightClicks

```
void AVAppUnregisterForPageViewRightClicks
(AVPageViewClickProc proc);
```

Description

Un-registers a user-supplied procedure to call each time a user clicks the right mouse button in an **AVPageView**.

To un-register, you *must* use the same callback that was used to register; you cannot use a newly created callback. To accomplish this, call [ASCallbackCreateProto](#) once before registering and use the value returned from this call both to register and un-register; do not call [ASCallbackCreateProto](#) a second time when un-registering.

Parameters

proc	The original callback.

Return Value

None

Known Exceptions

[genErrNoMemory](#)

Header File

AVCalls.h

Related Methods

[AVAppRegisterForPageViewRightClicks](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



AVAppUnregisterGlobalCommand

```
void AVAppUnregisterGlobalCommand (AVCommand cmd) ;
```

Description

Removes an **AVCommand** from the global command list.

Parameters

cmd	The command.

Return Value

None

Header File

AVCalls.h

Related Methods

- [AVAppFindGlobalCommandByName](#)
- [AVAppRegisterGlobalCommand](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVAppUnregisterIdleProc

```
void AVAppUnregisterIdleProc (AVIDleProc idleProc,  
void* clientData);
```

Description

Un-registers a user-supplied idle procedure.

To un-register, you *must* use the same callback that was used to register; you cannot use a newly created callback. To accomplish this, call [ASCallbackCreateProto](#) once before registering and use the value returned from this call both to register and un-register; do not call [ASCallbackCreateProto](#) a second time when un-registering.

Parameters

idleProc	The original callback.
clientData	The original clientData.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppRegisterIdleProc](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppUnregisterNotification

```
void AVAppUnregisterNotification (NSElector nsel,
    ASExtension owner, void* proc, void* clientData);
```

Description

Un-registers a user-supplied notification procedure.

To un-register, you *must* use same callback, **clientData**, and **owner** that were used when the notification was registered using [AVAppRegisterNotification](#). To accomplish this, call [ASCallbackCreateNotification](#) once before registering and use the value returned from this call both to register and un-register; do not call [ASCallbackCreateNotification](#) a second time when un-registering.

Parameters

nsel	The notification type.
owner	The gExtensionID of the client registering the notification.
proc	The original callback.
clientData	The original clientData .

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppRegisterNotification](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVAppYieldToOtherApps

```
void AVAppYieldToOtherApps (double yieldTimeout);
```

Description

Yield to other applications for **yieldTimeout**.

NOTE: Macintosh platform clients will need to call **AVAppYieldToOtherApps** instead of **WaitNextEvent** to prevent problems with Mac OS Carbon event handling. **AVAppYieldToOtherApps** calls **ReceiveNextEvent**. It is recommended, that you use the **kEventDurationForever** constant as the argument to **AVAppYieldToOtherApps**, though this will impact portability since it is a CodeWarrior define (in **CarbonEvents.h**).

Parameters

yieldTimeout	The minimum amount of time to yield in seconds. Only applicable for Mac OS 8.6 and 9.x. A no-op for Windows and UNIX.

Return Value

None

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



AVHasAuxDataHandler

```
ASBool AVHasAuxDataHandler (ASAtom dataType);
```

Description

Indicates whether a handler exists for the specified data type.

Parameters

dataType	Name of data handler being queried.

Return Value

true if a handler is registered, **false** if a handler is not registered.

Header File

AVCalls.h

Related Methods

[AVDocSendAuxData](#)
[AVRegisterAuxDataHandler](#)
[AVUnregisterAuxDataHandler](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

AVRegisterAuxDataHandler

```
ASBool AVRegisterAuxDataHandler (ASExtension extension,  
ASAtom auxDataType, AVAuxDataHandler handler);
```

Description

Registers an **AuxDataHandler**.

A handler can be registered for more than one kind of auxiliary data. The type of data is passed to the **AuxDataHandler** at **Perform** time so that it can distinguish what type of data it is receiving.

Parameters

extension	The gExtensionID of the client registering the handler.
auxDataType	The ASAtom for the type of data that the handler accepts.
handler	The handler to register.

Return Value

true if the handler was registered, **false** if the handler could not be un-registered. For example, if another handler is already registered for this time.

Header File

AVCalls.h

Related Methods

[AVDocSendAuxData](#)
[AVHasAuxDataHandler](#)
[AVUnregisterAuxDataHandler](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

AVUnregisterAuxDataHandler

```
ASBool AVUnregisterAuxDataHandler (ASExtension extension,
ASAtom auxDataType, AVAuxDataHandler handler);
```

Description

Un-registers a previously registered **AuxDataHandler**.

Parameters

extension	The gExtensionID of the client un-registering the handler.
auxDataType	Type of data for the handler being un-registered.
handler	The handler to un-register.

Return Value

true if the handler was un-registered, **false** if the handler could not be un-registered. For example, returns **false** if the handler was never registered.

Header File

AVCalls.h

Related Methods

[AVHasAuxDataHandler](#)
[AVRegisterAuxDataHandler](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.



AVCommand

AVCommandCancel

```
AVCommandStatus AVCommandCancel (AVCommand cmd) ;
```

Description

Cancels the specified command. If the command is currently processing a file, the command might not attempt to roll back any modifications it has made to the document. This may leave the PDF in an invalid state.

Parameters

<code>cmd</code>	The command to cancel. See AVCommand Descriptions (Built-in Commands) .

Return Value

The current status of `cmd`. Should be `kAVCommandCanceled`.

Header File

`AVCalls.h`

Related Methods

[AVCommandReset](#)
[AVCommandWork](#)

Product

reader, base, pro

Availability

Available if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

AVCommandDestroy

```
void AVCommandDestroy (AVCommand cmd) ;
```

Description

Destroys the specified command and its associated resources.

Parameters

cmd	The command to destroy. See AVCommand Descriptions (Built-in Commands) .

Return Value

None

Header File

AVCalls.h

Related Methods

[AVCommandNew](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVCommandExecute

[AVCommandStatus](#) AVCommandExecute ([AVCommand](#) cmd) ;

Description

This method runs the specified **AVCommand**, after it has been set up using [AVCommandSetInputs](#) and [AVCommandSetParams](#).

The method runs the command using the following process:

- Reset the command if its status is not **AVCommandReady**
- If only a [PDDoc](#) input has been configured, **AVCommandExecute** does nothing to the inputs. If no [AVDoc](#) or [PDDoc](#) inputs have been configured, **AVCommandExecute** configures the command to use the active **AVDoc**. In all cases, if an **AVDoc** has been configured (either by **AVCommandExecute** or its caller) the implementation ensures that the [PDDoc](#) input matches the **AVDoc** input.
- If the command can display a dialog (that is, if the command handler has a "ShowDialog" callback and the command's **kAVCommandKeyCanShowDialog** property is **true**), present the dialog to allow the user to alter the command's parameters.
- Repeatedly call [AVCommandWork](#) on **cmd** until that method returns a status other than **kAVCommandWorking**.

This method should be used when the client does not need the level of control provided when calling **AVCommandWork** directly. Specifically, this method is useful for using a global command from within a menu or toolbar button's execution procedure.

Parameters

cmd	The command to execute. See AVCommand Descriptions (Built-in Commands) .

Return Value

The current status of **cmd**.

Header File

AVCalls.h

Related Methods

[AVCommandSetInputs](#)
[AVCommandSetParams](#)
[AVCommandWork](#)

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



AVCommandGetAVDoc

```
AVDoc AVCommandGetAVDoc (AVCommand cmd) ;
```

Description

Retrieves the **AVDoc** from a command's inputs **ASCab**.

NOTE: In some cases, this call returns a **NULL AVDoc** even when **AVCommandGetPDDoc** returns a non-**NULL PDDoc** and an **AVDoc** is open on that **PDDoc**. In these cases, you should probably ignore the **AVDoc** or use **AVDocFromPDDoc** to find the associated **AVDoc**.

Parameters

cmd	The command to be queried. See AVCommand Descriptions (Built-in Commands) .

Return Value

The **AVDoc** if it was found, **NULL** otherwise.

Header File

AVCalls.h

Related Methods

[AVCommandSetInputs](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVCommandGetCab

[ASCab](#) AVCommandGetCab ([AVCommand](#) cmd, const char* key, [ASBool](#) create);

Description

Returns the **ASCab** stored in the specified command under **key**. If the cabinet is not available, this method will create it if **create** is set to **true**. The command retains ownership of the returned cabinet

This method can only be exercised by an [AVCommandHandler](#). Command handlers can use this call to attach arbitrary information to a command. Only the command handler should access these cabinets. See the examples for [AVCommandSetInputs](#) and [AVCommandSetParams](#).

Parameters

cmd	The command to be queried. See AVCommand Descriptions (Built-in Commands) .
key	The key that the cabinet is stored under.
create	Pass true to have this method create the cabinet if it does not exist.

Return Value

The cabinet if it was found/created, **NULL** otherwise.

Header File

AVCalls.h

Related Methods

[AVCommandPutCab](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVCommandGetCancelProc

```
ASCancelProc AVCommandGetCancelProc (AVCommand cmd) ;
```

Description

Returns a cancellation procedure for the command. If the user interface policy is anything other than **kAVCommandUIInteractive**, this cancellation procedure must be used. If the UI policy is **kAVCommandUIInteractive**, an appropriate default cancel proc will be provided but the command may use a more appropriate one if it wishes. Pass in the command itself as the client data.

Parameters

cmd	The command whose cancelation procedure is returned. See AVCommand Descriptions (Built-in Commands) .

Return Value

The cancelation procedure.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVCommandGetConfig

```
void AVCommandGetConfig (AVCommand cmd, ASCab config)
```

Description

Retrieves the configuration parameters for the specified command.

Configuration values are controlled by the mechanism that is driving the command—that is, a client client or the batch framework. For example, one configuration value is whether to display a dialog while executing; a setting that the user can toggle through the UI, or a client client can set through [AVCommandSetConfig](#).

To retrieve all the configuration parameters, pass in an empty cabinet for **config** and the method will fill in the cabinet with the current configuration keys/values. To retrieve specific configuration parameter values, fill the initial dictionary with the keys you're interested in, each with a **NULL** value, and the method will attempt to fill in the appropriate value for each key.

Currently the only configuration parameters defined are:

- UIPolicy (**kASValueInteger**)—The local UI policy for this command.
- ParamsDescOpen (**kASValueBool**)—Indicates whether the description of this command's parameters is visible in the sequence view.

NOTE: The caller retains ownership of **config**.

NOTE: Normally the batch sequence framework controls the configuration of commands in batch sequences.

Parameters

cmd	The command whose configuration parameters are being retrieved. See AVCommand Descriptions (Built-in Commands) .
config	(Filled by the method) An ASCab into which the parameters are written.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVCommandSetConfig](#)
[AVCommandGetUIPolicy](#)

Product

reader, base, pro

**Availability**

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



AVCommandGetInputs

```
void AVCommandGetInputs (AVCommand cmd, ASCab inputs);
```

Description

Retrieves the input parameters of the specified command. The inputs of a command consist of an **ASCab** detailing properties of the object the command will operate on. The contents of the cabinet are dependent on the mechanism through which the command is initialized. The batch framework attempts to provide the following:

Key (see Command Keys)	Stored As	Description
kAVCommandKeyPDDoc	kASValuePointer	The document being processed.
kAVCommandKeyBaseFileName	kASValueString	The name of the input file, without extension.
kAVCommandKeyOrigExtension	kASValueString	The extension of the input file.

Client clients should also attempt to provide this information. The content of the inputs cabinet is not limited to the keys listed above. Clients (or the command handler itself) can specify additional inputs as necessary. The caller retains ownership of **inputs**.

NOTE: [AVCommandExecute](#) will also attempt to provide **kAVCommandKeyAVDoc**, a pointer to the [AVDoc](#) being operated on.

NOTE: You can use [AVCommandGetPDDoc](#) and [AVCommandGetAVDoc](#) to quickly retrieve the [PDDoc](#) or [AVDoc](#) in a command's inputs.

Parameters

cmd	The command whose input parameters are being retrieved. See AVCommand Descriptions (Built-in Commands) .
inputs	<i>(Filled by the method)</i> The input parameters.

Header File

AVCalls.h

Related Methods

[AVCommandSetInputs](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



AVCommandGetName

```
ASAtom AVCommandGetName (AVCommand cmd) ;
```

Description

Returns the name of the command.

Parameters

cmd	The command whose name is returned. See AVCommand Descriptions (Built-in Commands) .

Return Value

The command name.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVCommandGetParams

```
void AVCommandGetParams (AVCommand cmd, ASCab params);
```

Description

Retrieves the parameter set for the specified command. It is possible that some commands have no parameters that can be configured, in which case no data will be written to **params**. The **params** cabinet should be empty when passed in. Upon return it will be filled with copies of all of the command's parameter settings.

NOTE: The caller retains ownership of **params**.

Parameters

cmd	The command whose parameters are being retrieved. See AVCommand Descriptions (Built-in Commands) .
params	<i>(Filled by the method)</i> An ASCab into which the parameter set is written.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVCommandSetParams](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



AVCommandGetPDDoc

`PDDoc AVCommandGetPDDoc (AVCommand cmd) ;`

Description

Retrieves the **PDDoc** from a command’s inputs **ASCab**.

Parameters

<code>cmd</code>	The command to be queried. See AVCommand Descriptions (Built-in Commands) .

Return Value

A **PDDoc** if it was found, **NULL** otherwise.

Header File

`AVCalls.h`

Product

reader, base, pro

Availability

Available if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVCommandGetProgressMonitor

[ASProgressMonitor](#) AVCommandGetProgressMonitor ([AVCommand](#) cmd) ;

Description

Returns a progress monitor the command can use to report progress. The command itself should be passed in as the **clientData** associated with the monitor's callbacks.

Command handlers should use this progress monitor rather than a custom implementation.

If the command is driven by the batch framework, the progress monitor updates the batch progress dialog. If the command is driven by a client *and* the **AVDoc** input key has been set, the progress monitor updates the progress pane at the bottom of the document's window, otherwise the progress monitor's callbacks are no-ops.

Parameters

cmd	The command to be queried. See AVCommand Descriptions (Built-in Commands) .

Return Value

The command progress monitor.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVCommandGetProps

```
void AVCommandGetProps (AVCommand cmd, ASCab props);
```

Description

Retrieves the properties of the specified command. Properties are values intrinsic to the command or to the command's current state.

To retrieve properties, create a cabinet containing the keys for the properties you're interested in (the values should be invalid, such as **NULLs**), and pass it into this method. The command will copy the requested properties into your cabinet. If the cabinet is empty, the command will fill in all properties.

The following properties are pre-defined:

Property	Type	Description
CanShowDialog	kASValueBool	Indicates that the command can show a configuration dialog. Support of this property is optional—defaults to false .
CanDescribeParams	kASValueBool	Indicates that the command can provide a description of its parameters. Support of this property is optional—defaults to false . If the command sets this property to true , it should support the ParamDesc property also.
ParamsDesc	kASValueCabinet	A cabinet containing numeric keys—that is, "1", "2", ..., "n" with kASValueText values describing the command's parameters. The text items will be displayed in the sequence dialog in the numerical order of the keys.
CanBatch	kASValueBool	Indicates that the command should appear in the list of commands that can be batched. Support of this property is optional—defaults to false . If the command set returns this property value as true , it MUST support the GenericTitle and GroupTitle properties also.
GroupTitle	kASValueText	Name of the group to which this command belongs. Currently the GroupTitles are localized strings, so vary from one language to another. The English GroupTitles are those listed in the edit sequence dialog: "Comments," "Document," "JavaScript," "Page," and "PDF Consultant."
GenericTitle	kASValueText	Generic name for this command, for example, "Show/Hide Bookmarks", "Rotate Pages", and so forth. This is used when displaying the command in the list of global commands.

Property	Type	Description
Title	kASValueText	Specific title for this command, for example, "Hide Bookmarks" taking the command's parameters into account. This is used for displaying a specific command within a sequence.

Parameters

cmd	The command whose properties are being retrieved. See AVCommand Descriptions (Built-in Commands) .
props	<i>(Filled by the method)</i> The command properties. The caller retains ownership of the ASCab .

Return Value

None

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVCommandGetReportProc

```
ASReportProc AVCommandGetReportProc (AVCommand cmd) ;
```

Description

Returns the [ASReportProc](#) associated with the specified command. [AVCommandHandlers](#) should report all error and status messages through this procedure.

Pass the [AVCommand](#) itself as the `clientData` parameter when calling the procedure.

Parameters

<code>cmd</code>	The command whose ASReportProc is returned. See AVCommand Descriptions (Built-in Commands) .

Return Value

The [ASReportProc](#).

Header File

`AVCalls.h`

Product

reader, base, pro

Availability

Available if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

AVCommandGetStatus

[AVCommandStatus](#) AVCommandGetStatus ([AVCommand](#) cmd) ;

Description

Returns the current status of the specified command.

Parameters

cmd	The command whose status is returned. See AVCommand Descriptions (Built-in Commands) .

Return Value

The current status of **cmd**.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



AVCommandGetUIPolicy

```
AVCommandUIPolicy AVCommandGetUIPolicy (AVCommand cmd) ;
```

Description

Retrieves the user interface policy that the command will respect while it is executed. The UI policy is determined by a combination of the command's configuration and the context in which the command is executing (for example, a batch sequence).

Parameters

cmd	The command whose UI policy is returned. See AVCommand Descriptions (Built-in Commands) .

Return Value

The user interface policy.

Header File

AVCalls.h

Related Methods

[AVCommandGetCancelProc](#)
[AVCommandSetConfig](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



AVCommandNew

```
AVCommand AVCommandNew (ASAtom name) ;
```

Description

Creates a new command of the specified type. An [AVCommandHandler](#) for that command type must have already been registered. See [AVCommand Descriptions \(Built-in Commands\)](#) for the names and parameters of built-in commands, as defined in `AVCmdDefs.h`.

Parameters

name	The new command's type name.

Return Value

The new command or **NULL** if the specified type has not been registered.

Header File

`AVCalls.h`

Related Methods

[AVCommandDestroy](#)
[AVAppFindGlobalCommandByName](#)

Product

reader, base, pro

Availability

Available if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

AVCommandPutCab

```
void AVCommandPutCab (AVCommand cmd, const char* key,
    ASCab cabVal);
```

Description

Stores a cabinet in the specified command. If a cabinet is currently stored under **key**, it is overwritten. The command assumes ownership of **cabVal**.

This method can only be exercised by an [AVCommandHandler](#). Command handlers can use this call to store state information (such as parameters) in a command. Only the command handler associated with the command should access these cabinets. See the examples for [AVCommandSetInputs](#) and [AVCommandSetParams](#).

Parameters

cmd	The command. See AVCommand Descriptions (Built-in Commands) .
key	(May be NULL) The key name that the cabinet will be stored under. If NULL , a numerical name is generated and used.
cabVal	(May be NULL) The cabinet to store. If NULL , the method destroys the cabinet currently stored under key (if any).

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

AVCalls.h

Related Methods

[AVCommandGetCab](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



AVCommandReset

[AVCommandStatus](#) AVCommandReset ([AVCommand](#) cmd) ;

Description

Resets the specified command. The command is expected to clean up its current state and assume a ready state.

If the command is currently processing a file, the command might not attempt to roll back any modifications it has made to the PDF. This may leave the PDF in an invalid state.

Parameters

cmd	The command to reset. See AVCommand Descriptions (Built-in Commands) .

Return Value

The current status of **cmd**.

Header File

AVCalls.h

Related Methods

[AVCommandCancel](#)

[AVCommandWork](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVCommandSetConfig

```
AVCommandStatus AVCommandSetConfig (AVCommand cmd,
ASCab config);
```

Description

Sets the configuration for the specified command. See [AVCommandGetConfig](#) for more information.

config will be merged with the current configuration cabinet to generate the new configuraton. To erase configuration settings, set their values to **NULL** in your cabinet.

Currently the only configuration parameters defined are:

- UIPolicy (**kASValueInteger**) - The local UI policy for this command.
- ParamsDescOpen (**kASValueBool**) - Indicates whether the description of this command's parameters is visible in the sequence view.

NOTE: The caller retains ownership of **config**.

Parameters

cmd	The command whose configuration parameters are being set. See AVCommand Descriptions (Built-in Commands) .
config	The configuration settings.

Return Value

The current status of **cmd**. The configuration settings will not be updated if the command is not in a ready state.

Header File

AVCalls.h

Related Methods

[AVCommandGetCancelProc](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVCommandSetInputs

AVCommandStatus AVCommandSetInputs (**AVCommand** cmd,
ASCab inputs) ;

Description

Sets the input parameters of the specified command. The inputs of a command consist of an **ASCab** detailing properties of the object the command will operate on. Use this method and **AVCommandSetParams** to set up a command before calling **AVCommandExecute** to execute it.

A command handler should provide at least the following information.:

Key (see Command Keys)	Stored As	Description
kAVCommandKeyPDDoc	kASValuePointer	The document being processed.
kAVCommandKeyBaseFileName	kASValueString	The name of the input file, without extension.
kAVCommandKeyOrigExtension	kASValueString	The extension of the input file.

The command handler can specify additional inputs as necessary. The caller retains ownership of **inputs**.

NOTE: **AVCommandExecute** will also attempt to provide **kAVCommandKeyAVDoc**, a pointer to the **AVDoc** being operated on.

NOTE: You can use **AVCommandGetPDDoc** and **AVCommandGetAVDoc** to quickly retrieve the **PDDoc** or **AVDoc** in a command's inputs.

Parameters

cmd	The command whose inputs are being set. See AVCommand Descriptions (Built-in Commands) .
inputs	The input parameters.

Return Value

The current status of **cmd**. The input settings will not be updated if the command is not in a ready state.

Header File

AVCalls.h

Related Methods[AVCommandGetInputs](#)[AVCommandSetParams](#)[AVCommandExecute](#)**Example**

```

* ConfigureCommandInputs
** -----
** Sets the inputs for the command.
** Returns the status of the command.
*/
static AVCommandStatus ConfigureCommandInputs (AVCommand cmd,
                                              AVDoc avDoc, PDDoc pdDoc, ASPathName path)
{
    ASCab volatile inputs = NULL;
    AVCommandStatus cmdStatus = kAVCommandInError;
    char *baseFileName = NULL;
    /* Allocate memory for the baseFileName. The ASCab assumes ownership
    of the memory so we don't ASfree it afterwards.*/
    baseFileName = (char *)ASmalloc(sizeof(char) * 16);
    if (!baseFileName)
        return cmdStatus;
    DURING
    /* Store the inputs in the container cab. If we don't pass a base
    filename input, the command uses the PDF filename,
    e.g Doc1.pdf > Doc1_0001.tif */

        inputs = ASCabNew();
        ASCabPutString (inputs, kAVCommandKeyBaseFileName,
                        strcpy(baseFileName, "OutputImg"));
        ASCabPutPointer (inputs, kAVCommandKeyPDDoc, PDDoc, pdDoc,
                        NULL);
        ASCabPutPointer (inputs, kAVCommandKeyAVDoc, AVDoc, avDoc,
                        NULL);
    /* Set the command inputs and destroy the container ASCab.
        cmdStatus = AVCommandSetInputs (cmd, inputs);
    HANDLER
    END_HANDLER

    if (inputs)
        ASCabDestroy (inputs);
    return cmdStatus;
}

```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVCommandSetParams

[AVCommandStatus](#) AVCommandSetParams ([AVCommand](#) cmd,
[ASCab](#) params) ;

Description

Sets the parameters for the specified command. The parameters are those pieces of information that can be controlled by the user, such as the pages to act on. Use this method and [AVCommandSetInputs](#) to set up a command before calling [AVCommandExecute](#) to execute it.

Setting any parameters for a command sets *all* parameters. The command supplies appropriate default parameters for any that are missing from **params**. Passing an empty cabinet causes all parameters to default.

If the command is not in a ready state, the parameters are not updated.

The caller retains ownership of **params**.

Parameters

cmd	The command whose parameters are being set. See AVCommand Descriptions (Built-in Commands) .
params	The parameters to be set.

Return Value

The status of **cmd** after the parameters are set.

Header File

AVCalls.h

Related Methods

[AVCommandGetParams](#)
[AVCommandSetInputs](#)
[AVCommandExecute](#)

Example

```

/* ConfigureCommandParameters
** -----
** Sets all of the command parameters explicitly.
** Returns the status of the command.
**/
static AVCommandStatus ConfigureCommandParameters (AVCommand cmd,
    ASPathName path)
{
    ASCab volatile params = NULL;
    ASPathName volatile parentPath = NULL;
    AVCommandStatus cmdStatus = kAVCommandInError;

    DURING
        parentPath = ASFileSysAcquireParent (NULL, path);

        // Store the destination path in the command parameters cab.
        params = ASCabNew();
        ASCabPutPathName (params, kExtractImgCmdKeyDestDirectory,
            ASGetDefaultFileSys(), parentPath);

        // Set the remaining command parameters.
        ASCabPutInt (params, kExtractImgCmdKeyConvFormat,
            kImgConversionFormatTiff);
        ASCabPutInt (params, kExtractImgCmdKeyResolution,
            kImgResolution600);
        ASCabPutInt (params, kExtractImgCmdKeyColorSpace,
            kColorSpaceGrayscale);
        ASCabPutInt (params, kExtractImgCmdKeyOverwriteFiles, true);
        ASCabPutBool (params, kExtractImgCmdKeyInBatch, false);
        // Mark command as configured.
        ASCabPutBool (params, kExtractImgCmdKeyConfigured, true);
        // Set the command parameters.
        cmdStatus = AVCommandSetParams (cmd, params);
        // Release resources.
        ASCabDestroy (params);
    HANDLER
        if (params)
            ASCabDestroy (params);
        else if (parentPath)
            ASFileSysReleasePath (NULL, parentPath);
    END_HANDLER
    return cmdStatus;
}

```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVCommandShowDialog

[AVCommandStatus](#) AVCommandShowDialog ([AVCommand](#) cmd) ;

Description

Instructs the command to bring up its configuration dialog and gather parameter information from the user. The dialog is shown regardless of the **UIPolicy** of the command.

If the user confirms the configuration dialog, **cmd**'s parameter set is updated, otherwise **cmd** will enter a canceled state.

Query **cmd** for its **kAVCommandKeyCanShowDialog** property using [AVCommandGetProps](#) to determine it has a configuration dialog.

Parameters

cmd	The command for which the dialog is thrown. See AVCommand Descriptions (Built-in Commands) .

Return Value

The current status of **cmd**.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



AVCommandWork

[AVCommandStatus](#) AVCommandWork ([AVCommand](#) cmd) ;

Description

Instructs the command do some work based on its current parameters.

A command will either divide its processing across multiple **AVCommandWork** calls, or perform all the processing within a single call.

If **cmd** is in a working state upon return, it is the client's responsibility to poll the command's "cancel proc" before calling **AVCommandWork** again to continue processing.

Parameters

cmd	The command that is instructed to do some work. See AVCommand Descriptions (Built-in Commands) .

Return Value

The current status of **cmd**. No processing is performed if the command cannot be put in a ready or working state.

Header File

AVCalls.h

Related Methods

[AVCommandCancel](#)
[AVCommandExecute](#)
[AVCommandReset](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVConversion

AVConversionConvertFromPDFWithHandler

```
AVConversionStatus AVConversionConvertFromPDFWithHandler
(AVConversionFromPDFHandler handler, ASCab settings,
AVConversionFlags flags, PDDoc doc, ASPathName path,
ASFileSys fileSys, AVStatusMonitorProcs statusMonitor);
```

Description

Converts a PDF document to another file format using the handler specified.

Parameters

handler	Specifies which conversion handler to use.
settings	ASCab containing the settings to be used in the conversion operation. Pass NULL to use the default settings.
flags	Conversion flags.
doc	PDF document to be converted.
path	The desired location for the output file.
fileSys	The file system from which path was obtained.
statusMonitor	Contains the progress monitor, cancel proc, and error reporting proc to be used by the converter. Can be NULL , or any of its members can be NULL .

Return Value

One of the **AVConversionStatus** codes.

Header File

AVCalls.h

Related Methods

[AVConversionEnumFromPDFConverters](#)
[AVConversionConvertToPDFWithHandler](#)

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVConversionConvertStreamFromPDFWithHandler

AVConversionStatus AVConversionConvertStreamFromPDFWithHandler
(**AVConversionFromPDFHandler** inHandler, **ASCab** inSettings,
AVConversionFlags flags, **PDDoc** inPDDoc, **ASStm** stream,
ASCab metaData, **AVStatusMonitorProcs** statusMonitor);

Description

Converts a PDF document to a stream using the handler specified.

Parameters

inHandler	Specifies which "ConvertFromPDF" handler to use.
inSettings	ASCab containing the settings to be used in the conversion operation. Pass NULL to use the default settings.
flags	Conversion flags.
inPDDoc	PDF document to be converted.
stream	The desired location for the output stream.
metaData	The file metadata.
statusMonitor	Contains the progress monitor, cancel proc, and error reporting proc to be used by the converter. Can be NULL , or any of its members can be NULL .

Return Value

One of the **AVConversionStatus** codes.

Header File

AVCalls.h

Related Methods

[AVConversionConvertFromPDFWithHandler](#)
[AVConversionConvertStreamToPDFWithHandler](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVConversionConvertStreamFromStructNodeWithHandler

```
AVConversionStatus
AVConversionConvertStreamFromStructNodeWithHandler
(AVConversionFromPDFHandler inHandler, ASCab inSettings,
AVConversionFlags flags, AVStructNode inStructNode,
ASStm stream, ASCab metaData,
AVStatusMonitorProcs statusMonitor);
```

Description

Converts a structure node to a stream using the handler specified.

Parameters

inHandler	Specifies which "ConvertFromPDF" handler to use.
inSettings	ASCab containing the settings to be used in the conversion operation. Pass NULL to use the default settings.
flags	Conversion flags.
inStructNode	The structure node to be converted.
stream	The desired location for the output stream.
metaData	The file metadata.
statusMonitor	Contains the progress monitor, cancel proc, and error reporting proc to be used by the converter. Can be NULL, or any of its members can be NULL.

Return Value

One of the AVConversionStatus codes.

Header File

AVCalls.h

Related Methods

AVConversionConvertStreamFromPDFWithHandler

Product

reader, base, pro

Availability

Available if PI_ACROVIEW_VERSION (in PIRequir.h) is set to 0x00060000 or higher.

AVConversionConvertStreamToPDF

```
AVConversionStatus AVConversionConvertStreamToPDF (
    AVConversionFlags flags, const char *mimeType, ASStm stream,
    ASCab metaData, PDDoc* outPDDoc,
    AVStatusMonitorProcs statusMonitor);
```

Description

Converts the specified stream to a PDF document using whatever converter is found. Use this function if you do not know which handler to use.

Multiple conversion handlers can register their services for the same file description, so the first one that matches the file type passed in that has the correct **canDoSync** settings is used.

The converters are enumerated in priority order, starting with the highest priority.

Parameters

flags	Conversion flags.
mimeType	The Mime type.
stream	The stream to convert.
metaData	The metadata.
outPDDoc	<i>(Filled by the method)</i> It is the caller's responsibility to close the document.
statusMonitor	Contains the progress monitor, cancel proc, and error reporting proc to be used by the converter. Can be NULL , or any of its members can be NULL .

Return Value

One of the **AVConversionStatus** codes.

Header File

AVCalls.h

Related Methods

[AVConversionConvertToPDF](#)
[AVConversionConvertStreamToPDFWithHandler](#)

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVConversionConvertStreamToPDFWithHandler

AVConversionStatus AVConversionConvertStreamToPDFWithHandler
(**AVConversionToPDFHandler** inHandler, **ASCab** inSettings,
AVConversionFlags flags, **ASStm** stream, **ASCab** metaData,
PDDoc* outPDDoc, **AVStatusMonitorProcs** statusMonitor);

Description

Converts the specified file to a PDF document using the specified handler.

Parameters

inHandler	Specifies which "ConvertFromPDF" handler to use.
inSettings	ASCab containing the settings to be used in the conversion operation. Pass NULL to use the default settings.
flags	Conversion flags.
stream	The stream to convert.
metaData	The metadata.
outPDDoc	<i>(Filled by the method)</i> It is the caller's responsibility to close the document.
statusMonitor	Contains the progress monitor, cancel proc, and error reporting proc to be used by the converter. Can be NULL , or any of its members can be NULL .

Return Value

One of the **AVConversionStatus** codes.

Header File

AVCalls.h

Related Methods

[AVConversionConvertStreamToPDF](#)
[AVConversionConvertToPDFWithHandler](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVConversionConvertToPDF

```
AVConversionStatus AVConversionConvertToPDF
(AVConversionFlags flags, ASPathName path, ASFileSys fileSys,
PDDoc* doc, AVStatusMonitorProcs statusMonitor);
```

Description

Converts the specified file to a PDF document using whatever converter is found. Use this function if you do not know which handler to use.

Multiple conversion handlers can register their services for the same file description, so the first one that matches the file type passed in that has the correct **canDoSync** settings is used.

The converters are enumerated in priority order, starting with the highest priority.

Parameters

flags	Conversion flags.
path	The location of the input file.
fileSys	The file system from which path was obtained.
doc	<i>(Filled by the method)</i> It is the caller's responsibility to close the document.
statusMonitor	Contains the progress monitor, cancel proc, and error reporting proc to be used by the converter. Can be NULL , or any of its members can be NULL .

Return Value

One of the **AVConversionStatus** codes.

Header File

AVCalls.h

Related Methods

```
AVConversionConvertToPDF
AVConversionConvertToPDFWithHandler
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVConversionConvertToPDFWithHandler

```
AVConversionStatus AVConversionConvertToPDFWithHandler
(AVConversionToPDFHandler handler, ASCab settings,
AVConversionFlags flags, ASPathName path, ASFileSys fileSys,
PDDoc* doc, AVStatusMonitorProcs statusMonitor);
```

Description

Converts a file to a PDF document using the handler specified.

Parameters

handler	Specifies which ConvertToPDF handler to use.
settings	ASCab containing the settings to be used in the conversion operation. Pass NULL to use the default settings.
flags	Conversion flags.
path	The location of the input file.
fileSys	The file system from which path was obtained.
doc	<i>(Filled by the method)</i> It is the caller's responsibility to close the document.
statusMonitor	Contains the progress monitor, cancel proc, and error reporting proc to be used by the converter. Can be NULL , or any of its members can be NULL .

Return Value

One of the **AVConversionStatus** codes.

Header File

AVCalls.h

Related Methods

```
AVConversionEnumToPDFConverters
AVConversionConvertToPDF
AVConversionEnumFromPDFConverters
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVConversionEnumFromPDFConverters

```
void AVConversionEnumFromPDFConverters
(AVConversionFromPDFEnumProc proc,
AVConversionEnumProcData data);
```

Description

Enumerates all registered “ConvertFromPDF” conversion handlers.

Parameters

proc	User-supplied callback.
data	Pointer to user-defined data to pass to proc each time it is called.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVConversionEnumToPDFConverters](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVConversionEnumToPDFConverters

```
void AVConversionEnumToPDFConverters  
(AVConversionToPDFEnumProc proc,  
AVConversionEnumProcData data);
```

Description

Enumerates all registered “ConvertToPDF” conversion handlers.

Parameters

proc	User-supplied callback.
data	Pointer to user-defined data to pass to proc each time it is called.

Return Value

None

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



AVCrypt

AVAuthOpen

```
ASBool AVAuthOpen (PDDoc pdDoc);
```

Description

Determines if the current user is authorized to open a document.

If no password is required to open the document, the user will be granted access automatically. If a password is required, the method queries the user for the password and uses it to request open permissions for the document.

Clients: This method can be used as a standard, interactive authorization callback when opening documents through **PDDocOpen**.

Parameters

pdDoc	The document.

Return Value

true if the user is authorized to open the document, **false** otherwise.

Header File

AVCalls.h

Related Methods

[PDDocAuthorize](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

AVCryptDoStdSecurity

ASBool AVCryptDoStdSecurity (**PDDoc** pdDoc, void* secData);

Description

Displays the settings dialog for the built-in standard security handler, allowing the user to change a document's permissions. The initial configuration of the dialog may be set by populating the **StdSecurityDataRec** structure prior to calling the method.

NOTE: This method does not modify the document in any way. It is present in the API so that other security handlers can use it, if they choose, as a standard way to display/obtain permissions from a user.

Parameters

pdDoc	The document for which new security data is obtained.
secData	<i>(Filled by the method)</i> Pointer to a StdSecurityDataRec structure to hold the permissions selected by the user.

Return Value

Returns **true** if the user confirmed her selections, **false** if she pressed cancel.

Header File

AVCalls.h

Related Methods

PDDocAuthorize

Example

```
StdSecurityDataRec secData;
memset (&secData, 0, sizeof(StdSecurityDataRec));
secData.size = sizeof(StdSecurityDataRec);

if(AVCryptDoStdSecurity (pdDoc, &secData)) {
    /* User confirmed selection */
}
```

Product

reader, base, pro

Availability

Clients: Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVCryptGetPassword

```
ASBool AVCryptGetPassword (PDDoc pdDoc, PDPerms permWanted,
void** authData);
```

Description

Displays a standard dialog box that lets a user enter a password. This method does not attempt to validate the password entered by the user. That operation is performed by [PDDocAuthorize](#).

It is the client's responsibility to release the memory associated with the password using [ASfree](#).

This method is present in the API so that other security handlers can use it, if they choose, as a standard way to obtain a password from a user.

Parameters

pdDoc	The document whose password is obtained from the user. This parameter is used to generate label within the dialog.
permWanted	The permissions requested. Must be an OR of the PDPerms values.
authData	<i>(Allocated and filled by the method)</i> If the method returns true, authData will reference a char* containing the password.

Return Value

Returns **true** if the user confirmed her selection, **false** if the she pressed cancel.

Header File

AVCalls.h

Related Methods

[PDDocAuthorize](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVDoc

AVDocAlert

```
ASInt32 AVDocAlert (AVDoc doc, ASInt32 iconType,
const char* msg, const char* button1, const char* button2,
const char* button3, ASBool beep);
```

Description

Displays an alert containing the specified message, icon, and one to three buttons with the specified titles. See [AAlert](#) for more information.

On the Windows platform, the modal parent for **doc** (as returned by [WinAppGetModalParent](#)) is used as the owner window of the message dialog box. As such this method can be used to display alert dialogs in the context of an external window that is a web browser.

NOTE: The implementations of the **AVDocAlert** methods call [AAlert](#), which is a replaceable method. If [AAlert](#) is replaced, the **AVDocAlert** methods are also affected.

Parameters

doc	(Windows only) The AVDoc whose modal parent is used as the owner window of the message dialog box.
iconType	The icon to display. Must be one of the AAlert Icons . Macintosh users: These constants are defined as per the standard Macintosh user interface guidelines.
msg	The message to display.
button1 , button2 , button3	Titles for up to three buttons. Rules: • Use NULL to suppress a button's display. • At least button1 must be non-NULL. • button3 is not displayed if button2 is NULL.
beep	Pass true to perform a system beep when the alert is shown.

Return Value

The button number (1, 2, or 3) on which the user clicked.

Header File

AVCalls.h

**Related Methods**

[AVAlert](#)
[AVDocAlertConfirm](#)
[AVDocAlertNote](#)
[AVDocAlertYesNo](#)

Product

reader, base, pro

Availability

Available if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.

AVDocAlertConfirm

ASBool AVDocAlertConfirm (**AVDoc** doc, const char* msg);

Description

Displays a dialog box containing the **ALERT CAUTION** icon, the specified message and **OK** and **Cancel** buttons. The method also performs a system beep. See **AVDocAlert** for more information.

Parameters

doc	(Windows only) The AVDoc whose modal parent is used as the owner window of the message dialog box.
msg	The message to display.

Return Value

true if the user clicked **OK**, **false** if the user clicked **Cancel**.

Header File

AVCalls.h

Related Methods

[AVDocAlert](#)
[AVDocAlertNote](#)
[AVDocAlertYesNo](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION**(in **PIRequir.h**) is set to **0x00040000** or higher.

AVDocAlertNote

```
void AVDocAlertNote (AVDoc doc, const char* msg);
```

Description

Displays a dialog box containing the **ALERT NOTE** icon, the specified message and an **OK** button. The method also performs a system beep. See [AVDocAlert](#) for more information.

Parameters

doc	(Windows only) The AVDoc whose modal parent is used as the owner window of the message dialog box.
msg	The message to display.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVDocAlert](#)
[AVDocAlertConfirm](#)
[AVDocAlertYesNo](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

AVDocAlertYesNo

```
ASInt32 AVDocAlertYesNo (AVDoc doc, ASInt32 iconType,
const char* msg, ASBool cancel, ASBool beep);
```

Description

Displays an dialog box containing the specified message, icon, and two or three buttons with the titles **Yes**, **No**, and (optionally) **Cancel**. See [AVDocAlert](#) for more information.

Parameters

doc	(Windows only) The AVDoc whose modal parent is used as the owner window of the message dialog box.
iconType	The icon to display. Must be one of the AAlert Icons . Macintosh users: These constants are defined as per the standard Macintosh user interface guidelines.
msg	The message to display.
cancel	true if cancel button should be provided, false otherwise.
beep	true if it beeps, false otherwise.

Return Value

The button number (1, 2, or 3) on which the user clicked.

Header File

AVCalls.h

Related Methods

[AVDocAlert](#)
[AVDocAlertConfirm](#)
[AVDocAlertNote](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

AVDocBeginUndoOperation

```
void AVDocBeginUndoOperation (AVDoc doc);
```

Description

Begins an undo group for the document. To create an undo group, call this method, create each instance with its handler and data, then call [AVDocEndUndoOperation](#).

All [AVUndo](#) objects that are added to a group are presented in the UI as a single undo operation. Undo groups that are nested in other groups are considered part of the parent group.

Parameters

doc	The document whose undo group is started.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVDocClearUndos](#)
[AVDocEndUndoOperation](#)
[AVDocGetTopUndo](#)
[AVUndoNew](#)

Related Callbacks

[AVUndoBeginEndProc](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVDocClearSelection

```
ASBool AVDocClearSelection (AVDoc doc, ASBool clearHighlight);
```

Description

Clears and destroys the current selection by calling the appropriate selection server's [AVDocSelectionLosingSelectionProc](#).

Parameters

doc	The document whose selection is cleared.
clearHighlight	Pass true , to instruct the Acrobat viewer to remove the selection's highlighting; if it has not already been removed. The selection handler may only mark the highlighted regions of the display invalid, but does not force an immediate redraw of the page view. Use AVPageViewDrawNow to force an immediate redraw if you wish.

Return Value

true if the current selection was cleared, **false** otherwise.

Notifications

Broadcasts [AVDocWillClearSelection](#) if the selection type is not **ASAtomNull**.

Header File

AVCalls.h

Related Methods

[AVDocSetSelection](#)
[AVDocCopySelection](#)
[AVDocEnumSelection](#)
[AVDocDoSelectionProperties](#)
[AVDocGetSelection](#)
[AVDocDeleteSelection](#)

Example

```
PDTextSelect TextSelection;
HiliteEntry Hilite;
PDPPage CurrentPDPPage = PDDocAcquirePage(CurrentPDDoc, PageNum);
    Hilite.offset= (ASUns16) Offset;
    Hilite.length= 0;
    TextSelection = PDTextSelectCreatePageHilite(CurrentPDPPage
        ,&Hilite, 1);
AVDocSetSelection(CurrentAVDoc,
    ASAtomFromString("Text"), (void *)
    TextSelection, true);
if(show)
    AVDocShowSelection(CurrentAVDoc);
else
    AVDocClearSelection(CurrentAVDoc, true);
PDPPageRelease(CurrentPDPPage);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVDocClearUndos

```
void AVDocClearUndos (AVDoc doc);
```

Description

Clears the entire undo list for the **AVDoc** and releases all associated **AVUndo** objects.

Parameters

doc	The document whose undo list is cleared.

Return Value

None.

Header File

AVCalls.h

Related Methods

[AVUndoNew](#)

Related Callbacks

[AVUndoReleaseProc](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



AVDocClose

```
ASBool AVDocClose (AVDoc doc, ASBool noSave);
```

Description

Closes the document window, optionally prompting the user to save the document if it has been modified. When this method closes the **AVDoc**, it also closes the underlying **PDDoc**.

You can replace this method with your own version, using [HFTReplaceEntry](#).

Parameters

doc	The document to close.
noSave	If true , the document is closed without prompting the user and without saving, even if the document has been modified. Because this can cause data loss without user approval, use this feature judiciously. If false , prompts user to save the document if it has been modified.

Return Value

true if the document closed, **false** if it did not (for example, if the user was prompted with the **Save** dialog and chose **Cancel**). The document will always close if **noSave** is **true**.

Notifications

[AVDocWillClose](#)
[AVDocDidClose](#)

Header File

AVCalls.h

Related Methods

[AVDocOpenFromFile](#)
[AVDocOpenFromPDDoc](#)
[AVDocDoSave](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVDocCopyAction

```
PDAction AVDocCopyAction (AVDoc srcDoc, PDAction action,
AVDoc destDoc);
```

Description

Makes a copy of the action for the (possibly different) specified **AVDoc**. Calls the **AVActionCopyProc** callback in **AVActionHandlerProcs**.

NOTE: The Acrobat viewers define **AVActionCopyProc** callbacks for all currently defined actions.

Parameters

srcDoc	The document containing action .
action	The PDAction to copy.
destDoc	The document to which action is copied.

Return Value

A copy of the action.

Known Exceptions

avErrBadActionCopy - the associated action handler has not implemented the **AVActionCopyProc** callback.

pdErrBadAction - the action dictionary is invalid.

pdErrOpNotPermitted - the user does not have the required permissions level for **destDoc** to perform this operation.

Raises the standard exceptions if memory limit problems occur.

Header File

AVCalls.h

Related methods

[AVDocCopyActionCommon](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020003** or higher.



AVDocCopyActionCommon

```
PDAction AVDocCopyActionCommon (AVDoc srcDoc, PDAction action,
AVDoc destDoc);
```

Description

Makes a copy of the action for the (possibly different) specified **AVDoc**.

This copy includes only those fields that are described in the [PDF Reference](#) as common to all actions. (This list currently includes **Type**, **S**, and **Next**.) This method is a proper “starting point” for the design of a Copy method for a custom action. This allows a client to concentrate on those member objects specific to the custom action.

Parameters

srcDoc	The document containing action .
action	The PDAction to copy.
destDoc	The document to which action is copied.

Return Value

A copy of the action.

Known Exceptions

Raises the standard exceptions if memory limit problems occur.

Header File

AVCalls.h

Related methods

[AVDocCopyAction](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020003** or higher.

AVDocCopyAdditionalActions

```
void AVDocCopyAdditionalActions (AVDoc srcDoc, CosObj srcDict,
AVDoc destDoc, CosObj destDict);
```

Description

Copies any additional actions (**AA**) from a Cos Dictionary to a Cos Dictionary in the (possibly different) specified **AVDoc**.

This method copies the following keys: **E, X, D, U, O, C, FP, PP, NP**, and **LP**. This method is designed to aid clients in copying additional actions for pages.

For copying annotations, it is better to use [AVDocCopyAnnotCommon](#).

Parameters

srcDoc	The document containing srcDict.
srcDict	The dictionary from which the action is copied.
destDoc	The document to which the action is copied.
destDict	The dictionary to which the action is copied.

Return Value

A copy of the action.

Known Exceptions

[cosErrExpectedDict](#) - the object stored under the /AA key in **srcDict** is not a CosDict.

Raises the standard exceptions if memory limit problems occur.

Header File

AVCalls.h

Related methods

[AVDocCopyAction](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020003** or higher.



AVDocCopyAnnot

```
PDAnnot AVDocCopyAnnot (AVDoc srcDoc, PDAnnot annot,
AVDoc destDoc);
```

Description

Makes a copy of the annotation for the (possibly different) specified **AVDoc**. Calls the **AVAnnotHandlerCopyProc** callback in **AVAnnotHandler**.

NOTE: The Acrobat viewers define **AVAnnotHandlerCopyProc** callbacks for all currently defined annotations.

Parameters

srcDoc	The document containing annot .
annot	The annotation to copy.
destDoc	The document to which annot is copied.

Return Value

A copy of the annotation.

Known Exceptions

avErrBadAnnotationCopy - the associated annotation handler has not implemented the **AVAnnotHandlerCopyProc** callback.

Raises the standard exceptions if memory limit problems occur.

Header File

AVCalls.h

Related methods

[AVDocCopyAnnotCommon](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020003** or higher.



AVDocCopyAnnotCommon

PDAnnot AVDocCopyAnnotCommon (**AVDoc** fromDoc, **PDAnnot** anAnnot, **AVDoc** toDoc) ;

Description

Makes a copy of the annotation for the (possibly different) specified **AVDoc**.

This copy includes only those fields that are described in the [PDF Reference](#) as common to all annotations. (This list currently includes **Type**, **Subtype**, **Rect**, **Border**, **C**, **T**, **M**, **F**, **H**, **AS**, **BS**, and **AA**.) This method is a proper “starting point” for the design of a Copy method for a custom annotation. This allows a client to concentrate on those member objects specific to the custom annotation.

Parameters

fromDoc	The document whose annotation is copied.
anAnnot	The annotation to copy.
toDoc	The document to which the annotation is copied.

Return Value

A copy of the annotation.

Known Exceptions

Raises the standard exceptions if memory limit problems occur.

Header File

AVCalls.h

Related methods

[AVDocCopyAnnot](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020003** or higher.

AVDocCopySelection

```
void AVDocCopySelection (AVDoc doc);
```

Description

Copies the current selection to the clipboard, if possible. The selection is copied if the selection server has a [AVDocSelectionCopyProc](#) callback, and the selection server's [AVDocSelectionCanCopyProc](#) callback returns **true**. If the selection server does not have a [AVDocSelectionCanCopyProc](#) method, a default value of **true** is used.

The selection is copied by calling the selection server's [AVDocSelectionCopyProc](#) callback.

Parameters

doc	The document whose selection is copied.

Return Value

None

Known Exceptions

Only those raised by the selection server's [AVDocSelectionCopyProc](#) and [AVDocSelectionCanCopyProc](#) callbacks.

Header File

AVCalls.h

Related methods

- [AVDocRegisterSelectionServer](#)
- [AVDocSetSelection](#)
- [AVDocDeleteSelection](#)
- [AVDocClearSelection](#)
- [AVDocGetSelectionType](#)
- [AVDocEnumSelection](#)



Example

```
PDTextSelect TextSelection;
HiliteEntry Hilite;
PDPage CurrentPDPage =
    PDDocAcquirePage(CurrentPDDoc, PageNum);
Hilite.offset = (ASUns16) Offset;
Hilite.length = 0;
TextSelection =
    PDTextSelectCreatePageHilite(
        CurrentPDPage, &Hilite, 1);
AVDocSetSelection(CurrentAVDoc,
    ASAtomFromString("Text"), (void *)
    TextSelection, true);
AVDocCopySelection(doc);
AVDocDeleteSelection(doc);
PDPageRelease(CurrentPDPage);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVDocDeleteSelection

```
ASBool AVDocDeleteSelection (AVDoc doc);
```

Description

Deletes the specified document's current selection, if possible. The selection is deleted if changing the selection is currently permitted, the selection server has an [AVDocSelectionDeleteProc](#) callback, and the selection server's [AVDocSelectionCanDeleteProc](#) callback returns **true**. If the selection server does not have a [AVDocSelectionCanDeleteProc](#) callback, a default value of **true** is used.

The selection is deleted by calling the selection server's [AVDocSelectionDeleteProc](#) callback.

Parameters

doc	The document whose selection is deleted.

Return Value

true if the current selection was actually deleted, **false** otherwise.

Known Exceptions

Only those raised by the selection server's [AVDocSelectionDeleteProc](#) and [AVDocSelectionCanDeleteProc](#) callbacks.

Notifications

[AVDocDidDeleteSelection](#)

Header File

AVCalls.h

Related methods

[AVDocRegisterSelectionServer](#)
[AVDocGetSelection](#)
[AVDocSetSelection](#)
[AVDocClearSelection](#)
[AVDocCopySelection](#)
[AVDocEnumSelection](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVDocDoActionPropsDialog

```
ASBool AVDocDoActionPropsDialog (AVDoc doc, PDAction* action,  
const char* dialogTitle);
```

Description

Displays a modal dialog that allows a user to specify an action. Used, for example, by forms to add actions to fields.

Parameters

doc	The document in which the action is specified.
action	The specified action.
dialogTitle	Title for the dialog.

Return Value

true if the user clicked the **Set Link** button, **false** if the user clicked **Cancel**.

Header File

AVCalls.h

Related Methods

[AVDocPerformAction](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

AVDocDoCopyAs

```
ASBool AVDocDoCopyAs (AVDoc doc) ;
```

Description

Prompts the user with a standard file dialog and copies the file `doc` byte for byte. Displays a progress monitor showing the progress of the file copy.

Parameters

<code>doc</code>	The document to copy.

Return Value

`true` if the copy was successful, `false` otherwise.

Header File

AVCalls.h

Related Methods

[AVDocDoPrint](#)
[AVDocDoSave](#)
[AVDocDoSaveAs](#)

Product

reader, base, pro

Availability

Available if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.



AVDocDoPrint

```
void AVDocDoPrint (AVDoc doc);
```

Description

Performs the print operation, including user dialogs.

You can replace this method with your own version, using [HFTReplaceEntry](#).

Parameters

doc	The document to print.

Return Value

None

Notifications

[AVDocDidPrint](#)
[AVDocWillPrint](#)

Header File

AVCalls.h

Related Methods

[AVDocDoCopyAs](#)
[AVDocDoSave](#)
[AVDocPrintPages](#)
[AVDocPrintPagesWithParams](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.



AVDocDoSave

```
ASBool AVDocDoSave (AVDoc doc);
```

Description

Saves a file, including handling any user interface (for example, a **Save File** dialog) that is needed. Clients do not need to call this method directly, but it is available for clients that need to override the Acrobat viewer's built-in save method, for example to save the changes made to a PDF file into a special file, but not save the original PDF file.

You can replace this method with your own version, using [HFTReplaceEntry](#).

Parameters

doc	The document to save.

Return Value

true if the document was successfully saved, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVDocDoCopyAs](#)
[AVDocDoPrint](#)
[AVDocDoSaveAs](#)
[AVDocDoSaveAsWithParams](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVDocDoSaveAs

```
ASBool AVDocDoSaveAs (AVDoc doc) ;
```

Description

Displays a file dialog and saves the document to a potentially new name. Allows clients (such as Optimizer) to do their own file saving.

You can replace this method with your own version, using [HFTReplaceEntry](#).

Parameters

doc	The document to save.

Return Value

true if the document was successfully saved, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVDocDoCopyAs](#)
[AVDocDoPrint](#)
[AVDocDoSave](#)
[AVDocDoSaveAsWithParams](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

AVDocDoSaveAsWithParams

```
ASBool AVDocDoSaveAsWithParams (AVDoc doc,
AVDocSaveParams params);
```

Description

Saves a file, using the parameters specified in **paramsIn**.

You can replace this method with your own version, using [HFTReplaceEntry](#).

Parameters

doc	The document to save.
params	A structure with information telling how the AVDoc is saved.

Return Value

true if the document was successfully saved, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVDocDoCopyAs](#)
[AVDocDoPrint](#)
[AVDocDoSave](#)
[AVDocDoSaveAs](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

AVDocDoSelectionProperties

```
void AVDocDoSelectionProperties (AVDoc doc);
```

Description

Displays the user interface, if any, for setting the current selection's properties. It does this by invoking the [AVDocSelectionPropertiesProc](#) callback, if any, of the current selection's selection server.

Parameters

doc	The document containing the selection whose properties are set.

Return Value

None

Known Exceptions

Only those raised by the selection server's [AVDocSelectionPropertiesProc](#) callback.

Header File

AVCalls.h

Related Methods

[AVDocRegisterSelectionServer](#)
[AVDocSetSelection](#)

Example

```
if (PtInRect(&myRect, point))
    AVDocDoSelectionProperties(doc);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVDocEndUndoOperation

```
void AVDocEndUndoOperation (AVDoc doc, onst AText undoTitle,
const AText redoTitle);
```

Description

Ends an undo group for the document. To create an undo group, call [AVDocBeginUndoOperation](#), create each instance with its handler and data, then call this method.

Parameters

doc	The document whose undo group is ended.
undoTitle	The title to be used as the "Undo" string in the UI.
redoTitle	The title to be used as the "Redo" string in the UI.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVDocBeginUndoOperation](#)
[AVDocClearUndos](#)
[AVDocGetTopUndo](#)
[AVUndoNew](#)

Related Callbacks

[AVUndoBeginEndProc](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVDocEnumSelection

```
void AVDocEnumSelection (AVDoc doc, AVSelectionEnumProc proc,
void* clientData);
```

Description

Enumerates the elements of the current selection by calling the current selection server's [AVDocSelectionEnumSelectionProc](#) callback. If the selection server does not have an [AVDocSelectionEnumSelectionProc](#), calls **proc** and passes the entire selection to it in the **aSelectedObject** parameter.

Parameters

doc	The document whose selection is enumerated.
proc	User-supplied callback to call for each element in the selection. Enumeration ends if proc returns false .
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Known Exceptions

Only those raised by the selection server's [AVDocSelectionEnumSelectionProc](#) callback, and those raised by **proc**.

Header File

AVCalls.h

Related Methods

[AVDocRegisterSelectionServer](#)
[AVDocSetSelection](#)
[AVDocClearSelection](#)
[AVDocDeleteSelection](#)
[AVDocCopySelection](#)
[AVDocSelectionEnumPageRanges](#)

Example

```
static ACCB1 ASBool ACCB2 mySelectEnum(
    AVDoc doc, void* clientData,
    void* selectedObject)
{
    ASInt32 page = PDTextSelectGetPage(
        (PDTextSelect) selectedObject);
    return false;
}

AVDocEnumSelection(doc,
    ASCallbackCreateProto(
        AVSelectionEnumProc, &mySelectEnum),
    NULL);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVDocFromPDDoc

```
AVDoc AVDocFromPDDoc (PDDoc pdDoc) ;
```

Description

Gets the **AVDoc** associated with a **PDDoc**.

Parameters

pdDoc	The PDDoc whose AVDoc is to be returned.

Return Value

If an **AVDoc** is already opened for this **PDDoc**, returns the **AVDoc**. Otherwise returns **NULL**.

Header File

AVCalls.h

Related Methods

[AVDocGetPDDoc](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVDocGetActiveTool

```
AVTool AVDocGetActiveTool (AVDoc avDoc);
```

Description

Gets the active tool for the **AVDoc**. The call provides the additional context needed to retrieve the active tool for documents that are displayed in external windows such as a web browser.

NOTE: It is recommended that you use **AVDocGetActiveTool** rather than **AVAppGetActiveTool**, preferably specifying a particular document.

Parameters

avDoc	The document whose active tool is obtained, or NULL to get the active tool for the main application window.

Return Value

The active tool object.

Known Exceptions

genErrNoMemory

Header File

AVCalls.h

Related Methods

AVAppGetActiveTool
AVAppGetDefaultTool
AVAppGetLastActiveTool
AVDocSetActiveTool

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVDocGetAVWindow

```
AVWindow AVDocGetAVWindow (AVDoc doc) ;
```

Description

Gets the **AVWindow** in which the document is displayed.

Parameters

doc	The document whose AVWindow is obtained.

Return Value

The document's **AVWindow**.

Header File

AVCalls.h

Related Methods

[AVDocGetPageView](#)

Example

```
AVWindow myWin = AVDocGetAVWindow(doc) ;
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVDocGetClientName

```
AVTArraySize AVDocGetClientName (AVDoc avDoc, char* buffer,
AVTArraySize bufSize);
```

Description

Gets the **AVDoc** client (container application) name.

This method can be used by a client to determine which client application caused the Acrobat viewer to open a document.

Parameters

avDoc	The document whose client name is obtained.
buffer	<i>(Filled by the method)</i> The buffer into which the client name is written. May be up to 255 characters. The client name is null-terminated. If the client name is longer than bufSize , the first bufSize – 1 bytes are copied into it, followed by a NULL .
bufSize	The size of buffer , in bytes.

Return Value

The number of characters written into **buffer**, excluding the **NULL** termination. Returns 0 if the specified document does not have a client associated with it.

Header File

AVCalls.h

Related Methods

[AVDocSetClientName](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020001** or higher.



AVDocGetNthPageView

```
AVPageView AVDocGetNthPageView (AVDoc doc, ASCount n);
```

Description

Gets the specified **AVPageView** for the specified document.

Parameters

doc	The document whose AVPageView is obtained.
n	The index of the page view to obtain. The index range is 0 to (AVDocGetNumPageViews - 1).

Return Value

The document's nth **AVPageView**.

Header File

AVCalls.h

Related Methods

[AVDocGetNumPageViews](#)

[AVDocGetPageView](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



AVDocGetNumPageViews

```
ASCCount AVDocGetNumPageViews (AVDoc doc);
```

Description

Gets the number of page views for the specified document.

Parameters

doc	The document whose page view count is obtained.

Return Value

The number of **AVPageView** objects associated with the document as an **ASCCount**.

Header File

AVCalls.h

Related Methods

[AVDocGetNthPageView](#)

[AVDocGetPageView](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVDocGetPageText

```
void AVDocGetPageText (AVDoc doc, PDPageNumber pageNum,
PDTextSelect pdText, ASAtom format, AVTextCopyProc copyProc,
void* clientData);
```

Description

Gets the text from the specified text selection, converts it to the specified format, and passes it to a user-supplied procedure.

Parameters

doc	The document from which text is obtained.
pageNum	The page number in doc from which text is obtained. The first page in a document is page 0.
pdText	The text selection whose text is obtained. Passes NULL to get all the text on the page.
format	The format in which text is desired. The following are allowed values (these strings must be converted to ASAtoms using ASAtomFromString): All — (Mac OS/Windows) Calls copyProc once with each available format. Text — (Mac OS/Windows) Calls copyProc twice. The first time, it passes the text in the specified selection. The text is passed using the same platform encoding as when it is put onto the clipboard (format = Text). The second time, it passes a Unicode version of the text (format = UCSText). Style — (Mac OS only) Calls copyProc twice. The first time, it passes the Text representation. The second time, it passes a StyleInfo record. RTF — (Mac OS/Windows) Calls copyProc twice. The first time, it passes the text in Rich Text Format. The second time, it passes a Unicode version of the text. Upon using/manipulating these texts, you should check the format of these texts in copyProc.
copyProc	User-supplied callback to which the text is passed.
clientData	Pointer to user-supplied data to pass to copyProc each time it is called.

Return Value

None

**Known Exceptions**

[pdErrOpNotPermitted](#)

Header File

AVCalls.h

Related Methods

[PDTextSelectCreateRanges](#)

[PDTextSelectCreatePageHilite](#)

[PDTextSelectCreateWordHilite](#)

[PDDocCreateTextSelect](#)

[PDWordFinderEnumWords](#)

[PDWordFinderAcquireWordList](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020001** or higher.

AVDocGetPageView

```
AVPageView AVDocGetPageView (AVDoc doc);
```

Description

Gets the **AVPageView** for the specified document.

Parameters

doc	The document whose AVPageView is obtained.

Return Value

The document's **AVPageView**.

Header File

AVCalls.h

Related Methods

[AVDocGetAVWindow](#)

[AVDocGetViewMode](#)

Example

```
AVPageView avPageView =
    AVDocGetPageView(AVAppGetActiveDoc());
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVDocGetPDDoc

```
PDDoc AVDocGetPDDoc (AVDoc avDoc);
```

Description

Gets the **PDDoc** to associate with the specified **AVDoc**.

Parameters

avDoc	The document whose PDDoc is obtained.

Return Value

The **PDDoc** associated with **avDoc**.

Header File

AVCalls.h

Related Methods

[AVDocOpenFromPDDoc](#)

[PDEnumDocs](#)

Example

```
ACCB1 ASBool ACCB2 myAppEnumProc(
    AVDoc doc, void* clientData)
{
    if (PDDocGetNumPages (AVDocGetPDDoc (doc) )
        > 100)
        AAlertNote ("Long document");
    return true;
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVDocGetSelection

```
void* AVDocGetSelection (AVDoc doc);
```

Description

Gets the current selection for the specified document.

Parameters

doc	The document whose selection is obtained.

Return Value

The current selection, or **NULL** if there is no selection. See [Selection Types](#) for a list of the data types returned for the built-in selection types. A **NULL** return value from this method is not sufficient to determine that there is no selection, use [AVDocGetSelectionType](#) instead.

Header File

AVCalls.h

Related Methods

[AVDocSetSelection](#)
[AVDocGetSelectionType](#)
[AVDocClearSelection](#)
[AVDocDeleteSelection](#)
[AVDocEnumSelection](#)
[AVDocCopySelection](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVDocGetSelectionServerByType

[AVDocSelectionServer](#) AVDocGetSelectionServerByType
([ASAtom](#) type);

Description

Gets the selection server that handles the specified selection type.

Parameters

type	The ASAtom corresponding to the type for which the selection server was registered. The string for type can be converted to an ASAtom using ASAtomFromString .

Return Value

The selection server that handles the specified type. Returns **NULL** if no such selection server is registered.

Header File

AVCalls.h

Related Methods

[AVDocRegisterSelectionServer](#)

Example

```
AVDocSelectionServer server;
if (server =
    AVDocGetSelectionServerByType (
        ASAtomFromString("imageSelect"))
{
    AVDocClearSelection(doc, true);
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVDocGetSelectionType

```
ASAtom AVDocGetSelectionType (AVDoc doc);
```

Description

Gets the current selection's type for the specified document.

NOTE: To get information about the selected annotation, use [AVPageViewGetFocusAnnot](#).

Parameters

<code>doc</code>	The document whose selection type is obtained.

Return Value

The **ASAtom** corresponding to the current selection type. Returns **ASAtomNull** if there is no selection. The **ASAtom** returned can be converted to a string using [ASAtomGetString](#). See [Selection Types](#) for a list of the built-in selection types.

Header File

AVCalls.h

Related Methods

[AVDocGetSelection](#)
[AVDocSetSelection](#)

Example

```
PDTextSelect ts;

if (ASAtomFromString("Text") ==
    AVDocGetSelectionType(doc))
{
    ts = AVDocGetSelection(doc);
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVDocGetServerType

```
AVDocServerType AVDocGetServerType (AVDoc doc) ;
```

Description

Gets the server type for the specified document.

Parameters

<code>doc</code>	The document whose server type is obtained.

Return Value

The server type value.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVDocGetSplitterPosition

```
ASInt16 AVDocGetSplitterPosition (AVDoc doc);
```

Description

Gets the splitter position. The splitter is the vertical division between the bookmark/thumbnail pane and the document pane. The default splitter location is saved in the Acrobat viewer's preferences file, and can be read/set using [AVAppGetPreference](#) and [AVAppSetPreference](#).

Parameters

doc	The document whose splitter position is obtained.

Return Value

The width of the bookmark/thumbnail pane, measured in pixels. Returns 0 if the bookmark/thumbnail pane is not currently displayed.

Header File

AVCalls.h

Related Methods

[AVDocSetSplitterPosition](#)
[AVAppGetPreference](#)

Example

```
if (AVDocGetSplitterPosition(doc) < 72)
    AVDocSetSplitterPosition(doc, 72);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVDocGetTopUndo

[AVUndo](#) AVDocGetTopUndo ([AVDoc](#) doc, const char *desiredType);

Description

Returns the most recent [AVUndo](#) record in the document's undo list, if it of the desired type.

Parameters

<code>doc</code>	The document.
<code>desiredType</code>	The type of the desired undo record.

Return Value

The undo record object, or **NULL** if the undo list is empty or if the type of the most recent undo record does not match **desiredType**.

Header File

AVCalls.h

Related Methods

[AVDocBeginUndoOperation](#)
[AVDocClearUndos](#)
[AVDocGetTopUndoAndRedo](#)
[AVUndoGetType](#)
[AVUndoNew](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVDocGetTopUndoAndRedo

```
void AVDocGetTopUndoAndRedo (AVDoc doc, AVUndo *undo,
                             AVUndo *redo, const char *desiredType);
```

Description

Returns the most recent undo and redo records in the document’s undo list, if they are of the desired type.

Parameters

doc	The document.
undo	<i>(Filled by the method)</i> A pointer to the latest undo, or NULL if the undo list is empty or if the type of the most recent undo record does not match desiredType .
redo	<i>(Filled by the method)</i> A pointer to the latest redo, or NULL if the redo list is empty or if the type of the most recent redo record does not match desiredType .
desiredType	The type of the desired undo record.

Return Value

None

Header File

AVCalls.h

Related Methods

- AVDocBeginUndoOperation
- AVDocClearUndos
- AVDocGetTopUndo
- AVUndoGetType
- AVUndoNew

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



AVDocGetViewDef

```
void AVDocGetViewDef (AVDoc doc, AVDocViewDef viewDef);
```

Description

NOTE: Numeric types in [AVDocViewDef](#) have changed in Acrobat 6.0, and the method has been superseded by [AVDocGetViewDefEx](#).

Fills out the given [AVDocViewDef](#) structure with the information needed to restore this document's state at a later date. You must set the "use" fields as desired.

Parameters

doc	The document whose state is recorded.
viewDef	A pointer to the structure that stores the state.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVDocGetViewDefEx](#)
[AVDocSetViewDef](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

AVDocGetViewDefEx

```
void AVDocGetViewDefEx (AVDoc doc, ASCab viewDef);
```

Description

Fills out the given view definition with the information needed to restore this document's state at a later date. If you pass an empty **ASCab**, the method fills it with all of the view information. Otherwise, it fills only those fields that are present in the **ASCab**.

The **ASCab** can contain the same fields defined for **AVDocViewDef**, plus an additional field, **ocgStates**, which is an **ASCab** containing a set of optional content states.

NOTE: Supersedes **AVDocGetViewDef** in Acrobat 6.0.

Parameters

doc	The document whose state is recorded.
viewDef	A pointer to the ASCab that stores the state.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVDocGetViewDef](#)
[AVDocSetViewDefEx](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVDocGetViewMode

```
PDPageMode AVDocGetViewMode (AVDoc doc);
```

Description

Gets the current view mode.

Parameters

doc	The document whose view mode is obtained.

Return Value

The current view mode.

Header File

AVCalls.h

Related Methods

[AVDocSetViewMode](#)

[PDDocGetPageMode](#)

Example

```
if (AVDocGetViewMode(doc) == PDUseNone)
    AVDocSetViewMode(doc, PDUseBookmarks);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVDocIsDead

```
ASBool AVDocIsDead (AVDoc avDoc);
```

Description

Determines whether a given document is “dead.” When the connection to a document is severed (for example, when its HTTP connection is broken) the document becomes “dead” for an interval before it is closed. During that interval, the document may be visible and open but no operations should be performed on the document.

Parameters

avDoc	The document being investigated.

Return Value

true if the given document is dead; **false** otherwise.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



AVDocIsExternal

```
ASBool AVDocIsExternal (AVDoc doc);
```

Description

Determines whether a given document is displayed in an application's external window (such as in a Web browser's window).

Parameters

<code>doc</code>	The document being tested.

Return Value

true if the given document is displayed in an application's external window, **false** otherwise.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.



AVDocIsReadOnly

```
ASBool AVDocIsReadOnly (AVDoc doc);
```

Description

Checks to see if an **AVDoc** is read-only.

Parameters

doc	The AVDoc whose read-only state is checked.

Return Value

true if the given document is read-only, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVDocSetReadOnly](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.



AVDocIsSlow

```
ASBool AVDocIsSlow (AVDoc avDoc);
```

Description

Determines whether a given document is on a slow file system (such as a Web browser).

Parameters

avDoc	The document being investigated.

Return Value

true if the given document is on a slow file system; **false** otherwise.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVDocOpenFromASFileWithParams

```
AVDoc AVDocOpenFromASFileWithParams (ASFile file, const
ASText tempTitle, AVDocOpenParams params);
```

Description

Opens and displays a document from a file, using the specified parameters to control the window's size, location, and visibility.

You can replace this method with your own version, using [HFTReplaceEntry](#).

If you want to display a page from a PDF file as a bitmap, use [PDPPageDrawContentsToWindow](#).

NOTE: Do not open and then immediately close an **AVDoc** without letting at least one event loop complete.

Parameters

file	The ASFile to open.
tempTitle	If tempTitle != NULL, pathName is a temporary file and tempTitle is used as the window's title.
params	Parameters used when opening the file. Can be NULL .

Return Value

The document that was opened.

Notifications

[AVDocWillOpenFromFile](#)
[AVDocDidOpen](#)
[AVAppFrontDocDidChange](#)
[AVDocDidActivate](#)
[AVDocDidDeactivate](#)

The following notifications are broadcast if the document has a valid open action:

[AVDocWillPerformAction](#)
[AVDocDidPerformAction](#)

Header File

AVCalls.h

Related Methods

[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromFile](#)
[AVDocOpenFromPDDoc](#)

AVDocOpenFromPDDocWithParams

AVDocClose

Example

```
AVDocOpenParamsRec params;
AVDoc myDoc;

params.size = sizeof(AVDocOpenParamsRec);
params.useFrame = false;
params.useVisible = true;
params.visible = false;
params.useServerType = false;
params.useReadOnly = true;
params.readOnly = true;
params.useViewType = true;
params.viewType = "AVPageView";
params.usePermReqProc = true;
params.permReqProc = ASCallbackCreateProto(AVDocPermReqProc,
&myPermReq);
myDoc =
    AVDocOpenFromFileWithParams(myPathName,
                                NULL, NULL, &params);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

Changed from **char*** to **ASText** argument type in **0x00060000**.

AVDocOpenFromASFileWithParamString

```
AVDoc AVDocOpenFromASFileWithParamString (ASFile file,
const ASText tempTitle, AVDocOpenParams params,
const char* s);
```

Description

Opens and displays a document from an **ASFile**, using the specified parameters to control the window's size, location, and visibility, and passing a URL open action string to control how the file is opened. For more information, see the document *URL Open Actions*.

If you want to display a page from a PDF file as a bitmap, use [PDPageDrawContentsToWindow](#).

Parameters

file	The ASFile to open.
tempTitle	An optional window title for the document.
params	Open parameters for the document.
s	A string containing the URL open action. For example: Help=contents&nameddest=distiller_mydest This would open the PDF in the help window, with the contents panel showing on the left, and the page referenced by the named destination distiller_mydest showing on the right.

Return Value

An **AVDoc** object for **file**.

Header File

AVCalls.h

Notifications

[AVDocWillOpenFromFile](#)
[AVDocDidOpen](#)
[AVAppFrontDocDidChange](#)
[AVDocDidActivate](#)
[AVDocDidDeactivate](#)

The following notifications are broadcast if the document has a valid open action:

[AVDocWillPerformAction](#)
[AVDocDidPerformAction](#)

Related Methods

[AVDocOpenFromASFileWithParams](#)

AVDocOpenFromFileWithParamString
AVDocOpenFromPDDocWithParamString

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVDocOpenFromFile

```
AVDoc AVDocOpenFromFile (ASPathName pathName,
ASFileSys fileSys, const AText tempTitle);
```

Description

Opens and displays a document from a file. This is equivalent to [AVDocOpenFromASFileWithParams\(pathName, fileSys, tempTitle, NULL\)](#). If you want to display a page from a PDF file as a bitmap, use [PDPageDrawContentsToWindow](#).

NOTE: Do not open and then immediately close an **AVDoc** without letting at least one event loop complete.

Parameters

pathName	The file to open.
fileSys	The file system on which pathName resides. You can obtain the default file system with ASGetDefaultFileSys .
tempTitle	If tempTitle != NULL , pathname is a temporary file, and tempTitle is used as the window's title.

Return Value

The document that was opened. Returns **NULL** if the viewer failed to open the document.

Notifications

[AVDocWillOpenFromFile](#)
[AVDocDidDeleteSelection](#)
[AVAppFrontDocDidChange](#)
[AVDocDidActivate](#)
[AVDocDidDeactivate](#)

The following notifications are broadcast if the document has a valid open action:

[AVDocWillPerformAction](#)
[AVDocDidPerformAction](#)

Header File

AVCalls.h

Related Methods

[AVDocOpenFromASFileWithParams](#)
[AVDocOpenFromPDDoc](#)
[AVDocOpenFromPDDocWithParams](#)
[AVDocClose](#)

**Example**

```
AVDoc myDoc = AVDocOpenFromFile(  
    myPathName, ASGetDefaultFileSys(),  
    NULL);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

Changed from **char*** to **ASText** argument type in **0x00060000**.



AVDocOpenFromFileWithParams

```
AVDoc AVDocOpenFromFileWithParams (ASPathName pathName,
ASFileSys fileSys, const AText tempTitle,
AVDocOpenParams params);
```

Description

Opens and displays a document from a file, using the specified parameters to control the window's size, location, and visibility.

You can replace this method with your own version, using [HFTReplaceEntry](#).

If you want to display a page from a PDF file as a bitmap, use [PDPageDrawContentsToWindow](#).

NOTE: Do not open and then immediately close an **AVDoc** without letting at least one event loop complete.

Parameters

pathName	The file to open.
fileSys	The file system on which pathName resides. You can obtain the default file system with ASGetDefaultFileSys .
tempTitle	If tempTitle != NULL, pathName is a temporary file and tempTitle is used as the window's title.
params	Parameters used when opening the file. Can be NULL .

Return Value

The document that was opened.

Notifications

[AVDocWillOpenFromFile](#)
[AVDocDidOpen](#)
[AVAppFrontDocDidChange](#)
[AVDocDidActivate](#)
[AVDocDidDeactivate](#)

The following notifications are broadcast if the document has a valid open action:

[AVDocWillPerformAction](#)
[AVDocDidPerformAction](#)

Header File

AVCalls.h

Related Methods

[AVDocOpenFromASFileWithParams](#)
[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromFile](#)
[AVDocOpenFromFileWithParamString](#)
[AVDocOpenFromPDDoc](#)
[AVDocClose](#)

Product

reader, base, pro

Availability

Available if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

Changed from `char*` to [ASText](#) argument type in `0x00060000`.

AVDocOpenFromFileWithParamString

```
AVDoc AVDocOpenFromFileWithParamString (ASPathName pathName,
ASFileSys fileSys, const AText tempTitle,
AVDocOpenParams params, const char* s);
```

Description

Opens and displays a document from a file, using the specified parameters to control the window's size, location, and visibility, and passing a URL open action string to control how the file is opened. For more information, see the document *URL Open Actions*.

You can replace this method with your own version, using [HFTReplaceEntry](#).

If you want to display a page from a PDF file as a bitmap, use [PDPPageDrawContentsToWindow](#).

NOTE: Do not open and then immediately close an **AVDoc** without letting at least one event loop complete.

Parameters

pathName	The file to open.
fileSys	The file system on which pathName resides. You can obtain the default file system with ASGetDefaultFileSys .
tempTitle	If tempTitle != NULL , pathName is a temporary file and tempTitle is used as the window's title.
params	Parameters used when opening the file. Can be NULL .
s	A string containing the URL open action. For example: Help=contents&nameddest=distiller_mydest This would open the PDF in the help window, with the contents panel showing on the left, and the page referenced by the named destination distiller_mydest showing on the right.

Return Value

The document that was opened.

Notifications

[AVDocWillOpenFromFile](#)
[AVDocDidOpen](#)
[AVAppFrontDocDidChange](#)
[AVDocDidActivate](#)
[AVDocDidDeactivate](#)

The following notifications are broadcast if the document has a valid open action:

[AVDocWillPerformAction](#)
[AVDocDidPerformAction](#)

Header File

AVCalls.h

Related Methods

[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromFileWithParams](#)
[AVDocOpenFromFile](#)
[AVDocOpenFromPDDoc](#)
[AVDocClose](#)

Product

reader, base, pro

Availability

Available if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

AVDocOpenFromPDDoc

```
AVDoc AVDocOpenFromPDDoc (PDDoc doc, const AText tempTitle);
```

Description

Opens and returns an **AVDoc** view of **pdDoc**. In addition, acquires **pdDoc**. This method is equivalent to **AVDocOpenFromPDDocWithParams** (**pdDoc**, **tempTitle**, **NULL**). If you want to display a page from a PDF file as a bitmap, use **PDPageDrawContentsToWindow**.

NOTE: Do not open and then immediately close an **AVDoc** without letting at least one event loop complete.

NOTE: **AVDocClose** should be used in place of **PDDocClose** after **AVDocOpenFromPDDoc** is called. **AVDocClose** will decrement the **pdDoc** appropriately and free document-related resources.

Parameters

doc	The document to open.
tempTitle	If tempTitle != NULL , pathname is a temporary file and tempTitle is used as the window's title.

Return Value

NULL if failure occurs.

Known Exceptions

Raises **genErrGeneral** if **pdDoc** is **NULL** or has 0 pages. Raises **pdErrBadAction** if the document's open action is recursive. Raises **avErrCantOpenMoreThanTenDocs** if the maximum number of documents is already open. Raises **genErrNoMemory** if there is insufficient memory to open the document.

Notifications

AVAppFrontDocDidChange
AVDocDidActivate
AVDocDidDeactivate

The following notifications are broadcast if the document has a valid open action:

AVDocWillPerformAction
AVDocDidPerformAction

Header File

AVCalls.h

Related Methods

AVDocOpenFromPDDocWithParams

AVDocOpenFromPDDocWithParamString
AVDocOpenFromFile
AVDocOpenFromASFileWithParams
AVDocClose

Example

```
AVDocOpenFromPDDoc (somePDDoc, NULL) ;
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

Changed from **char*** to **ASText** argument type in **0x00060000**.

AVDocOpenFromPDDocWithParams

```
AVDoc AVDocOpenFromPDDocWithParams (PDDoc pdDoc,
const AText tempTitle, AVDocOpenParams params);
```

Description

Opens and displays a document from a **PDDoc**, using the specified parameters to control the window's size, location, and visibility.

If you want to display a page from a PDF file as a bitmap, use [PDPPageDrawContentsToWindow](#).

NOTE: Do not open and then immediately close an AVDoc without letting at least one event loop complete.

Parameters

pdDoc	The document to open and display.
tempTitle	If tempTitle != NULL , pathname is a temporary file and tempTitle is used as the window's title.
params	Parameters used when opening the file. Can be NULL .

Return Value

The document that was opened.

Notifications

[AVAppFrontDocDidChange](#)
[AVDocDidActivate](#)
[AVDocDidDeactivate](#)

The following notifications are broadcast if the document has a valid open action:

[AVDocWillPerformAction](#)
[AVDocDidPerformAction](#)

Header File

AVCalls.h

Related Methods

[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromASFileWithParams](#)
[AVDocOpenFromPDDoc](#)
[AVDocOpenFromPDDocWithParamString](#)
[AVDocClose](#)

Example

```
AVDocOpenParamsRec params;  
  
params.size = sizeof(AVDocOpenParamsRec);  
params.useFrame = false;  
params.useVisible = true;  
params.visible = false;  
AVDocOpenFromPDDocWithParams(somePDDoc,  
    NULL, &params);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

Changed from **char*** to **ASText** argument type in **0x00060000**.

AVDocOpenFromPDDocWithParamString

```
AVDoc AVDocOpenFromPDDocWithParamString (PDDoc pdDoc,
const ASText tempTitle, AVDocOpenParams params,
const char* s);
```

Description

Opens and displays a document from a **PDDoc**, using the specified parameters to control the window's size, location, and visibility, and passing a URL open action string to control how the file is opened. For more information, see the document *URL Open Actions*.

If you want to display a page from a PDF file as a bitmap, use [PDPageDrawContentsToWindow](#).

NOTE: Do not open and then immediately close an **AVDoc** without letting at least one event loop complete.

Parameters

pdDoc	The document to open and display.
tempTitle	If tempTitle != NULL, pathname is a temporary file and tempTitle is used as the window's title.
params	Parameters used when opening the file. Can be NULL .
s	A string containing the URL open action. For example: Help=contents&nameddest=distiller_mydest This would open the PDF in the help window, with the contents panel showing on the left, and the page referenced by the named destination distiller_mydest showing on the right.

Return Value

The document that was opened.

Notifications

[AVAppFrontDocDidChange](#)
[AVDocDidActivate](#)
[AVDocDidDeactivate](#)

The following notifications are broadcast if the document has a valid open action:

[AVDocWillPerformAction](#)
[AVDocDidPerformAction](#)

Header File

AVCalls.h

Related Methods

[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromFileWithParamString](#)
[AVDocOpenFromPDDoc](#)
[AVDocOpenFromPDDocWithParams](#)
[AVDocClose](#)

Example

```
AVDocOpenParamsRec params;  
  
params.size = sizeof(AVDocOpenParamsRec);  
params.useFrame = false;  
params.useVisible = true;  
params.visible = false;  
AVDocOpenFromPDDocWithParams(somePDDoc,  
                             NULL, &params);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVDocPerformAction

```
void AVDocPerformAction (AVDoc doc, PDAction action);
```

Description

Performs an action.

Parameters

doc	The document containing the action to perform.
action	The action to perform.

Return Value

None

Known Exceptions

[pdErrBadAction](#)

Notifications

[AVDocWillPerformAction](#)
[AVDocDidPerformAction](#)

Header File

AVCalls.h

Related Methods

[AVDocDoActionPropsDialog](#)

Example

```
DURING
    action = PDBookmarkGetAction(item);
HANDLER
    err = ERRORCODE;
END_HANDLER
if(!err){
    DURING
        AVDocPerformAction(doc, action);
    HANDLER
        err = ERRORCODE;
    END_HANDLER
}
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVDocPerformActionEx

```
void AVDocPerformActionEx (AVDoc doc, PDAction action,  
AVActionContext data);
```

Description

Same as [AVDocPerformAction](#), but provides context for the execution of the action.

Parameters

doc	The document containing the action to perform.
action	The action to perform.
data	Context for the execution of the action.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVDocGetPDDoc](#)

[AVDocPerformAction](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVDocPermRequest

```
ASBool AVDocPermRequest (AVDoc doc, PDPermReqObj obj,
PDPermReqOpr opr);
```

Description

An exact way to ask if an operation is permitted. This routine should be used to query all UI-level permissions—for example, to determine whether a tool button or menu item is enabled or whether a tool can act upon a given document. This routine should be used instead of some combination of [AVDocIsReadOnly](#)/[AVDocIsExternal](#)/[PDDocPermRequest](#). **AVDocPermRequest** calls **PDDocPermRequest** internally in the process of determining whether to grant the request.

Parameters

doc	The document opened in Acrobat.
obj	Description of target object of permissions request.
opr	Description of the target operation of a permissions request.

Return Value

true if operation is permitted; **false** otherwise.

Header File

AVCalls.h

Related Methods

```
AVDocIsReadOnly
AVDocIsExternal
PDDocPermRequest
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVDocPrintPages

```
void AVDocPrintPages (AVDoc doc, ASInt32 firstPage,
ASInt32 lastPage, ASInt32 psLevel, ASBool binaryOK,
ASBool shrinkToFit);
```

Description

Prints without displaying any user dialogs. The current printer, page settings, and job settings are used. Printing is complete when this method returns.

You can replace this method with your own version, using [HFTReplaceEntry](#).

NOTE: If security has been set on a file so that it is not printable, the document will not print, but no error is raised. Check the security before printing the file.

Parameters

doc	The document from which pages are printed.
firstPage	The first page in doc to print.
lastPage	The last page in doc to print.
psLevel	Applies to PostScript printing. Must be either 1, 2, or 3. If 1, Level 1 PostScript code is generated. If 2, Level 2 PostScript code is generated. If 3, Level 3 PostScript code is generated.
binaryOK	Applies to PostScript printing. If true , the PostScript code may contain binary data. If false , all binary data is encoded into an ASCII format.
shrinkToFit	If true , the page is shrunk (if necessary) to fit into the image-able area of a page in the printer. If false , pages are printed at actual size and may appear clipped on the printed page.

Return Value

None

Known Exceptions

Raises [genErrBadParm](#) if an invalid parameter is provided. Can raise any of the [CosErrExpected](#) exceptions, such as [ErrSysCosSyntax](#) or [cosErrExpectedNumber](#).

In general, this method can raise any exception that can occur during the parsing of a page and its resources, such as [pdErrUnknownProcsets](#) or [pdErrUnableToExtractFontErr](#).

Notifications

[PDDocWillPrintPages](#)
[PDDocWillPrintPage](#)



[PDDocDidPrintPage](#)
[PDDocDidPrintPages](#)
[PDDocDidPrintTiledPage](#)
[PDDocPrintingTiledPage](#)
[PDDocWillPrintTiledPage](#)

Header File

AVCalls.h

Related Methods

[AVDocDoPrint](#)
[AVDocPrintPagesWithParams](#)

Example

```
AVDocPrintPages(AVAppGetActiveDoc(), 0, 2,  
                2, false, false);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVDocPrintPagesWithParams

```
void AVDocPrintPagesWithParams (AVDoc doc,  
AVDocPrintParams params);
```

Description

Prints a document with a full range of options. Printing is complete when this method returns. Performs “embedded” printing; that is, allows a PDF page to print within a bounding rectangle on a page. Allows interactive printing to the specified printer (Windows, Mac OS, and UNIX).

You can replace this method with your own version, using [HFTReplaceEntry](#).

NOTE: If security has been set on a file so that it is not printable, the document will not print, but no error is raised. Check the security before printing the file.

Parameters

doc	The AVDoc from which to print pages.
params	A structure containing printing parameters. See AVDocPrintParams . NOTE: With Adobe Reader, you cannot use the emitToFile flag. For Adobe Reader, use the emitToPrinter , interactive , or embedded flags. NOTE: In Adobe Acrobat 6.0 Standard, the printer mark drawing flags (marksFlags) in the parameters structure are ignored.

Return Value

None

Known Exceptions

Raises [genErrBadParm](#) if an invalid parameter is provided. Can raise any of the **CosErrExpected** exceptions, such as [ErrSysCosSyntax](#) or [cosErrExpectedNumber](#).

In general, this method can raise any exception that can occur during the parsing of a page and its resources, such as [pdErrUnknownProcsets](#) or [pdErrUnableToExtractFontErr](#).

Notifications

- [PDDocWillPrintPages](#)
- [PDDocWillPrintPage](#)
- [PDDocDidPrintPage](#)
- [PDDocDidPrintPages](#)
- [PDDocDidPrintTiledPage](#)
- [PDDocPrintingTiledPage](#)

[PDDocWillPrintTiledPage](#)**Header File**

AVCalls.h

Related Methods[AVDocDoPrint](#)[AVDocPrintPages](#)**Example One**

```

AVDocPrintParamsRec myParams;
AVDoc doc;
char buf[255];
ASPathName name = NULL;
doc = AVAppGetActiveDoc();

name = ASPathFromPlatformPath(
(void*)"c:\\test.ps");

if (name) {
    AAlertNote("Printing Starts");
    memset(&myParams, 0, sizeof(AVDocPrintParamsRec));
    myParams.size = sizeof(AVDocPrintParamsRec);

    myParams.emitToFile = true;
    myParams.interactive = false;
    myParams.emitToPrinter = false;
    myParams.embedded = false;

    myParams.doColorSeparations = false;
    myParams.binaryOK = false;
    myParams.shrinkToFit = true;
    myParams.firstPage = -1L; /* should print all pages */
    myParams.lastPage = -1L;
    myParams.psLevel = 2L;

    myParams.fileSysName = ASAtomNull;
    myParams.filePathName = name;
    myParams.printerSpec = NULL;

    myParams.doColorSeparations = false;
    myParams.emitFileOption = kAVEmitFilePS;
    myParams.emitFontOption = kAVEmitFontEmbeddedFonts;
DURING
    AVDocPrintPagesWithParams(doc, &myParams);

HANDLER
    ASGetErrorString(ERRORCODE, buf, sizeof(buf));
    AAlertNote(buf);
    return;

```

```

END_HANDLER
AVAlertNote("Printing Ends");
}
else
AVAlertNote("bad path name");

```

Example Two

```

void doPrint (AVDoc avdoc, char *psFileName, int pageSpec)
{
    PDDoc pddoc;
    AVDocPrintParamsRec params;
    ASPathName aspathname;
    PDPageRange *pageRangeArray;
    AVRect32 avrect32;
    ASCallback pageSetupCallback;

    pageRangeArray = (PDPageRange *)malloc (sizeof (PDPageRange));
    pddoc = AVDocGetPDDoc (avdoc);
    memset (&params, 0, sizeof (params));
    params.size = sizeof (params);
    params.cancelDialog = 0;
    params.psLevel = 2;
    params.binaryOK = false;
    params.shrinkToFit = false;
    params.fileSysName = (ASGetDefaultFileSys ())->getFileSysName ();
    aspathname = ASPathFromPlatformPath (psFileName);
    params.filePathName = aspathname;
    memset (&avrect32, 0, sizeof (avrect32));
    params.embeddedRect = avrect32;
    params.printerSpec = NULL;
    params.interactive = false;
    params.embedded = false;
    params.emitToPrinter = false;
    params.emitToFile = true;
    params.doColorSeparations = false;
    params.emitFileOption = kAVEmitFilePS;
    params.emitFontOption = kAVEmitFontAllFonts;
    params.emitFlags = 0;
    params.firstPage = pageRangeArray ->startPage = 0;
    params.lastPage = pageRangeArray ->endPage =
        PDDocGetNumPages (pddoc);
    pageRangeArray->pageSpec = pageSpec;
    params.ranges = pageRangeArray;
    params.numRanges = 1;
    params.TTast42 = false;
    params.printAsImage = false;
    params.printerHasFarEastFonts = false;
    params.reverse = false;
    params.pageSpec = pageSpec;
}

```

```

    pageSetupCallback = ASCallbackCreateNotification
        (PSPrintAfterBeginPageSetup,
         &callbackPSPrintAfterBeginPageSetup);
    AVAppRegisterNotification (PSPrintAfterBeginPageSetupNSEL,
                              gExtensionID, pageSetupCallback, (void*)&params));

#ifdef WIN32
    UpdateWindow (gHWNDD);
#endif
    AVDocPrintPagesWithParams (avdoc, &params);
    AVAppUnregisterNotification (PSPrintAfterBeginPageSetupNSEL,
                                gExtensionID, pageSetupCallback, (void*)&params));
    free (pageRangeArray);
    ASFileSysReleasePath (ASGetDefaultFileSys (), aspathname);
}

```

Example Three

```

/***** AVPrintSnip*****/
-Checks the doc security settings before printing the file.
-Allows interactive printing to the specified printer (Windows, Mac
OS, and UNIX).
-Prints a document with a full range of options.
-Prints to a PS file or a PS printer.
*****/

ACCB1 void ACCB2 AVPrintSnip()
{
    AVDoc d = AVAppGetActiveDoc();
    if(d == NULL){
        AAlertNote ("A document needs to be open...");
        return;
    }

    // If security has been set on a file so that it is not printable,
    //the document won't print, but no error is raised by the viewer.
    // Check the security before printing the file.
    if (!AVDocPermRequest(d, PDPermReqObjDoc, PDPermReqOprPrintHigh) ||
        !AVDocPermRequest(d, PDPermReqObjDoc, PDPermReqOprPrintLow))
    {
        AAlertNote ("Printing is not permitted");
        return;
    }

    AVDocPrintParamsRec r;
    memset (&r, 0, sizeof (AVDocPrintParamsRec));

    r.size = sizeof(AVDocPrintParamsRec);
    r.pageSpec = PDAllPages;

    // if true, displays dialog boxes;

```

```

// otherwise does not display dialog boxes.
r.interactive = false;

// you have choices of printing to a ps file or to a ps printer
if (EMIT_TO_FILE == true){
    r.emitToPrinter = false;
// only one of emitToPrinter or emitToFile must be true
    r.emitToFile = true;
    r.filePathName = ASFileSysCreatePathName(NULL,
        ASAtomFromString("Cstring"), "test_print.ps", NULL);
}
if (EMIT_TO_PRINTER == true){
    r.emitToPrinter = true;
    r.emitToFile = false;
    r.filePathName = ASFileSysCreatePathName(NULL,
        ASAtomFromString("Cstring"), PRINTER_NAME, NULL);
}

// specify your own printing rec here
r.embedded = true;
// Gets the rectangle that bounds a page view.
// This may include some of the "gray area" outside a page's crop box.
AVRect bounds;
AVPageViewGetGrayRect(AVDocGetPageView(d), &bounds);
r.embeddedRect = bounds;

r.firstPage = 0;
r.lastPage = PDDocGetNumPages(AVDocGetPDDoc(d)) - 1;
r.psLevel = 2;
r.binaryOK = false;
r.shrinkToFit = false;
r.cancelDialog = false;
r.doColorSeparations = false;

/* file output options: PostScript or EPS, with or without a
   preview. Must be one of the following values:
   kAVEmitFilePS - PostScript file
   kAVEmitFileEPSNoPrev - EPS file with no preview
   kAVEmitFileEPSMacStdPrev - EPS file with standard preview
   kAVEmitFileEPSMacExtPrev - EPS file with extended preview
*/
r.emitFileOption = false;
r.emitFlags = 0;

/* font option is one of the following:
kAVEmitFontNoFonts (Obs after Acrobat 3.0) Embed no fonts.
kAVEmitFontAllEmbType1 (Obs after Acrobat 3.0) Embed all Type 1
embedded fonts.
kAVEmitFontAllType1 (Obs after Acrobat 3.0) Embed all Type 1 fonts.
kAVEmitFontEmbeddedFonts Emit all embedded fonts.
kAVEmitFontAllFonts Emit all fonts.

```

```

*/
r.emitFontOption = kAVEmitFontAllFonts;
r.ranges = NULL;
r.numRanges = 0;
r.printAsImage = false;
r.reverse = false;
r.transparencyLevel = 3;

strcpy(&r.destProfile[0], "Printer/PostScript printing");

ASInt32 error = 0;

DURING
    AVDocPrintPagesWithParams(d, &r);
HANDLER
    error = ERRORCODE;
    ASRaise (error);
END_HANDLER
if (!error)
    AAlertNote ("The file was printed successfully.");
}

```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

AVDocPrintSeparations

```
void AVDocPrintSeparations (AVDocPrintSepsParams params);
```

Description

Prints document color separations using the supplied parameters. Printing is complete when this method returns.

For Adobe Reader and Adobe Acrobat Standard, this method does nothing.

NOTE: If security has been set on a file so that it is not printable, the document will not print, but no error is raised. Check the security before printing the file.

Parameters

params	A structure containing print separation parameters. This structure includes the document object.

Return Value

None

Related Methods

[PDPageMakeSeparations](#)

Product

Pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVDocRegisterSelectionServer

```
ASBool AVDocRegisterSelectionServer
(AVDocSelectionServer server);
```

Description

Registers a new selection server with the Acrobat viewer. Selection servers allow the selection of items other than those that can be selected in the as-shipped Acrobat viewer. For example, a selection server could allow a user to select a sampled image.

This method can be used to replace an existing selection server that handles the same selection type.

Parameters

server	Structure containing the selection server's callback functions. This structure must not be freed after calling AVDocRegisterSelectionServer .

Return Value

Always returns **true**.

Known Exceptions

Raises [genErrBadParm](#) if **server** is **NULL**, if the **size** field in the **AVDocSelectionServerRec** is incorrect, or if the selection server's [AVDocSelectionGetTypeProc](#) is **NULL**.

Header File

AVCalls.h

Related Methods

- [AVDocClearSelection](#)
- [AVDocCopySelection](#)
- [AVDocDeleteSelection](#)
- [AVDocDoSelectionProperties](#)
- [AVDocEnumSelection](#)
- [AVDocGetSelection](#)
- [AVDocShowSelection](#)

Example

```

AVDocRegisterSelectionServer:
static AVDocSelectionServerRec mySelServer;

DrawImageSelectionCallback =
    ASCallbackCreateProto(
        AVPageViewDrawProc,
        &DrawImageSelection);
memset(&mySelServer, 0,
    sizeof(AVDocSelectionServerRec));
mySelServer.size =
    sizeof(AVDocSelectionServerRec);/*
    important for future */
mySelServer.GetType =
    ASCallbackCreateProto(
        AVDocSelectionGetTypeProc,
        &mySelServerGetType);
mySelServer.GettingSelection =
    ASCallbackCreateProto(
        AVDocSelectionGettingSelectionProc,
        &mySelServerGettingSelection);
mySelServer.LosingSelection =
    ASCallbackCreateProto(
        AVDocSelectionLosingSelectionProc,
        &mySelServerLosingSelection);
mySelServer.ShowSelection =
    ASCallbackCreateProto(
        AVDocSelectionShowSelectionProc,
        &mySelServerShowSelection);

AVDocRegisterSelectionServer(&mySelServer);

```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVDocSelectionEnumPageRanges

```
void AVDocSelectionEnumPageRanges (AVDoc doc,
AVSelectionPageRangeEnumProc enumProc, void* clientData);
```

Description

Enumerates the pages on which there is a selection by calling the current selection server's [AVDocSelectionEnumPageRangesProc](#) callback. This method allows determining which pages are currently involved in a selection, regardless of the selection type. This facilitates discovering whether a given point is in a selection or not.

Pages are enumerated in ascending order, and consecutive pages are grouped into a single page range.

Parameters

doc	The AVDoc from which to enumerate page ranges.
enumProc	User-supplied function that is called for each page on which there is a selection.
clientData	Pointer to user-supplied data to pass to enumProc each time it is called.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVDocEnumSelection](#)

Example

```
AVDoc avDoc;
AVSelectionPageRangeEnumProc pageSelectionEnumeratorCB;
pageSelectionEnumeratorCB = ASCallbackCreateProto
    (AVSelectionPageRangeEnumProc, pageSelectionEnumerator);
AVDocSelectionEnumPageRanges(avDoc, pageSelectionEnumeratorCB, void);
...
ACCB1 ASBool ACCB2 pageSelectionEnumerator (AVDoc doc, void* clientData,
ASInt32 firstPage, ASInt32 lastPage) {
    /* Some operations on each page range */ ... }
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

AVDocSendAuxData

```
ASBool AVDocSendAuxData (AVDoc avDoc, ASAtom auxDataType,
void* auxData, ASInt32 auxDataLen);
```

Description

Sends auxiliary data to an **AVDoc**. If an **AVAuxDataHandler** exists for the data type, the handler is called with **avDoc**, **auxDataType**, and **auxData**. The definition of the auxiliary data is dependent on the **AVAuxDataHandler**.

For any value of **auxDataType**, the **auxData** parameter must be predefined so that a user of this method knows what type of data to send. It is expected that the implementor of an **AVAuxDataHandler** provides this definition.

Parameters

avDoc	The target document.
auxDataType	The type of data being sent.
auxData	A pointer to the data.
auxDataLen	The length of the data.

Return Value

true if **auxData** was accepted by a handler, **false** otherwise.

Header File

AVCalls.h

Related Methods

- [AVHasAuxDataHandler](#)
- [AVRegisterAuxDataHandler](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

AVDocSetActiveTool

```
void AVDocSetActiveTool (AVDoc avDoc, AVTool tool,  
ASBool persistent);
```

Description

Sets the active tool for the **AVDoc**.

NOTE: It is recommended that you use **AVDocSetActiveTool** rather than **AVAppSetActiveTool**, preferably specifying a particular document.

Parameters

avDoc	The document whose active tool is set, or NULL to set the active tool for the application.
tool	The new active tool.
persistent	When true , the tool stays active after it is used. Passed to the tool's Activate callback.

Return Value

None

Known Exceptions

genErrNoMemory

Header File

AVCalls.h

Related Methods

AVDocGetActiveTool

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVDocSetClientName

```
void AVDocSetClientName (AVDoc avDoc, char* clientName);
```

Description

Sets the **AVDoc** client (container application) name. This method can be used by clients that open documents in the viewer via DDE or Apple events. Most clients will not open documents in this way, however, making this method unnecessary for most clients.

Parameters

avDoc	The document whose client names is set.
clientName	The buffer from which the client name is read. May be up to 255 characters, and must be null-terminated.

Return Value

None

Known Exceptions

[genErrNoMemory](#)

Header File

AVCalls.h

Related Methods

[AVDocGetClientName](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020001** or higher.

AVDocSetDead

```
void AVDocSetDead (AVDoc doc, ASBool dead);
```

Description

Indicates the file stream for this document is terminated, although the **AVDoc** is still open. No more data can be read from this **AVDoc**.

AVDocs that are marked as dead may start returning **NULL** at anytime between the initial call to **AVDocSetDead** and the time that the **AVDoc** is actually closed. So callers of **AVDocGetAVWindow** and other **AVDoc** property methods must check for a **NULL** return value.

Parameters

doc	The document to set as dead.
dead	true if the document's file stream is terminated, false otherwise.

Return Value

None

Notifications

If **dead** is **true**, broadcasts **AVDocWantsToDie**.

Header File

AVCalls.h

Related Methods

[AVDocClose](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.



AVDocSetReadOnly

```
void AVDocSetReadOnly (AVDoc doc, ASBool readOnly);
```

Description

Sets the read-only state of an **AVDoc**.

Parameters

doc	The AVDoc to set to read-only.
readOnly	true if the given document is set to read-only, false if it is set to read-write.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVDocIsReadOnly](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION**(in `PIRequir.h`) is set to **0x00040000** or higher.



AVDocSetSelection

```
ASBool AVDocSetSelection (AVDoc doc, ASAtom type, void* data,
ASBool highlight);
```

Description

Sets the document's current selection to the specified selection by calling the appropriate selection server's [AVDocSelectionGettingSelectionProc](#) callback. Clears the previous selection, if any, by calling the previous selection server's [AVDocSelectionLosingSelectionProc](#) callback.

Parameters

doc	The AVDoc in which the selection is set.
type	Selection type. Can be either a built-in type or one supported by a selection server added by a client. Can be converted to an ASAtom using ASAtomFromString . See Selection Types for a list of the built-in selection types.
data	Data structure representing the selection. data 's type depends on what is passed in type . See Selection Types for a list of the data types for the built-in selection types.
highlight	Clients should pass true , which tells the Acrobat viewer to highlight the selection because it has not already been highlighted. This only marks the highlighted regions of the display invalid, but does not immediately redraw the screen. Use AVPageViewDrawNow to force an immediate redraw if you wish.

Return Value

true if the selection was set successfully, **false** otherwise. Examples of why this method fails include:

- No selection server for **type**.
- Attempting to set the selection during link creation.

Known Exceptions

Only those exceptions raised by the previous selection server's [AVDocSelectionGettingSelectionProc](#), and those raised by the new selection server's [AVDocSelectionGettingSelectionProc](#).

Notifications

[AVDocDidSetSelection](#)

Header File

AVCalls.h

**Related Methods**

[AVGrafSelectCreate](#)
[AVPageViewDrawNow](#)
[AVDocRegisterSelectionServer](#)
[AVDocClearSelection](#)
[AVDocDeleteSelection](#)
[AVDocDoSelectionProperties](#)
[AVDocEnumSelection](#)
[AVDocCopySelection](#)
[PDDocCreateTextSelect](#)

Example

```

/* Select text */
PDTextSelect TextSelection;
HiliteEntry Hilite;

PDPPage CurrentPDPPage =
    PDDocAcquirePage(CurrentPDDoc, PageNum);
Hilite.offset= (ASUns16) Offset;
Hilite.length= 0;
TextSelection =
    PDTextSelectCreatePageHilite(
        CurrentPDPPage, &Hilite, 1);
AVDocSetSelection(CurrentAVDoc,
    ASAtomFromString("Text"), (void *)
    TextSelection, true);
AVDocShowSelection(CurrentAVDoc);
PDPPageRelease(CurrentPDPPage);

/* select 0th link annot on 0th page
(assumes that it's a link) */
#define k_Annot
    ASAtomFromString("Annotation")
CosObj tmp;
PDAnnot annot, *annotSel;
PDAction action;

DURING
    PDDocAcquirePage(pdDoc, 0); /* acquire
    0th page */
    annot = PDPPageGetAnnot(pdPage, 0); /* get
    0th annot on pdPage */
    AnnotSel = ASmalloc(sizeof(PDAnnot));
    if(annotSel){
        tmp = PDActionGetCosObj(action); /*
        so we can copy its contents */
        *annotSel = tmp; /*copy the contents*/

        /* the selection server will ASfree
        the pointer when done with
        annotSel */
        AVDocSetSelection(avdoc, k_Annot,
            &annotSel, true); }
    PDPPageRelease(pdPage);
HANDLER
    if(pdPage)
        PDPPageRelease(pdPage);
END_HANDLER

```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVDocSetSplitterPosition

```
void AVDocSetSplitterPosition (AVDoc doc,
ASInt16 newPosition);
```

Description

Sets the splitter position. The splitter is the vertical division between the bookmark/thumbnail pane and the document pane. The default splitter location is saved in the Acrobat viewer's preferences file, and can be read/set using [AVAppGetPreference](#) and [AVAppSetPreference](#).

Parameters

doc	The document whose splitter position is set.
newPosition	The new splitter position. This value specifies the width of the bookmark/thumbnail pane in pixels.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVDocGetSplitterPosition](#)

[AVAppSetPreference](#)

Example

```
if (AVDocGetSplitterPosition(doc) < 72)
    AVDocSetSplitterPosition(doc, 72);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVDocSetViewDef

```
void AVDocSetViewDef (AVDoc doc, AVDocViewDef viewDef);
```

Description

NOTE: Numeric types in [AVDocViewDef](#) have changed in Acrobat 6.0, and the method has been superseded by [AVDocGetViewDefEx](#).

Sets the document's state to match the information in **viewDef**.

Parameters

doc	The document whose state is updated.
viewDef	A pointer to the structure that stores the state.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVDocGetViewDef](#)
[AVDocGetViewDefEx](#)
[AVDocSetViewDefEx](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

AVDocSetViewDefEx

```
void AVDocSetViewDefEx (AVDoc doc, ASCab viewDef);
```

Description

Sets the document's state to match the information in **viewDef**.

NOTE: Supersedes [AVDocGetViewDef](#) in Acrobat 6.0.

Parameters

doc	The document whose state is updated.
viewDef	A pointer to the ASCab that stores the state.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVDocGetViewDefEx](#)
[AVDocSetViewDef](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVDocSetViewMode

```
void AVDocSetViewMode (AVDoc doc, PDPageMode newMode);
```

Description

Sets the current view mode.

This method does nothing if the current view mode is full-screen, **PDFullScreen**. In this case, call **AVAppEndFullScreen** first.

Parameters

doc	The document whose view mode is set.
newMode	The view mode to set.

Return Value

None.

Header File

AVCalls.h

Related Methods

[AVAppEndFullScreen](#)
[AVDocGetViewMode](#)

Example

```
if (AVDocGetViewMode(doc) == PDUseNone)
    AVDocSetViewMode(doc, PDUseBookmarks);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVDocShowSelection

```
void AVDocShowSelection (AVDoc doc);
```

Description

Displays the current selection by calling the selection server's [AVDocSelectionShowSelectionProc](#) callback. Does nothing if the document has no selection or the current selection's server has no [AVDocSelectionShowSelectionProc](#) callback.

Parameters

doc	The document whose selection is shown.

Return Value

None

Known Exceptions

Only those raised by the selection server's [AVDocSelectionShowSelectionProc](#) callback.

Header File

AVCalls.h

Related Methods

[AVDocSetSelection](#)
[AVGrafSelectCreate](#)
[AVDocRegisterSelectionServer](#)
[PDDocCreateTextSelect](#)
[PDTextSelectCreatePageHilite](#)
[PDTextSelectCreateWordHilite](#)



Example

```

PDTextSelect TextSelection;
HiliteEntry Hilite;
PDPPage CurrentPDPPage = PDDocAcquirePage(
    CurrentPDDoc, PageNum);
Hilite.offset= (ASUns16) Offset;
Hilite.length= 0;
TextSelection =
    PDTextSelectCreatePageHilite(
        CurrentPDPPage, &Hilite, 1);
AVDocSetSelection(CurrentAVDoc,
    ASAtomFromString("Text"), (void *)
        TextSelection, true);
if(show)
    AVDocShowSelection(CurrentAVDoc);
else
    AVDocClearSelection(CurrentAVDoc, true);
PDPPageRelease(CurrentPDPPage);

```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVGrafSelect

AVGrafSelectCreate

```
AVGrafSelect AVGrafSelectCreate (AVPageView pageView,
const ASFixedRectP selRect);
```

Description

Creates a graphics selection. After creation, the selection can be set using [AVDocSetSelection](#).

Parameters

pageView	The AVPageView in which a graphics selection is created.
selRect	Pointer to the ASFixedRect bounding the region from which a graphics selection is created, specified in user space coordinates.

Return Value

The newly created **AVGrafSelect**, or **NULL** if **selRect** is **NULL** or is an empty rectangle.

Known Exceptions

[genErrBadParm](#)
[genErrNoMemory](#)

Header File

AVCalls.h

Related Methods

[AVGrafSelectDestroy](#)
[AVDocSetSelection](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVGrafSelectDestroy

```
void AVGrafSelectDestroy (AVGrafSelect avGraf);
```

Description

Destroys a graphics selection. Use this method to destroy the selection only if you have *not* called [AVDocSetSelection](#) with it. If the selection has been set by a call to [AVDocSetSelection](#), it will automatically be destroyed by a call to [AVDocClearSelection](#) or the next call to [AVDocSetSelection](#).

Parameters

avGraf	The graphics selection to destroy.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

AVCalls.h

Related Methods

[AVGrafSelectCreate](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVGrafSelectGetBoundingRect

```
void AVGrafSelectGetBoundingRect (AVGrafSelect avGraf,
ASFixedRect* boundRectP);
```

Description

Gets the specified graphics selection's bounding rectangle.

Parameters

avGraf	The graphics selection whose bounding rectangle is obtained.
boundRectP	Pointer to the graphics selection's bounding rectangle, specified in user space coordinates.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

AVCalls.h

Related Methods

[AVGrafSelectCreate](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVMenu

AVMenuAcquire

```
AVMenu AVMenuAcquire (AVMenu menu);
```

Description

Acquires the specified menu. Increments the menu's reference count. When you are done using the menu item, release it using [AVMenuRelease](#).

Parameters

menu	The menu to acquire.

Return Value

The menu acquired.

Header File

AVCalls.h

Related Methods

[AVMenubarAcquireMenuByIndex](#)
[AVMenubarAcquireMenuByName](#)
[AVMenubarAcquireMenuItemByPredicate](#)
[AVMenuRelease](#)
[AVMenuNew](#)

Example

```
AVMenu tempMenu;
ASInt32 i;
for (i=0; i< AVMenubarGetNumMenus (
    AVAppGetMenubar()); i++)
{
    tempMenu = AVMenubarAcquireMenuByIndex(
        AVAppGetMenubar(), i);
    if (AVMenuGetName(tempMenu) ==
        ASAtomFromString("Edit"))
        AVMenuRemove(tempMenu);
    break;
}
AVMenuRelease(tempMenu);
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenuAcquireMenuItemByIndex

```
AVMenuItem AVMenuAcquireMenuItemByIndex (AVMenu menu,
ASInt32 menuItemIndex);
```

Description

Acquires the menu item at the specified location in the specified menu. When you are done using the menu item, release it using [AVMenuItemRelease](#). Menu item indices are generally not reliable—they change as clients add, remove, or rearrange menu items, and may differ in different versions of the Acrobat viewer (if menu items are rearranged, removed, or added). Menu items should generally be acquired using [AVMenubarAcquireMenuItemByName](#), which is generally reliable.

Parameters

menu	The menu in which a menu item is acquired.
menuItemIndex	The index of the menu item in menu to acquire. The first item in a menu has an index of zero. Even if the Acrobat viewer is displaying short menus, the index includes any intervening long-mode-only menu items (irrelevant for Acrobat 3.0 or later since there are no short menus).

Return Value

The menu item whose index is specified. Returns **NULL** if **menu** is **NULL**, if the index is less than zero, or the index is greater than the number of menu items in the menu.

Header File

AVCalls.h

Related Methods

[AVMenuGetNumMenuItems](#)
[AVMenubarAcquireMenuItemByPredicate](#)
[AVMenubarAcquireMenuItemByName](#)
[AVMenuItemAcquire](#)
[AVMenuItemRelease](#)

Example

```
AVMenuItem tempItem;
ASInt32 i;

for(i=0;i< AVMenuGetNumMenuItems(
    editMenu); i++)
{
    tempItem =
        AVMenuAcquireMenuItemByIndex(
            editMenu, i);
    AVMenuItemGetTitle(tempItem, buf,
        sizeof(buf));
    if(!strcmp(buf, "Paste"))
        AVMenuItemSetTitle(tempItem, "Glue");
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenuAddMenuItem

```
void AVMenuAddMenuItem (AVMenu menu, AVMenuItem menuItem,
                        ASInt32 menuItemIndex);
```

Description

Inserts a menu item in a specified location in a menu, and acquires the item. Does nothing if the menu or menu item is **NULL**, or the menu item is already in a menu.

Parameters

menu	The menu to which a menu item is added.
menuItem	The menu item to add.
menuItemIndex	The location in menu to add menuItem . The first item in a menu has an index of zero. Even if the Acrobat viewer is displaying short menus, the index includes any intervening long-mode-only menu items (irrelevant for Acrobat 3.0 or later since there are no short menus). Pass APPEND_MENUITEM (see <code>AVExpT.h</code>) to append the menu item to the end of the menu.

Return Value

None

Known Exceptions

[genErrNoMemory](#)

Header File

`AVCalls.h`

Related Methods

[AVMenuItemRemove](#)

Example

```
myButtonItem = AVMenubarAcquireMenuItemByName(menubar, "Hello");
helloIndex = AVMenuGetMenuItemIndex(menu, myButtonItem);
AVMenuAddMenuItem(menu, notesMenuItem,
                  helloIndex+1);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVMenuClone

```
AVMenu AVMenuClone (AVMenu menu) ;
```

Description

(New in Acrobat 5.0.5)

Clones a menu, creating a new menu which is an exact duplicate of the original, except that shortcuts are not created for the items.

Parameters

menu	The menu that is cloned.

Return Value

The new menu.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050001** or higher.



AVMenuDoPopUp

[AVMenuItem](#) AVMenuDoPopUp ([AVMenu](#) menu, [ASInt16](#) x, [ASInt16](#) y, [ASBool](#) rightButton, [ASInt32](#) choice)

Description

Pops up a submenu.

Parameters

menu	The menu to be displayed.
x	The x-coordinate of the upper left corner of the menu.
y	The y-coordinate of the upper left corner of the menu.
rightButton	true if the right mouse button (where applicable) was used to invoke the popup, false otherwise.
choice	The index of the AVMenuItem that should appear under the mouse at pop-up time.

Return Value

The menu item selected from **menu**.

Header File

AVCalls.h

Related Methods

[AVPageViewDoPopupMenu](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVMenuGetMenuItemIndex

```
ASInt32 AVMenuGetMenuItemIndex (AVMenu menu,
AVMenuItem menuItem);
```

Description

Gets the index of the specified menu item in the specified menu.

Indices must be used with caution, because they change as clients add, remove, or rearrange menu items.

Parameters

menu	The menu in which menuItem is located.
menuItem	The menu item whose index is obtained.

Return Value

The index of **menuItem** in **menu**. The first item in a menu has an index of zero. Even if the Acrobat viewer is displaying short menus, the index includes any intervening long-mode-only menu items (irrelevant for Acrobat 3.0 or later since there are no short menus).

Returns **BAD_MENUITEM_INDEX** (see `AVExpT.h`) if **menuItem** is not in **menu**. Also returns **BAD_MENUITEM_INDEX** if **menu** is **NULL** or **menuItem** is **NULL**.

Header File

AVCalls.h

Related Methods

[AVMenuGetNumMenuItems](#)

Example

```
myButtonMenuItem =
    AVMenubarAcquireMenuItemByName (menubar,
    "Hello");
helloIndex = AVMenuGetMenuItemIndex (menu,
    myButtonMenuItem);
AVMenuAddMenuItem (menu, notesMenuItem,
    helloIndex+1);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenuGetName

```
ASAtom AVMenuGetName (AVMenu menu);
```

Description

Gets the **ASAtom** for the menu's language-independent name.

Parameters

menu	The menu whose language-independent name is obtained.

Return Value

The menu's name. The **ASAtom** can be converted to a string using [ASAtomGetString](#). Returns **NULL** if **menu** is **NULL**.

Header File

AVCalls.h

Related Methods

[AVMenuAcquire](#)
[AVMenuGetTitle](#)
[AVMenuNew](#)

Example

```
AVMenu tempMenu;
ASInt32 i;

for (i=0; i< AVMenubarGetNumMenus (
    AVAppGetMenubar()); i++) {
    tempMenu =
        AVMenubarAcquireMenuByIndex (
            AVAppGetMenubar(), i);
    if (AVMenuGetName (tempMenu) ==
        ASAtomFromString ("Edit"))
    {
        AVMenuRemove (tempMenu);
        break;
    }
}
AVMenuRelease (tempMenu);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVMenuGetNumMenuItems

[AVTArraySize](#) AVMenuGetNumMenuItems ([AVMenu](#) menu) ;

Description

Gets the number of menu items in a menu, including those that are visible only in long-menus mode.

Parameters

menu	The menu for which the number of menu items is obtained.

Return Value

The number of menu items in the specified menu. Returns 0 if **menu** is **NULL**.

Header File

AVCalls.h

Related Methods

[AVMenuAcquireMenuItemByIndex](#)

Example

```
if (AVMenuGetNumMenuItems(myMenu)) {
    /* nonzero means process it */
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVMenuGetParentMenubar

[AVMenubar](#) AVMenuGetParentMenubar ([AVMenu](#) menu) ;

Description

Gets the parent menubar for the specified menu.

Parameters

menu	The menu whose parent menubar is obtained.

Return Value

The menubar to which the menu is attached. Returns **NULL** if the menu has not yet been added to the menubar or if the menu is a submenu.

Header File

AVCalls.h

Related Methods

[AVMenuGetParentMenuItem](#)
[AVMenubarGetMenuIndex](#)
[AVMenuItemGetParentMenu](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVMenuGetParentMenuItem

```
AVMenuItem AVMenuGetParentMenuItem (AVMenu menu);
```

Description

Gets the parent menu item for the specified menu.

Parameters

menu	The menu whose parent menu item is obtained.

Return Value

The menu item for which the specified menu is a submenu. Returns **NULL** if the specified menu is not a submenu.

Header File

AVCalls.h

Related Methods

[AVMenuGetParentMenubar](#)
[AVMenubarGetMenuIndex](#)
[AVMenuItemGetParentMenu](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenuGetTitle

```
AVTArraySize AVMenuGetTitle (AVMenu menu, char* buffer,
AVTArraySize bufferSize);
```

Description

Gets the menu's title as it appears in the user interface. The length of the title remains 0 if menu is **NULL**.

Parameters

menu	The menu whose title is obtained.
buffer	<i>(Filled by the method)</i> Buffer into which the title is copied. If buffer is NULL , it is not filled; the length of the title is still returned.
bufferSize	The maximum number of characters the buffer can hold.

Return Value

If **menu** is nonzero, returns the length of the title. If **menu** is **NULL** and **buffer** is not, **buffer** is zero-ed. Returns 0 if **buffer** is **NULL**.

Header File

AVCalls.h

Related Methods

[AVMenuGetName](#)
[AVMenuNew](#)
[AVMenuItemGetTitle](#)

Example

```
AVMenu tempMenu;
ASInt32 i;
for(i=0; i< AVMenubarGetNumMenus(
    AVAppGetMenubar()); i++){
    tempMenu = AVMenubarAcquireMenuByIndex( AVAppGetMenubar(), i);
    AVMenuGetTitle(tempMenu, buf, sizeof(buf));
    if(!strcmp(buf, "Edit")){
        AVMenuSetTitle(tempMenu, "Change");
        AVMenuRelease(tempMenu);
        break;
    }
    AVMenuRelease(tempMenu);
}
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenuGetTitleAsASText

```
void AVMenuGetTitleAsASText (AVMenu menu, AText title);
```

Description

Gets the menu's title as it appears in the user interface, as an **AText** object.

Parameters

menu	The menu whose title is obtained.
title	<i>(Filled by the method)</i> The text object containing the title.

Return Value

None.

Header File

AVCalls.h

Related Methods

[AVMenuGetName](#)
[AVMenuGetTitle](#)
[AVMenuNewWithASText](#)
[AVMenuItemGetTitleAsASText](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVMenuIsHiddenOnMenubar

```
ASBool AVMenuIsHiddenOnMenubar (AVMenu menu);
```

Description

Tests whether a menu is hidden on the menubar.

Parameters

menu	The menu to test.

Return Value

true if menu is hidden, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVMenubarAddHiddenMenu](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

AVMenuNew

```
AVMenu AVMenuNew (const char* title, const char* name,
ASExtension owner);
```

Description

Creates and acquires a new menu with the given title and language-independent name. The menu can be added to the menubar using [AVMenubarAddMenu](#). When you are done using the menu, release it using [AVMenuRelease](#).

Parameters

title	The string that appears in the user interface. In Windows, an ampersand (&) character in the string results in underlining the character after it on the menu.
name	Language-independent name of the menu to create: it is the value returned by AVMenuGetName . It must <i>not</i> contain any spaces. Developers should prefix the names of menus they add with the name of their client and a colon, to avoid collisions in the menu name space. For example, a client named myPlug might add menus named myPlug:DrawTools and myPlug:Checkout .
owner	The gExtensionID extension registering the menu.

Return Value

The newly created menu.

Known Exceptions

[genErrNoMemory](#)

Header File

AVCalls.h

Related Methods

[AVMenuRelease](#)
[AVMenubarAddMenu](#)
[AVMenuGetName](#)

Example

```
AVMenu notesMenu;
notesMenu = AVMenuNew("Notes", "NewNotes", gExtensionID);
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenuNewWithASText

```
AVMenu AVMenuNewWithASText (ASConstText title,
const char* name, ASExtension owner);
```

Description

Creates and acquires a new menu with the given title and language-independent name. The menu can be added to the menubar using [AVMenubarAddMenu](#). When you are done using the menu, release it using [AVMenuRelease](#).

Parameters

title	A constant text object containing the string that appears in the user interface. In Windows, an ampersand (&) character in the string results in underlining the character after it on the menu.
name	Language-independent name of the menu to create: it is the value returned by AVMenuGetName . It must <i>not</i> contain any spaces. Developers should prefix the names of menus they add with the name of their client and a colon, to avoid collisions in the menu name space. For example, a client named myPlug might add menus named myPlug:DrawTools and myPlug:Checkout .
owner	The gExtensionID extension registering the menu.

Return Value

The newly created menu.

Known Exceptions

[genErrNoMemory](#)

Header File

AVCalls.h

Related Methods

[AVMenuRelease](#)
[AVMenubarAddMenu](#)
[AVMenuGetName](#)
[AVMenuGetTitleAsASText](#)
[AVMenuNew](#)

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVMenuRelease

```
void AVMenuRelease (AVMenu menu);
```

Description

Releases the specified menu. Decrements the reference count and automatically destroys the menu when its reference count is zero.

Parameters

menu	The menu to release.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVMenuAcquire](#)

[AVMenuNew](#)

Example

```
AVMenu tempMenu;
ASInt32 i;

for(i=0; i< AVMenubarGetNumMenus(
    AVAppGetMenubar()); i++)
{
    tempMenu = AVMenubarAcquireMenuByIndex(
        AVAppGetMenubar(), i);
    if (AVMenuGetName(tempMenu) ==
        ASAtomFromString("Edit"))
        AVMenuRemove(tempMenu);
    break;
}
AVMenuRelease(tempMenu);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenuRemove

```
void AVMenuRemove (AVMenu menu);
```

Description

Removes a menu from the menubar and releases it. If the menu is a submenu, this method does nothing.

Parameters

menu	The menu to remove.

Return Value

None

Header File

AVCalls.h

Related Methods

- [AVMenuItemRemove](#)
- [AVMenuRelease](#)

Example

```
AVMenu tempMenu;
ASInt32 i;

for(i=0; i< AVMenubarGetNumMenus(
    AVAppGetMenubar()); i++){
    tempMenu =
        AVMenubarAcquireMenuByIndex(
            AVAppGetMenubar(), i);
    if (AVMenuGetName(tempMenu) ==
        ASAtomFromString("Edit")) {
        AVMenuRemove(tempMenu);
        break;
    }
    AVMenuRelease(tempMenu);
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenubar

AVMenubarAcquireMenuByIndex

```
AVMenu AVMenubarAcquireMenuByIndex (AVMenubar menubar,
ASInt32 menuIndex);
```

Description

Acquires the menu with the specified index. Menu indices are generally not reliable—they change as clients add, remove, or rearrange menus, and may differ in different versions of the Acrobat viewer (if menus are rearranged, removed, or added). Menus should generally be acquired using [AVMenubarAcquireMenuByName](#), which is generally reliable.

Parameters

menubar	The menubar in which the menu is located.
menuIndex	The index (in menubar) of the menu to acquire.

Return Value

The menu with the specified index. Returns **NULL** if no such menu or if **menubar** is **NULL**.

Header File

AVCalls.h

Related Methods

[AVMenubarAcquireMenuByName](#)
[AVMenubarAcquireMenuByPredicate](#)
[AVAppGetMenubar](#)
[AVMenuRelease](#)

Example

```
AVMenu fileMenu;

fileMenu = AVMenubarAcquireMenuByIndex(
    AVAppGetMenubar(), 1);
if (AVMenubarGetMenuIndex(fileMenu) != 1)
    AVAlertNote("Odd.");
AVMenuRelease(fileMenu);
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenubarAcquireMenuByName

[AVMenu](#) AVMenubarAcquireMenuByName ([AVMenubar](#) menubar, const char* name);

Description

Acquires the menu or submenu that has the specified language-independent menu name (case-sensitive). When you are done using the menu, release it using [AVMenuRelease](#). Acquiring a menu by name is generally reliable, because names (unlike indices) do not change as menus are added or rearranged.

Parameters

menubar	The menubar in which the menu item is located.
name	The language-independent name of the menu to acquire. See Menu and Menu Item Names for a list of the names of the built-in menus in Acrobat.

Return Value

The menu with the specified name.

Header File

AVCalls.h

Related Methods

[AVAppGetMenubar](#)
[AVMenuRelease](#)
[AVMenubarAcquireMenuItemByName](#)
[AVMenubarAcquireMenuItemByPredicate](#)
[AVMenubarAcquireMenuByIndex](#)

Example

```
menu = AVMenubarAcquireMenuByName (menubar,
    "Tools");
if (menu){
    myButtonItem = AVMenuItemNew(...);
    ...
    AVMenuRelease (menu);
}
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenubarAcquireMenuByPredicate

```
AVMenu AVMenubarAcquireMenuByPredicate (AVMenubar menubar,
AVMenuPredicate predicate, void* clientData);
```

Description

Acquires a menu using a user-supplied selection routine. This method can also be used to enumerate all menus. When you are done using the menu that is acquired, release it using [AVMenuRelease](#).

Parameters

menubar	The menubar containing the menu to acquire.
predicate	User-supplied AVMenuPredicate function that determines which menu is acquired. Menus are searched depth-first. The first menu for which predicate returns true is acquired. If predicate always returns false , all menus will be enumerated.
clientData	Pointer to user-supplied data to pass to predicate each time it is called.

Return Value

The first menu for which **predicate** returned **true**. Returns **NULL** if **predicate** never returned **true** or if **menubar** is **NULL**.

Header File

AVCalls.h

Related Methods

[AVMenuRelease](#)
[AVMenubarAcquireMenuByName](#)
[AVMenubarAcquireMenuByIndex](#)
[AVAppGetMenubar](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION**(in **PIRequir.h**) is set to **0x00020000** or higher.

AVMenubarAcquireMenuItemByName

[AVMenuItem](#) AVMenubarAcquireMenuItemByName ([AVMenubar](#) menubar, const char* name);

Description

Acquires the menu item with the specified language-independent menu item name (case-sensitive). This method automatically searches all menus and submenus. When you are done using the menu item, release it using [AVMenuItemRelease](#). Acquiring a menu item by name is generally reliable, because names (unlike indices) do not change as menu items are added or rearranged.

Parameters

menubar	The menubar in which the menu item is located.
name	The language-independent name of the menu item to acquire. See Menu and Menu Item Names for the language-independent names of the menu items built into Adobe Reader and Acrobat. The language-independent names of menu items added by Adobe clients are described in the technical notes for those clients.

Return Value

The menu item with the specified name. Returns **NULL** if no such menu item exists, if **menubar** is **NULL**, or if **name** is **NULL**.

Header File

AVCalls.h

Related Methods

[AVAppGetMenubar](#)
[AVMenuItemRelease](#)
[AVMenubarAcquireMenuByName](#)
[AVMenubarAcquireMenuItemByPredicate](#)
[AVMenuAcquireMenuItemByIndex](#)

Example

```
myButtonMenuItem =
    AVMenubarAcquireMenuItemByName(menubar, "Hello");
helloIndex = AVMenuGetMenuItemIndex(menu, myButtonMenuItem);
AVMenuAddMenuItem(menu, notesMenuItem, helloIndex+1);
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenubarAcquireMenuItemByPredicate

```
AVMenuItem AVMenubarAcquireMenuItemByPredicate
(AVMenubar menubar, AVMenuItemPredicate predicate,
void* clientData);
```

Description

Acquires a menu item using a user-supplied selection routine. This method may also be used to enumerate all menu items. When you are done using the menu item that is acquired, release it using [AVMenuItemRelease](#).

Parameters

menubar	The menubar containing the menu to acquire.
predicate	User-supplied function that determines which menu item is acquired. Menus items are searched depth-first. The first menu item for which predicate returns true is acquired. If predicate always returns false , all menu items will be enumerated.
clientData	Pointer to user-supplied data to pass to predicate each time it is called.

Return Value

The first menu item for which **predicate** returned **true**. Returns **NULL** if predicate never returns **true** or if **menubar** is **NULL**.

Header File

AVCalls.h

Related Methods

[AVMenuItemRelease](#)
[AVMenubarAcquireMenuByName](#)
[AVMenuAcquireMenuItemByIndex](#)
[AVAppGetMenubar](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVMenubarAddHiddenMenu

```
void AVMenubarAddHiddenMenu (AVMenubar menubar, AVMenu menu);
```

Description

Inserts a hidden menu into the menubar. Does nothing if **menubar** is **NULL** or **menu** is **NULL**.

Parameters

menubar	The menubar into which the menu is added.
menu	The menu to add.

Return Value

None

Known Exceptions

[genErrNoMemory](#)

Header File

AVCalls.h

Related Methods

[AVMenuIsHiddenOnMenubar](#)
[AVMenubarAddMenu](#)
[AVMenuNew](#)
[AVMenuRemove](#)

Example

```
AVMenu hiddenNotesMenu;

hiddenNotesMenu = AVMenuNew("Notes", "NewNotes", gExtensionID);
AVMenubarAddHiddenMenu(AVAppGetMenubar(),
    hiddenNotesMenu);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

AVMenubarAddMenu

```
void AVMenubarAddMenu (AVMenubar menubar, AVMenu menu,
ASInt32 menuIndex) ;
```

Description

Inserts a menu into the menubar. Does nothing if **menubar** is **NULL** or **menu** is **NULL**.

Parameters

menubar	The menubar into which the menu is added.
menu	The menu to add.
menuIndex	The position at which the menu is added. The left-most menu in a menubar has an index of zero. Passing a value of APPEND_MENU (see AVExpT.h .) adds the menu to the end of the menubar.

Return Value

None

Known Exceptions

[genErrNoMemory](#)

Header File

AVCalls.h

Related Methods

[AVMenuNew](#)
[AVMenuRemove](#)
[AVMenubarAddHiddenMenu](#)

Example

```
AVMenu notesMenu;

notesMenu = AVMenuNew("Notes", "NewNotes",
gExtensionID);
AVMenubarAddMenu (AVAppGetMenubar(),
notesMenu, 3);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVMenubarGetMenuIndex

ASInt32 AVMenubarGetMenuIndex (**AVMenubar** menubar, **AVMenu** menu) ;

Description

Gets the index of the specified menu in the menubar.

Parameters

menubar	The menubar in which the menu is located.
menu	The menu whose index is obtained.

Return Value

The specified menu's index. Returns **BAD_MENU_INDEX** if the menu is not in the menubar, is a submenu, if **menubar** is **NULL**, or if **menu** is **NULL**.

Header File

AVCalls.h

Related Methods

[AVAppGetMenubar](#)

Example

```
AVMenu fileMenu;
fileMenu = AVMenubarAcquireMenuByIndex(
    AVAppGetMenubar(), 1);
if (AVMenubarGetMenuIndex(fileMenu) != 1)
    AAlertNote("Odd.");
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVMenubarGetNumMenus

[AVTArraySize](#) AVMenubarGetNumMenus ([AVMenubar](#) menubar) ;

Description

Gets the number of menus in **menubar**.

Parameters

menubar	The menubar for which the number of menus is obtained.

Return Value

The number of menus in the menubar, not including submenus. Returns 0 if **menubar** is **NULL**.

Header File

AVCalls.h

Related Methods

[AVAppGetMenubar](#)

[AVMenubarAcquireMenuByIndex](#)

Example

```
AVMenubar bar = AVAppGetMenubar();
ASInt32 numMenus = AVMenubarGetNumMenus(bar);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVMenubarHide

```
void AVMenubarHide (AVMenubar menubar);
```

Description

Hides the menubar.

Parameters

menubar	The menubar to hide.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVMenubarShow](#)

[AVAppGetMenubar](#)

Example

```
AVMenubarHide (AVAppGetMenubar () );
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVMenubarShow

```
void AVMenubarShow (AVMenubar menubar);
```

Description

Shows the menubar.

Parameters

menubar	The menubar to show.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVMenubarHide](#)

[AVAppGetMenubar](#)

Example

```
AVMenubarShow (AVAppGetMenubar ()) ;
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVMenuItem

AVMenuItemAcquire

```
AVMenuItem AVMenuItemAcquire (AVMenuItem menuItem);
```

Description

Acquires a menu item. Increments the menu item's reference count.

Parameters

menuItem	The menu item to acquire.

Return Value

The menu item acquired.

Header File

AVCalls.h

Related Methods

[AVMenuItemRelease](#)
[AVMenuubarAcquireMenuItemByPredicate](#)
[AVMenuubarAcquireMenuItemByName](#)
[AVMenuAcquireMenuItemByIndex](#)

Example

```
AVMenuItem tempItem;
ASInt32 i;

for(i=0;i< AVMenuGetNumMenuItems(
    editMenu);i++){
    tempItem = AVMenuAcquireMenuItemByIndex(
        editMenu, i);
    if (ASAtomFromString("Paste") ==
        AVMenuItemGetName(tempItem)){
        AVMenuItemRemove(tempItem);
        break;
    }
    AVMenuItemRelease(tempItem);
}
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenuItemAcquireSubmenu

[AVMenu](#) AVMenuItemAcquireSubmenu ([AVMenuItem](#) menuItem) ;

Description

Acquires the submenu attached to the specified menu item, if there is one. When you are done with the submenu, release it using [AVMenuRelease](#).

Parameters

menuItem	The menu items whose submenu is obtained.

Return Value

The specified menu item's submenu. Returns **NULL** if **menuItem** is **NULL** or if **menuItem** does not have a submenu.

Header File

AVCalls.h

Related Methods

[AVMenuAcquire](#)

[AVMenuRelease](#)

Example

```
AVMenuItem tempItem;
AVMenu prefsMenu;
ASInt32 i;
buf[20];

for(i=0;i< AVMenuGetNumMenuItems(
    editMenu);i++){
    tempItem =
        AVMenuAcquireMenuItemByIndex(
            editMenu, i);
    AVMenuItemGetTitle(tempItem, buf,
        sizeof(buf));
    if(!strcmp(buf, "Preferences"))
        break;
    AVMenuItemRelease(tempItem);
}
prefsMenu = AVMenuItemAcquireSubmenu(
    tempItem);
/* now add items to prefsMenu */
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVMenuItemClone

```
AVMenuItem AVMenuItemClone (AVMenuItem menuItem) ;
```

Description

Creates a new menu item object using an existing menu item as a template. If the menu item contains a submenu, that submenu is also copied.

Parameters

menuItem	The menu item to copy.

Return Value

The new menu item object.

Header File

`AVCalls.h`

Related Methods

[AVMenuItemNew](#)

[AVMenuItemNewWithASText](#)

Product

reader, base, pro

Availability

Available if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

AVMenuItemExecute

```
void AVMenuItemExecute (AVMenuItem menuItem);
```

Description

Executes a menu item's [AVExecuteProc](#). Does nothing if `menuItem` is `NULL`, if `menuItem` has no [AVExecuteProc](#), or if `menuItem` is not enabled.

You cannot execute a menu item that has a submenu (for example, the **Pages** menu item in the **Document** menu).

Parameters

menuItem	The menu item to execute.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVMenuItemSetExecuteProc](#)
[AVMenuItemIsEnabled](#)

Example

```
AVMenuItem oldItem;
...
if (oldItem)
    AVMenuItemExecute (oldItem);
```

Product

reader, base, pro

Availability

Available if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

AVMenuItemGetLongOnly

```
ASBool AVMenuItemGetLongOnly (AVMenuItem menuItem);
```

Description

Gets the flag indicating whether a menu item is visible only in long-menus mode.

NOTE: In Acrobat 3.0 and later, this is irrelevant, since there is no long/short menus option.

Parameters

menuItem	The menu item whose flag is obtained.

Return Value

true if the menu item is visible only in long-menus mode. **false** if the menu item is visible in both long and short menus, or if **menuItem** is **NULL**.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVMenuItemGetName

```
ASAtom AVMenuItemGetName (AVMenuItem menuItem);
```

Description

Gets the atom for the language-independent name of the menu item.

Parameters

menuItem	The menu item whose language-independent name is obtained.

Return Value

The **ASAtom** corresponding to the name of the specified menu item, or **ASAtomNull** if **menuItem** is **NULL**. The **ASAtom** can be converted to a string using [ASAtomGetString](#).

Header File

AVCalls.h

Related Methods

[AVMenuItemGetTitle](#)

Example

```
AVMenuItem tempItem;
ASInt32 i;

for(i=0; i< AVMenuGetNumMenuItems(
    editMenu); i++){
    tempItem =
        AVMenuAcquireMenuItemByIndex(
            editMenu, i);
    if (ASAtomFromString("Paste") ==
        AVMenuItemGetName(tempItem)) {
        AVMenuItemSetTitle(tempItem, "Glue");
        AVMenuItemRelease(tempItem);
        break;
    }
    AVMenuItemRelease(tempItem);
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVMenuItemGetParentMenu

```
AVMenu AVMenuItemGetParentMenu (AVMenuItem menuItem);
```

Description

Gets the menu in which the specified menu item appears.

Parameters

menuItem	The menu item whose parent menu is obtained.

Return Value

The menu in which the specified menu item appears. Returns **NULL** if this menu item is not in a menu.

Header File

AVCalls.h

Related Methods

[AVMenuGetParentMenuItem](#)
[AVMenuItemAcquireSubmenu](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenuItemGetShortcut

```
ASBool AVMenuItemGetShortcut (AVMenuItem menuItem, char* key,
ASInt16* flags);
```

Description

Gets the shortcut key for the specified menu item.

Parameters

menuItem	The menu item whose shortcut is obtained.
key	<i>(Filled by the method)</i> The key that is a shortcut for the menu item, an ASCII character. The value NO_SHORTCUT (see <code>AVExpT.h</code>) indicates that the menu item has no shortcut.
flags	<i>(Filled by the method)</i> Modifier keys, if any, used as part of the shortcut. Must be an OR of the Modifier Keys values, except that AV_COMMAND will never be present.

Return Value

true if **menuItem** is not **NULL** and **menuItem** has a shortcut, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVMenuItemNew](#)

Example

```
AVMenuItem item;
ASInt16 flags;
char* key;
if (AVMenuItemGetShortcut (item, key,
&flags))
    AAlertNote ("shortcut exists");
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenuItemGetTitle

[AVTArraySize](#) AVMenuItemGetTitle ([AVMenuItem](#) menuItem,
char* buffer, [AVTArraySize](#) bufferSize);

Description

Gets a menu item’s title, which is the string that appears in the user interface.

Parameters

menuItem	The menu item whose title is obtained.
buffer	<i>(Filled by the method)</i> A buffer to hold the null-terminated name. If title is NULL , returns the length of the menu item’s name, but does not fill the buffer.
bufferSize	The maximum number of characters that can be placed into title . If the name is longer than this, only the first bufferSize – 1 characters are placed into buffer , and the buffer is null-terminated.

Return Value

The length of the title. Returns 0 if **menuItem** is **NULL**.

Header File

AVCalls.h

Related Methods

[AVMenuItemSetTitle](#)
[AVMenuItemGetName](#)
[AVMenuItemGetTitleAsASText](#)

Example

```
AVMenuItem tempItem;  
ASInt32 i;  
buf[20];  
  
for(i=0;i< AVMenuGetNumMenuItems(editMenu);i++){  
    tempItem = AVMenuAcquireMenuItemByIndex(editMenu, i);  
    AVMenuItemGetTitle(tempItem, buf, sizeof(buf));  
    if(!strcmp(buf, "Paste"))  
        AVMenuItemSetTitle(tempItem, "Glue");  
    AVMenuItemRelease(tempItem);  
}
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenuItemGetTitleAsASText

```
void AVMenuItemGetTitleAsASText (AVMenuItem menuItem,
    AText title);
```

Description

Gets the menu item’s title as it appears in the user interface, as an **AText** object.

Parameters

menuItem	The menu item whose title is obtained.
title	<i>(Filled by the method)</i> The text object containing the title.

Return Value

None.

Header File

AVCalls.h

Related Methods

- AVMenuGetTitleAsASText
- AVMenuItemSetTitleWithASText
- AVMenuItemGetName
- AVMenuItemGetTitle

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVMenuItemIsEnabled

```
ASBool AVMenuItemIsEnabled (AVMenuItem menuItem);
```

Description

Tests whether the specified menu item is enabled.

Parameters

menuItem	The menu item whose enabled flag is obtained.

Return Value

true if **menuItem** is enabled, if **menuItem** is **NULL**, or if **menuItem** has no **AVComputeEnabledProc**.

false if the menu item is disabled or its **AVComputeEnabledProc** raises an exception.

Header File

AVCalls.h

Related Methods

[AVMenuItemSetComputeEnabledProc](#)

Example

```
if (AVMenuItemIsEnabled(item)
    AVMenuItemExecute(item);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVMenuItemIsMarked

```
ASBool AVMenuItemIsMarked (AVMenuItem menuItem);
```

Description

Tests whether **menuItem** is marked (for example, appears with a check mark).

Parameters

menuItem	The menu item whose marked state is obtained.

Return Value

true if **menuItem** is marked. **false** if **menuItem** is **NULL**, if the menu item does not have an **AVComputeMarkedProc**, or if it raises an exception.

Header File

AVCalls.h

Related Methods

[AVMenuItemSetComputeMarkedProc](#)

Example

```
if (AVMenuItemIsMarked(item))
    AVMenuItemExecute(item);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVMenuItemIsVisible

```
ASBool AVMenuItemIsVisible (AVMenuItem menuItem);
```

Description

Tests whether the specified menu item is visible on its parent menu. The visibility state is calculated using the [AVComputeVisibleProc](#).

Parameters

menuItem	The menu item whose visibility state is tested.

Return Value

true if **menuItem** is visible, or if **menuItem** has no [AVComputeVisibleProc](#).

false if the menu item is not visible, or if its [AVComputeVisibleProc](#) raises an exception (this is not recommended), or if **menuItem** is **NULL**.

Header File

AVCalls.h

Related Methods

[AVMenuItemSetComputeVisibleProc](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVMenuItemNew

```
AVMenuItem AVMenuItemNew (const char* title, const char* name,
AVMenu submenu, ASBool longMenusOnly, char shortcut,
AVFlagBits16 flags, AVIcon icon, ASExtension owner);
```

Description

Creates and acquires a new **AVMenuItem**. The menu item can be added to a menu using **AVMenuAddMenuItem**.

Release the **AVMenuItem** using **AVMenuItemRelease** after it has been added to a menu.

Parameters

title	The string shown in the user interface for this menu item. Use a hyphen to create a separator menu item. This value is also returned by AVMenuItemGetTitle . In Windows, an ampersand (&) character in the string results in underlining the character after it on the menu item.
name	The language-independent name of the menu item to create. This is the value returned by AVMenuItemGetName . name must <i>not</i> contain any spaces. Client developers should prefix the names of menu items they add with the name of their client and a colon, to avoid collisions in the menu item name space. For example, a client named myPlug might add menu items named myPlug:Scan and myPlug:Find .
submenu	Submenu (if any) for which this menu item is the parent. Pass NULL if this menu item does not have a submenu.
longMenusOnly	<i>(Ignored in Acrobat 3.0 or later)</i> If <i>true</i> , the menu item is visible only when the user selects "Full Menus." If <i>false</i> , the menu item is visible for both "Full Menus" and "Short Menus" modes.
shortcut	The key to use as a shortcut for the menu item, an ASCII character. Use NO_SHORTCUT (see AVExpT.h) if the menu item has no shortcut. The Acrobat viewer does not check for conflicts between shortcuts. The consequences of multiple menu items having the same shortcut is undefined. In Windows, the shortcut is not displayed for any menu item that also has an icon , although the shortcut will work.

flags	Modifier keys, if any, used as part of the shortcut. Must be an OR of the Modifier Keys values, except that AV_COMMAND cannot be specified.
icon	The icon to show in the menu item, or NULL if no icon is shown. In Mac OS, icon is a handle to a standard SICN resource. In Windows, icon is a 24×24 sample monochrome HBITMAP .
owner	The gExtensionID extension registering the menu item.

Return Value

The newly created menu item.

Header File

AVCalls.h

Related Methods

[AVMenuAddMenuItem](#)

[AVMenuItemRelease](#)

Example

```
ASCallback doHelloAlertCallback;
AVMenuItem myButtonMenuItem;

doHelloAlertCallback =
    ASCallbackCreateProto(AVExecuteProc,
        &doHelloAlert);
/* shortcut is CTRL-SHIFT-H */
myButtonMenuItem = AVMenuItemNew(
    "&Hello World", "Hello", NULL, false,
    "H", AV_CONTROL || AV_SHIFT, NULL,
    gExtensionID);
AVMenuItemSetExecuteProc(myButtonMenuItem,
    doHelloAlertCallback, NULL);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenuItemNewWithASText

```
AVMenuItem AVMenuItemNew (ASConstText title, const char* name,
AVMenu submenu, ASBool longMenusOnly, char shortcut,
AVFlagBits16 flags, AVIcon icon, ASExtension owner);
```

Description

Creates and acquires a new **AVMenuItem**. The menu item can be added to a menu using **AVMenuAddMenuItem**.

Release the **AVMenuItem** using **AVMenuItemRelease** after it has been added to a menu.

Parameters

title	A constant text object containing the string shown in the user interface for this menu item. Use a hyphen to create a separator menu item. This value is also returned by AVMenuItemGetTitleAsASText . In Windows, an ampersand (&) character in the string results in underlining the character after it on the menu item.
name	The language-independent name of the menu item to create. This is the value returned by AVMenuItemGetName . name must <i>not</i> contain any spaces. Client developers should prefix the names of menu items they add with the name of their client and a colon, to avoid collisions in the menu item name space. For example, a client named myPlug might add menu items named myPlug:Scan and myPlug:Find .
submenu	Submenu (if any) for which this menu item is the parent. Pass NULL if this menu item does not have a submenu.
longMenusOnly	<i>(Ignored in Acrobat 3.0 or later)</i> If <i>true</i> , the menu item is visible only when the user selects "Full Menus." If <i>false</i> , the menu item is visible for both "Full Menus" and "Short Menus" modes.
shortcut	The key to use as a shortcut for the menu item, an ASCII character. Use NO_SHORTCUT (see AVExpT.h) if the menu item has no shortcut. The Acrobat viewer does not check for conflicts between shortcuts. The consequences of multiple menu items having the same shortcut is undefined. In Windows, the shortcut is not displayed for any menu item that also has an icon , although the shortcut will work.

flags	Modifier keys, if any, used as part of the shortcut. Must be an OR of the Modifier Keys values, except that AV_COMMAND cannot be specified.
icon	The icon to show in the menu item, or NULL if no icon is shown. In Mac OS, icon is a handle to a standard SICN resource. In Windows, icon is a 24×24 sample monochrome HBITMAP .
owner	The gExtensionID extension registering the menu item.

Return Value

The newly created menu item.

Header File

AVCalls.h

Related Methods

[AVMenuAddMenuItem](#)
[AVMenuItemGetTitleAsASText](#)
[AVMenuItemRelease](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVMenuItemRelease

```
void AVMenuItemRelease (AVMenuItem menuItem);
```

Description

Releases a menu item. Decrements the reference count and destroys the menu item if the count is zero.

Parameters

menuItem	The menu item to release.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVMenuItemAcquire](#)
[AVMenubarAcquireMenuItemByPredicate](#)
[AVMenubarAcquireMenuItemByName](#)
[AVMenuAcquireMenuItemByIndex](#)

Example

```
AVMenuItem tempItem;
ASInt32 i;

for (i=0; i< AVMenuGetNumMenuItems (editMenu)
; i++) {
    tempItem =
        AVMenuAcquireMenuItemByIndex (
            editMenu, i);
    if (ASAtomFromString ("Paste") ==
        AVMenuItemGetName (tempItem) ) {
        AVMenuItemRemove (tempItem);
        break;
    }
    AVMenuItemRelease (tempItem);
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenuItemRemove

```
void AVMenuItemRemove (AVMenuItem menuItem);
```

Description

Removes a menu item from the menu hierarchy and releases it. Does nothing if `menuItem` is `NULL`.

Keyboard accelerators for the Acrobat viewer’s built-in menu items will always work, even if the menu item is removed.

Parameters

<code>menuItem</code>	The menu item to remove.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVMenuItemAcquire](#)
[AVMenuItemRelease](#)

Example

```
ASInt32 i;

for(i=0;i< AVMenuGetNumMenuItems(
    editMenu);i++){
    tempItem = AVMenuItemAcquireMenuItemByIndex(
        editMenu, i);
    if (ASAtomFromString("Paste") ==
        AVMenuItemGetName(tempItem)){
        AVMenuItemRemove(tempItem);
        break;
    }
    AVMenuItemRelease(tempItem);
}
```

Product

reader, base, pro

Availability

Available if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

AVMenuItemSetComputeEnabledProc

```
void AVMenuItemSetComputeEnabledProc (AVMenuItem menuItem,
AVComputeEnabledProc proc, void* data);
```

Description

Sets the user-supplied procedure to call to determine whether the menu item is enabled. Does nothing if `menuItem` is `NULL`.

Parameters

<code>menuItem</code>	The menu item whose <code>AVComputeEnabledProc</code> is set.
<code>proc</code>	User-supplied callback to call whenever the Acrobat viewer needs to know whether <code>menuItem</code> should be enabled. This procedure is called every time the menu is displayed, so it should not do compute-time-intensive processing.
<code>data</code>	Pointer to user-supplied data to pass to <code>proc</code> each time it is called.

Return Value

None

Header File

AVCalls.h

Related Methods

- [AVMenuItemIsEnabled](#)
- [AVMenuItemSetComputeMarkedProc](#)
- [AVMenuItemSetExecuteProc](#)

Example

```
ASCallback pDocExistEnableCallback; //Will hold ptr to callback
AVMenuItem myButtonMenuItem;

pDocExistEnableCallback =
    ASCallbackCreateProto(AVComputeEnabledProc,
        DocExistEnableCallback);
myButtonMenuItem = AVMenuItemNew(
    "Hello World", "Hello", NULL, false,
    NO_SHORTCUT, 0, NULL, gExtensionID);
AVMenuItemSetComputeEnabledProc(
    myButtonMenuItem,
    pDocExistEnableCallback, NULL);
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenuItemSetComputeMarkedProc

```
void AVMenuItemSetComputeMarkedProc (AVMenuItem menuItem,
AVComputeMarkedProc proc, void* data);
```

Description

Sets the user-supplied procedure that determines whether the menu item appears with a check mark. Does nothing if **menuItem** is **NULL**. If the menu item has no **AVExecuteProc** (see **AVMenuItemSetExecuteProc**), its **AVComputeMarkedProc** is never called. To avoid this, add an **AVExecuteProc** that does nothing, and if you wish the menu item to gray out, also add an **AVComputeEnabledProc** that always returns **false**.

Parameters

menuItem	The menu item whose AVComputeMarkedProc is set.
proc	User-supplied callback to call whenever the Acrobat viewer needs to know whether the menuItem should be marked.
data	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVMenuItemIsMarked](#)
[AVMenuItemSetExecuteProc](#)
[AVMenuItemSetComputeEnabledProc](#)

Example

```
ASCallback markedProcCallback;
AVMenuItem myButtonMenuItem;
markedProcCallback =
    ASCallbackCreateProto(AVExecuteProc, &markedProc);
myButtonMenuItem = AVMenuItemNew ("Hello World", "Hello", NULL, false,
    NO_SHORTCUT, 0, NULL, gExtensionID);
AVMenuItemSetComputeMarkedProc (
    myButtonMenuItem, markedProcCallback, NULL);
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenuItemSetComputeVisibleProc

```
void AVMenuItemSetComputeVisibleProc (AVMenuItem menuItem,
AVComputeVisibleProc proc, void* clientData);
```

Description

Sets the user-supplied procedure that determines whether the menu item appears when its parent menu is opened. Does nothing if menuItem is **NULL**.

Parameters

menuItem	The menu item whose visibility procedure is set.
proc	User-supplied procedure to call when the Acrobat viewer needs to know if the menu item is visible.
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVMenuItemIsVisible](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION**(in `PIRequir.h`) is set to **0x00060000** or higher.

AVMenuItemSetExecuteProc

```
void AVMenuItemSetExecuteProc (AVMenuItem menuItem,
AVExecuteProc proc, void* data);
```

Description

Sets the user-supplied procedure to execute whenever the menu item is chosen. Does nothing if **menuItem** is **NULL**. Clients must not set the execute procedure of the Acrobat viewer's built-in menu items.

Parameters

menuItem	The menu item whose execute procedure is set.
proc	User-supplied callback to call whenever menuItem is selected.
data	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVMenuItemExecute](#)
[AVMenuItemSetComputeMarkedProc](#)
[AVMenuItemSetComputeEnabledProc](#)

Example

```
ASCallback markedProcCallback;
AVMenuItem myButtonMenuItem;

doHelloAlertCallback =
    ASCallbackCreateProto(AVExecuteProc,
        &doHelloAlert);
myButtonMenuItem = AVMenuItemNew(
    "Hello World", "Hello", NULL, false,
    NO_SHORTCUT, 0, NULL, gExtensionID);
AVMenuItemSetExecuteProc(myButtonMenuItem,
    doHelloAlertCallback, NULL);
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenuItemSetTitle

```
void AVMenuItemSetTitle (AVMenuItem menuItem,
                        const char* title);
```

Description

Sets a menu item's title, which is the string that appears in the user interface. Use this method to manage menu items whose titles change (such as "show/hide fooWindow"), instead of inserting and removing menu items on the fly.

Parameters

menuItem	The menu item whose title is set.
title	The new menu title. It must be a null-terminated string.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVMenuItemGetTitle](#)
[AVMenuItemGetName](#)

Example

```
AVMenuItem tempItem;
ASInt32 i;
char buf[20];

for(i=0;i< AVMenuGetNumMenuItems(
    editMenu);i++){
    tempItem =
        AVMenuAcquireMenuItemByIndex(
            editMenu, i);
    AVMenuItemGetTitle(tempItem, buf,
        sizeof(buf));
    if(!strcmp(buf, "Paste"))
        AVMenuItemSetTitle(tempItem, "Glue");
    AVMenuItemRelease(tempItem);
}
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVMenuItemSetTitleWithASText

```
void AVMenuItemSetTitleWithASText (AVMenuItem menuItem,  
ASConstText title);
```

Description

Sets a menu item’s title, which is the string that appears in the user interface, from a constant text object. Use this method to manage menu items whose titles change (such as “show/hide fooWindow”), instead of inserting and removing menu items on the fly.

Parameters

menuItem	The menu item whose title is set.
title	The new menu title, as a constant text object.

Return Value

None

Header File

AVCalls.h

Related Methods

- AVMenuItemGetTitleAsASText
- AVMenuItemGetName

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVPageView

AVPageViewAcquireMachinePort

```
void* AVPageViewAcquireMachinePort (AVPageView pageView);
```

Description

Acquires the platform-specific object needed to draw into the Acrobat viewer's document window using a platform's native graphics calls. When done, release it using [AVPageViewReleaseMachinePort](#).

Parameters

pageView	The AVPageView whose platform-dependent port is acquired.

Return Value

A platform-dependent value.

- In Mac OS, it is a **GrafPtr**.
- In Windows, it is a [WinPort](#).
- In UNIX, it is a **Widget**.

Header File

AVCalls.h

Related Methods

[AVPageViewReleaseMachinePort](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVPageViewAppearanceGetAVMatrix

```
void AVPageViewAppearanceGetAVMatrix (AVPageView PageView,
AVFlagBits32 flags, CosObj appear, ASFixedRect* ar,
ASFixedMatrix* mr);
```

Description

Calculates a matrix for use with [AVPageViewDrawCosObj](#) or [AVPageViewDrawCosObjEx](#) that leaves the appearance unrotated and un-zoomed as specified by **flags**. This is typically used in conjunction with drawing an annotation appearance represented by **appear**.

Parameters

PageView	The page view for which the matrix is calculated.
flags	Annotation flags obtained by PDAnnotGetFlags .
appear	Appearance of the object, which is a Form XObject .
ar	Bounding rectangle for appear .
mr	<i>(Filled by the method)</i> The transformation matrix.

Header File

AVCalls.h

Related Methods

```
AVPageViewDrawCosObj
AVPageViewDrawCosObjEx
AVPageViewTransformRectRZ
```

Example

```
ASFixedMatrix mr;
ASFixedRect ar;
CosObj aprObj;
AVRect avr;
ASUns32 flags = PDAnnotGetFlags(annot);
/* Draw an annotation appearance */
AVPageViewAppearanceGetAVMatrix(pageView, flags, aprObj, &ar, &mr);
AVPageViewDrawCosObjEx(pageView, aprObj, &avr, &mr);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION**(in `PIRequir.h`) is set to **0x00040000** or higher.



The flag-value numeric type has changed in **0x00060000**.



AVPageViewBeginOperation

```
void AVPageViewBeginOperation (AVPageView pageView);
```

Description

Increments an internal variable. Neither drawing nor [AVPageViewDidChange](#) notifications will occur as long as the variable has a value greater than zero. In addition, frames are not pushed onto the view history stack.

Parameters

pageView	The page view whose SuspendDraw variable is incremented.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewEndOperation](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVPageViewClearFocusAnnot

```
ASBool AVPageViewClearFocusAnnot (AVPageView pageView);
```

Description

Removes the focus from the currently selected annotation.

Parameters

pageView	The page view holding the focus.

Return Value

false if there was no focus annotation in effect when the method was called; **true** if there was.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVPageViewDevicePointToPage

```
void AVPageViewDevicePointToPage (AVPageView pageView,
    AVDevCoord x, AVDevCoord y, ASFixedPoint* p);
```

Description

Transforms a point’s coordinates from device space to user space.

Parameters

pageView	Page view for which the point’s coordinates are transformed.
x	x–coordinate of the point to transform, specified in device space coordinates.
y	y–coordinate of the point to transform, specified in device space coordinates.
p	(Filled by the method) Pointer to a point whose user space coordinates correspond to x and y .

Return Value

None

Header File

AVCalls.h

Related Methods

- AVPageViewPointToDevice
- AVPageViewRectToDevice
- AVPageViewDeviceRectToPage

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewDeviceRectToPage

```
void AVPageViewDeviceRectToPage (AVPageView pageView,
const AVRect* rect, ASFixedRect* p);
```

Description

Transforms a rectangle from device space to user space coordinates. The resulting **ASFixedRect** is “normal,” that is, **left** < **right** and **bottom** < **top**.

Parameters

pageView	Page view for which the rectangle is transformed.
rect	Pointer to a device space rectangle whose coordinates are transformed to user space.
p	<i>(Filled by the method)</i> Pointer to a user space rectangle corresponding to rect .

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewPointToDevice](#)
[AVPageViewDevicePointToPage](#)
[AVPageViewDeviceRectToPageRZ](#)
[AVPageViewRectToDevice](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewDeviceRectToPageRZ

```
void AVPageViewDeviceRectToPageRZ (AVPageView pageView,
AVTFlagBits flags, AVDevCoord xHot, AVDevCoord yHot,
const AVDevRect* src, ASFixedRect* dest);
```

Description

Transforms an annotation’s rectangle from device space to user space coordinates, allowing for the annotation’s attributes of whether it should zoom or rotate when the page is zoomed or rotated. It also specifies a point that can remain in the view.

Parameters

pageView	Page view for which the rectangle is transformed.
flags	Flags to indicate whether the annotation rotates or zooms with the page view. These flags correspond to the annotation’s F key and can be obtained from PDAnnotGetFlags . Must be an OR of the following flags: ● pdAnnotNoZoom—Annotation does not zoom with the view. ● pdAnnotNoRotate—Annotation does not rotate with the page.
xHot	The x-coordinate of point that should remain in the view.
yHot	The y-coordinate of point that should remain in the view.
src	Pointer to a device space annotation rectangle whose coordinates are transformed to user space.
dest	<i>(Filled by the method)</i> Pointer to a user space rectangle corresponding to src .

Return Value

None

Header File

AVCalls.h

Related Methods

- [AVPageViewAppearanceGetAVMatrix](#)
- [AVPageViewDeviceRectToPage](#)
- [AVPageViewRectToDevice](#)
- [AVPageViewTransformRectRZ](#)

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

NOTE: The coordinate and flag numeric types changed in **0x00060000**.

AVPageViewDeviceToInfo

```
void AVPageViewDeviceToInfo (AVPageView pageView, AVlCoord x,
AVlCoord y, ASFixedPoint* info);
```

Description

Translates the given point from the device space coordinate system to the info space coordinate system.

Info space is the coordinate system used by the info palette to provide the user with visual feedback during layout operations. The origin of info space for a given page is the upper left corner of the page. The x-axis is horizontal and increases from left to right. The y-axis is vertical and increases from top to bottom. Units are measured in points. Conceptually, info space is the same as user space, with the y-axis flipped and the origin anchored at the upper left of the page.

Parameters

pageView	The page view holding the focus.
x	The x-coordinate in device space.
y	The y-coordinate in device space.
info	(Filled by the method) The translated point.

Return Value

None

Header File

AVCalls.h

Related Methods

- AVPageViewInfoToDevice
- AVPageViewInfoToPoint
- AVPageViewPointToInfo

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewDoPopupMenu

```
AVMenuItem AVPageViewDoPopupMenu (AVPageView pageView,
AVMenu menu, AVDevCoord xHit, AVDevCoord yHit,
ASBool rightMouse, AVMenuIndex choice);
```

Description

Displays the given **AVMenu** as a popup menu anchored at **xHit** and **yHit**, which are in device coordinates relative to **pageView**.

Parameters

pageView	The page view in which the menu appears.
menu	The displayed menu.
xHit	The x-coordinate of the upper left corner of the menu.
yHit	The y-coordinate of the upper left corner of the menu.
rightMouse	true if the right mouse button (where applicable) was used to invoke the popup, false otherwise.
choice	The index of the AVMenuItem that should appear under the mouse at pop-up time.

Return Value

The menu item selected from **menu**.

Header File

AVCalls.h

Related Methods

[AVToolButtonGetMenu](#)
[AVToolButtonSetMenu](#)
[AVMenuDoPopUp](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewDragOutNewRect

```
void AVPageViewDragOutNewRect (AVPageView pageView,
                               AVDevCoord xStart, AVDevCoord yStart, AVDevRect* resultRect);
```

Description

Allows the user to drag out a new rectangle. Call this method when the user clicks at some point and needs to create a rectangle. The method returns when the user releases the mouse button.

Parameters

pageView	The page view in which the rectangle is created.
xStart	The x-coordinate of the point where the user initially clicked, specified in device space coordinates.
yStart	The y-coordinate of the point where the user initially clicked, specified in device space coordinates.
resultRect	<i>(Filled by the method)</i> Pointer to the rectangle that the user dragged out, specified in device space coordinates.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewDragRect](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewDragOutNewRectSnapped

```
AVTFlagBits AVPageViewDragOutNewRectSnapped
(AVPageView pageView, AVDevCoord xStart, AVDevCoord yStart,
AVDevRect* resultRect, AVCursor* cursorArray,
AVTSmallArraySize nCursors);
```

Description

Drags out a rectangle anchored at the given point. The rectangle will be snapped to the layout grid.

Parameters

pageView	The page view in which the rect will be created.
xStart	x-coordinate of the anchor point.
yStart	y-coordinate of the anchor point.
resultRect	(Filled by the method) The resulting rectangle generated by the user.
cursorArray	A pointer to an array of AVCursors . This value is used in conjunction with nCursors and is treated as follows: 1. If NULL , the default cursor will be used during the drag. nCursors is ignored. 2. If non- NULL and nCursors is 1 , the supplied cursor will replace the default cursor during the drag. 3. If non- NULL and nCursors is 2 , the first cursor will replace the default cursor during the drag, and the second cursor will be used if the user presses the control key (Windows) or option key (Mac).
nCursors	Number of cursors supplied in cursorArray .

Return Value

A bit-wise OR of the [Modifier Keys](#) that were pressed as the rect was dragged out. See [AVSysGetModifiers](#).

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



NOTE: The coordinate and flag numeric types changed in 0x00060000.

AVPageViewDragRect

```
void AVPageViewDragRect (AVPageView pageView,
AVDevCoord xStart, AVDevCoord yStart, AVDevRect* startRect,
AVDevRect* resultRect, ASInt32 dragType, AVDevRect* extrema);
```

Description

Allows the user to move or resize a rectangle. Call this method when the user clicks on a rectangle to modify. It returns after the user releases the mouse button.

Parameters

pageView	The page view in which the rectangle is located.
xStart	The x-coordinate of the point where the user initially clicked, specified in device space coordinates.
yStart	The y-coordinate of the point where the user initially clicked, specified in device space coordinates.
startRect	Pointer to the initial rectangle, which is to moved or resized, specified in device space coordinates.
resultRect	<i>(Filled by the method)</i> Pointer to the resized or moved rectangle, specified in device space coordinates.
dragType	One of the AVDragType constants.
extrema	<i>(May be NULL)</i> If NULL , the rectangle returned by AVPageViewGetGrayRect is used instead. Pointer to the rectangle specifying the maximum limits of the user-dragged rectangle. The user cannot grow the rectangle outside extrema , specified in device space coordinates. Pass NULL if you do not wish to limit the changes the user is making. extrema is ignored if dragType is 0.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewDragOutNewRect](#)

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewDragRectSnapped

```
AVTFlagBits AVPageViewDragRectSnapped (AVPageView pageView,
AVDevCoord xStart, AVDevCoord yStart, AVDevRect* startRect,
AVDevRect* resultRect, ASInt32 dragType, AVDevRect* extrema,
AVCursor* cursorArray, AVTSmallArraySize nCursors);
```

Description

Allows the user to move or resize an existing rectangle. If the user has enabled **Snap To Grid** from the main menu, the resulting rectangle will be aligned to the layout grid according to the **dragType** parameter.

NOTE: Superseded in Acrobat 6.0 by [AVPageViewDragRectSnappedEx](#).

Parameters

pageView	The page view in which the rectangle resides.
xStart	The x-coordinate of the point where the user initially clicked, specified in device space coordinates.
yStart	The y-coordinate of the point where the user initially clicked, specified in device space coordinates.
startRect	The initial position of the rectangle.
resultRect	<i>(Filled by the method)</i> The position of the rectangle at the end of the operation.
dragType	One of the AVDragType constants. These determine whether the drag is a resize or a move operation. To move the rectangle, specify one of the following: kAVDragRect—the corner of the rectangle that is closest to (xStart,yStart) is snapped to the grid, and dragging begins from that point. kAVDragSnapToXXX - the corner or edge specified by "XXX" (TopLeft, Top, and so on) is snapped, and dragging begins from that point. To resize the rectangle, specify one of the following: kAVDragXXX - the corner or edge specified by "XXX" (TopLeft, Top, and so on) is dragged, and the opposite corner or edge is used as an anchor point.

extrema	(May be NULL) The rectangle the drag operation is restricted to. If NULL , the rectangle returned by AVPageViewGetGrayRect is used instead. Use this to restrict the drag operation to a bounding rectangle. May be NULL , in which case the bounding rectangle of the page view is used.
cursorArray	A pointer to an array of AVCursors . This value is used in conjunction with nCursors and is treated as follows: 1. If NULL, the default cursor will be used during the drag. nCursors is ignored. 2. If non-NULL and nCursors is 1, the supplied cursor will replace the default cursor during the drag. 3. If non-NULL and nCursors is 2, the first cursor will replace the default cursor during the drag, and the second cursor will be used if the user presses the control key (Windows) or option key (Mac).
nCursors	Number of cursors supplied in cursorArray .

Return Value

A bit-wise OR of the [Modifier Keys](#) that were pressed as the rect was dragged.

Header File

AVCalls.h

Product

reader, base, pro

Related Methods

[AVPageViewDragRectSnappedEx](#)
[AVPageViewSnapPointEx](#)
[AVSysGetModifiers](#)

Availability

Available if **PI_ACROVIEW_VERSION** (in [PIRequir.h](#)) is set to **0x00050000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewDragRectSnappedEx

[AVTFlagBits](#) AVPageViewDragRectSnappedEx
(const [AVDragRectParams](#) *params)

Description

Allows the user to move or resize an existing rectangle. This version of the method allows you to specify your own drawing procedure.

NOTE: Supersedes [AVPageViewDragRectSnapped](#) in Acrobat 6.0.

Parameters

<code>params</code>	The parameters with which to perform the drag operation. These include the page view object.

Return Value

A bit-wise OR of the [Modifier Keys](#) that were in effect when the mouse button was released.

Header File

AVCalls.h

Related Methods

[AVPageViewDragRectSnapped](#)
[AVPageViewSnapPointEx](#)
[AVSysGetModifiers](#)

Product

reader, base, pro

Availability

Available if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.



AVPageViewDrawAnnotSequence

```
void AVPageViewDrawAnnotSequence (AVPageView pv, PDAnnot an,
AVDevRect* bbox);
```

Description

Draws the annotation sequence number of an annotation in a rectangle.

Parameters

pv	The page view in which the annotation is drawn.
an	The annotation whose sequence number is drawn.
bbox	Bounding box in which to draw the sequence number of the annotation.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewDrawRect](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewDrawCosObj

```
void AVPageViewDrawCosObj (AVPageView pageView, CosObj cosObj,
AVDevRect* rect);
```

Description

Draws the **CosObj** (which currently must be a Form object) and scales it to fit **rect**. This method may be used to draw an annotation appearance.

Parameters

pageView	The page view.
cosObj	The CosObj to draw.
rect	The rectangle in which to draw.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewAppearanceGetAVMatrix](#)
[AVPageViewDrawCosObjEx](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewDrawCosObjEx

```
void AVPageViewDrawCosObjEx (AVPageView pageView,
                             CosObj cosObj, AVDevRect* rect, ASFixedMatrix* matrix);
```

Description

Draws the **CosObj** (which currently must be a Form object), scales it to fit **rect**, and transforms it according to **matrix**. This method is the same as [AVPageViewDrawCosObj](#) but applies an **ASFixedMatrix** transformation to the **CosObj** once the **CosObj** has been scaled to the anamorphic dimensions of the **AVRect**. This method may be used to draw an annotation appearance.

Parameters

pageView	The page view.
cosObj	The CosObj to draw.
rect	The rectangle in which to draw.
matrix	Pointer to a matrix specifying the matrix to transform the cosObj based on rect . For example, if the AVRect is [10 20 110 220] describing a 100 x 200 area near the top left corner of the page, a ASFixedMatrix of [0.5 0 0 0.5 50 100] would scale the CosObj to 50% and center it inside the AVRect .

Header File

AVCalls.h

Related Methods

[AVPageViewAppearanceGetAVMatrix](#)
[AVPageViewDrawCosObj](#)

Example

```
ASFixedMatrix mr;
ASFixedRect ar;
CosObj aprObj;
AVRect avr;
ASUns32 flags = PDAnnotGetFlags(annot);
/* Draw an annotation appearance */
AVPageViewAppearanceGetAVMatrix(pageView, flags, aprObj, &ar, &mr);
AVPageViewDrawCosObjEx(pageView, aprObj, &avr, &mr);
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020003** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewDrawNow

```
void AVPageViewDrawNow (AVPageView pageView);
```

Description

Forces any pending updates for the specified page view to finish drawing.

Parameters

pageView	The AVPageView to redraw.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewInvalidateRect](#)

Example

```
PDPage page;
AVRect avRect;
ASInt32 annotNum;
PDAnnot anAnnot;

for (annotNum = 0; annotNum <
    PDPageGetNumAnnots(page); annotNum++) {
    anAnnot = PDPageGetAnnot (page,
        annotNum);
    if (PDAnnotGetSubtype(anAnnot) ==
        ASAtomFromString("Text")) {
        AVPageViewGetAnnotRect (pageView,
            anAnnot, &avRect);
        AVPageViewInvalidateRect (pageView,
            &avRect);
    }
}
PDPageRelease (page);
AVPageViewDrawNow (pageView);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVPageViewDrawNowWithTransition

```
void AVPageViewDrawNowWithTransition (AVPageView pageView,  
PDTrans trans);
```

Description

Forces any pending updates for the specified page view to finish drawing, using the specified transition effect.

Parameters

pageView	The AVPageView to redraw.
trans	The transition to use.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewDrawNow](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVPageViewDrawPolygon

```
void AVPageViewDrawPolygon (AVPageView pageView,
AVDevCoord* x, AVDevCoord* y, ASCount numPoints,
ASBool invert);
```

Description

Draws a polygon, filled with the color most recently set using [AVPageViewSetColor](#), or inverted.

Parameters

pageView	The page view in which the polygon is drawn.
x	The list of x-coordinates for the vertices of the polygon.
y	The list of y-coordinates for the vertices of the polygon.
numPoints	The number of vertices in the polygon.
invert	When true, inverts the polygon instead of filling it with the set color.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewDrawPolygonOutline](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVPageViewDrawPolygonOutline

```
void AVPageViewDrawPolygonOutline (AVPageView pageView,
AVDevCoord* x, AVDevCoord* y, AVDevCoord numPoints,
ASBool invert);
```

Description

Draws a stroked (not filled) or inverted polygon, using the color most recently set using [AVPageViewSetColor](#).

Parameters

pageView	The page view in which the polygon is drawn.
x	The list of x-coordinates for the vertices of the polygon.
y	The list of y-coordinates for the vertices of the polygon.
numPoints	The number of vertices in the polygon.
invert	When true, inverts the polygon instead of stroking it with the set color.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewDrawPolygon](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVPageViewDrawRect

```
void AVPageViewDrawRect (AVPageView pageView,
    const AVDevRect* rect);
```

Description

Draws a rectangle filled with the color most recently set using [AVPageViewSetColor](#).

Parameters

pageView	The page view in which the rectangle is drawn.
rect	Pointer to the rectangle to draw, specified in device space coordinates.

Return Value

None

Header File

AVCalls.h

Related Methods

- [AVPageViewDrawRectOutline](#)
- [AVPageViewDrawAnnotSequence](#)

Example

```
void AVPageViewDrawRectSnip(ParamManager * thePM){
    AVDoc avDoc = SnippetRunnerUtils::getFrontAVDoc();
    if (avDoc != NULL){
        AVPageView thePageView = AVDocGetPageView(avDoc);
        ASFixed left,top,right,bottom;
        if ((thePM->getNextParamAsFixed(left)== true)&&
            (thePM->getNextParamAsFixed(top)==true) &&
            (thePM->getNextParamAsFixed(right)==true) &&
            (thePM->getNextParamAsFixed(bottom)==true)) {
            ASFixedRect rect;
            rect.left = left;
            rect.top = top;
            rect.right = right;
            rect.bottom = bottom;

            PDColorValueRec pdVal;
            pdVal.space=PDDeviceRGB;
            pdVal.value[0]=0;
            pdVal.value[1]=0xffff;
            pdVal.value[2]=0;
            AVRect newRect;
            AVPageViewRectToDevice(thePageView,&rect, &newRect);
```



```
        AVPageViewSetColor(thePageView, &pdVal);  
        AVPageViewDrawRect(thePageView, &newRect);  
    }  
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVPageViewDrawRectOutline

```
void AVPageViewDrawRectOutline (AVPageView pageView,
const AVDevRect* rect, AVDevSize lineWidth,
ASFixed* dashArray, AVTArraySize arrayLen);
```

Description

Draws a stroked, but not filled, rectangle using the color most recently set using [AVPageViewSetColor](#).

Parameters

pageView	The page view in which the rectangle is drawn.
rect	Pointer to the rectangle to draw, specified in device space coordinates.
lineWidth	Border width in pixels. For lineWidth > 1, the border line is entirely <i>inside</i> the rect , thus shrinking the area inside the outline.
dashArray	Pointer to an array of fixed numbers, whose elements alternately specify the length of dashes and gaps. Pass NULL to draw a solid outline.
arrayLen	Number of elements in a dashArray . Ignored if dashArray is NULL . The maximum allowed number of elements is currently 10.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewDrawRect](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewDrawRectOutlineWithHandles

```
void AVPageViewDrawRectOutlineWithHandles
(AVPageView pageView, AVDevRect* rect, ASBool bMidpoints,
ASBool bThin, ASFixed* dashArray, AVTArraySize arrayLen);
```

Description

Draws the specified rectangle in the page view with drag handles.

Parameters

pageView	The page view in which the rect is drawn.
rect	The rectangle that is to be drawn.
bMidpoints	If true , additional handles are drawn at the midpoint of each side of the rectangle.
bThin	If true , the rectangle is drawn using thin lines.
dashArray	Pointer to an array of fixed numbers, whose elements alternately specify the length of dashes and gaps. Pass NULL , to use a solid outline.
arrayLen	The number of elements in dashArray .

Return Value

None

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewEndOperation

```
void AVPageViewEndOperation (AVPageView pageView);
```

Description

Decrements an internal variable. Neither drawing nor [AVPageViewDidChange](#) notifications will occur as long as the variable has a value greater than zero. In addition, frames are not pushed onto the view history stack.

Parameters

pageView	The page view whose SuspendDraw variable is decremented.

Return Value

None

Notifications

[AVPageViewDidChange](#)

Header File

AVCalls.h

Related Methods

[AVPageViewBeginOperation](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVPageViewFilterKeyDownForFocusAnnot

```
ASBool AVPageViewFilterKeyDownForFocusAnnot
(AVPageView pageView, AVKeyCode key, AVFlagBits16 flags,
ASBool* annotWillLoseFocus);
```

Description

Processes a keystroke through the currently focused annotation. The current behavior is defined below:

ASKEY_ESCAPE: The annotation loses focus.

ASKEY_TAB: Focus switches to next annotation in tab order.

ASKEY_ENTER & ASKEY_SPACE: Performs the default action associated with the annotation.

Parameters

pageView	The page view in which the annotation resides.
key	The key code. See <code>ASKey.h</code> for possible values.
flags	A bit-wise OR of the Modifier Flags. See AVSysGetModifiers .
annotWillLoseFocus	<i>(Filled by the method)</i> If true upon return, the keystroke has caused the annotation to lose focus. May be NULL if the caller is not interested in that information.

Return Value

true if the keystroke was processed, **false** otherwise.

Header File

`AVCalls.h`

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVPageViewFocusAnnotPerformOp

```
ASBool AVPageViewFocusAnnotPerformOp (AVPageView pageView,
AVAnnotOp annotOp) ;
```

Description

Attempts to have the currently focused annotation perform the given operation. The method will fail if:

- There is no currently selected annotation.
- **annotOp** is other than **kAVAnnotDefaultAction** or **kAVAnnotShowMenu**.
- The annotation handler is not available or does not support the operation.

Parameters

pageView	The page view containing the annotation.
annotOp	The operation to perform.

Return Value

true if the operation was performed successfully, **false** otherwise.

Notifications

[AVPageViewAnnotDidPerformOp](#)

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVPageViewGetActiveBead

```
PDBead AVPageViewGetActiveBead (AVPageView pageView);
```

Description

Gets the currently active article thread bead in the specified page view.

Parameters

pageView	The page view whose currently active bead is obtained.

Return Value

The current bead of the current thread, or a **NULL** Cos object if there is no current thread.

Header File

AVCalls.h

Related Methods

[AVPageViewGetThreadIndex](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in PIRequir.h) is set to **0x00020000** or higher.

AVPageViewGetAnnotRect

```
void AVPageViewGetAnnotRect (AVPageView pageView,
                             PDAnnot anAnnot, AVDevRect* rect);
```

Description

Gets an annotation's bounding rectangle. This may be a super-set of the actual bounding box of a particular annotation view.

If the page view has more than one page displayed, as in the continuous modes, you should call [AVPageViewSetPageNum](#) with the annotation's page number before calling this method.

NOTE: The coordinate numeric type changed in **PI_ACROVIEW_VERSION** (in **PIRequir.h**) **0x00060000**.

Parameters

pageView	The page view for which the rectangle is transformed.
anAnnot	The annotation whose bounding rectangle is obtained.
rect	<i>(Filled by the method)</i> Pointer to the annotation's bounding rectangle, specified in device space coordinates.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewSetPageNum](#)
[AVPageViewTransformRectRZ](#)
[PDPageGetAnnot](#)

Example

```

PDPage page;
AVRect avRect;
ASInt32 annotNum;
PDAnnot anAnnot;

for (annotNum = 0; annotNum <
    PDPageGetNumAnnots(page); annotNum++) {
    anAnnot = PDPageGetAnnot (page, annotNum);
    if (PDAnnotGetSubtype(anAnnot) ==
        ASAtomFromString("Text")) {
        AVPageViewGetAnnotRect (pageView,
                                anAnnot, &avRect);
        AVPageViewInvalidateRect (pageView,
                                &avRect);
    }
}
PDPageRelease (page);
AVPageViewDrawNow (pageView);

```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVPageViewGetAperture

```
void AVPageViewGetAperture (AVPageView pageView,  
AVDevRect* rect);
```

Description

Gets the aperture of the specified page view. The aperture is the rectangular region of the window in which the document is drawn, measured in device space units.

Parameters

pageView	The page view whose aperture is obtained.
rect	<i>(Filled by the method)</i> Pointer to the aperture rectangle, specified in device space coordinates.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewGetGrayRect](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.



AVPageViewGetAVDoc

```
AVDoc AVPageViewGetAVDoc (AVPageView pageView);
```

Description

Gets the **AVDoc** for the document currently displayed in **pageView**.

Parameters

pageView	The page view whose AVDoc is obtained.

Return Value

The **AVDoc** for **pageView**.

Header File

AVCalls.h

Related Methods

[AVDocGetPageView](#)

[AVPageViewGetPage](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVPageViewGetColor

```
void AVPageViewGetColor (AVPageView pageView,  
    PDColorValue color);
```

Description

Gets the color that will be used for subsequent drawing by [AVPageViewDrawRect](#) and [AVPageViewDrawRectOutline](#).

Parameters

pageView	The page view whose drawing color is obtained.
color	<i>(Filled by the method)</i> The color to get.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

AVCalls.h

Related Methods

[AVPageViewSetColor](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVPageViewGetDevToPageMatrix

```
void AVPageViewGetDevToPageMatrix (AVPageView pageView,  
ASFixedMatrix* devToPageMatrix);
```

Description

Gets the matrix that transforms device space coordinates to user space coordinates for the specified page view.

Parameters

pageView	The page view whose matrix is obtained.
devToPageMatrix	<i>(Filled by the method)</i> Pointer to the transformation matrix.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewGetPageToDevMatrix](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVPageViewGetFocusAnnot

```
ASBool AVPageViewGetFocusAnnot (AVPageView pageView,  
PDAnnot* annot);
```

Description

Gets the currently focused annotation from the page view.

Parameters

pageView	The page view.
annot	<i>(Filled by the method)</i> The currently focused annotation. NULL if the method returns false .

Return Value

true if the annotation was obtained, **false** otherwise.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



AVPageViewGetFirstVisiblePageNum

```
PDFPageNumber AVPageViewGetFirstVisiblePageNum
(AVPageView pageView);
```

Description

Returns the page number of the first page that is visible on the screen.

Parameters

pageView	The page view whose first visible page number is obtained.

Return Value

Page number of the first page that is visible on the screen.

Header File

AVCalls.h

Related Methods

[AVPageViewGetLastVisiblePageNum](#)
[AVPageViewPageNumIsVisible](#)
[AVPageViewGetPageNum](#)
[AVPageViewSetPageNum](#)
[AVPageViewGetSelectedAnnotPageNum](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

AVPageViewGetGrayRect

```
void AVPageViewGetGrayRect (AVPageView pageView,
AVDevRect* greyRect);
```

Description

Gets the rectangle that bounds a page view. This may include some of the “gray area” outside a page’s crop box.

Parameters

pageView	The page view.
greyRect	<i>(Filled by the method)</i> Rectangle bounding the page view, specified in device space coordinates.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewDrawRect](#)
[AVPageViewGetAperture](#)

Example

```
AVRect bounds;

AVPageViewGetGratRect (pageView, &bounds);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.



AVPageViewGetLastVisiblePageNum

```
PDPageNumber AVPageViewGetLastVisiblePageNum
(AVPageView pageView);
```

Description

Returns the page number of the last page that is visible on the screen.

Parameters

pageView	The page view whose last visible page number is obtained.

Return Value

Page number of the last page that is visible on the screen.

Header File

AVCalls.h

Related Methods

[AVPageViewGetFirstVisiblePageNum](#)
[AVPageViewPageNumIsVisible](#)
[AVPageViewGetPageNum](#)
[AVPageViewSetPageNum](#)
[AVPageViewGetSelectedAnnotPageNum](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.



AVPageViewGetLayoutMode

```
PDLayoutMode AVPageViewGetLayoutMode (AVPageView pageView);
```

Description

Gets the page layout mode for a page view.

Parameters

pageView	The page view whose layout mode is obtained.

Return Value

pageView's page layout mode, described in [PDLayoutMode](#).

Header File

AVCalls.h

Related Methods

[AVPageViewSetLayoutMode](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

AVPageViewGetNumVisibleInks

ASInt32 AVPageViewGetNumVisibleInks (**AVPageView** pageView);

Description

Gets the number of inks (color separations) that are displayed for a page view. An ink, in this context, means the content that will appear on a single plate when the page is separated. For example, a **DeviceCMYK** color is four inks, and a separation color space is usually one ink.

Parameters

pageView	The page view whose number of visible inks is obtained.

Return Value

The number of visible inks.

Header File

AVCalls.h

Related Methods

[AVPageViewGetVisibleInks](#)
[AVPageViewSetInkPreview](#)
[AVPageViewSetVisibleInks](#)
[PDPageEnumInks](#)

Product

pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVPageViewGetMousePosition

```
void AVPageViewGetMousePosition (AVPageView pageView,
AVDevCoord* x, AVDevCoord* y);
```

Description

Gets the mouse position in **pageView**. The mouse position is specified in global coordinates (that is, in Mac OS, it is equivalent to calling the Toolbox **GetMouse** call and adding the window's offset on the screen; in UNIX, it is equivalent to calling **XQueryPointer** and adding the window's offset on the screen).

Parameters

pageView	The page view in which the mouse location is obtained.
x	<i>(Filled by the method)</i> The x-coordinate of the mouse location.
y	<i>(Filled by the method)</i> The y-coordinate of the mouse location.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVSysMouseIsStillDown](#)

Example

```
AVPageViewGetMousePosition(pageView,
    &xHit, &yHit);
if (AVPageViewIsAnnotAtPoint(pageView,
    xHit, yHit, &anAnnot)){
    /* Pass the click to the annotation,
       causing the annotation's draw method
       to call */
    ...
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

NOTE: The coordinate numeric types changed in **0x00060000**.

AVPageViewGetMousePositionSnapped

```
void AVPageViewGetMousePositionSnapped (AVPageView pageView,
    AVDevCoord* x, AVDevCoord* y, AVDragType direction);
```

Description

Gets the current mouse position snapped to the layout grid.

Parameters

pageView	The current page view.
x	<i>(Filled by the method)</i> x-coordinate of the mouse position.
y	<i>(Filled by the method)</i> y-coordinate of the mouse position.
direction	An AVDragType indicating how the point is to be translated. Not all AVDragTypes are allowed—only those indicating how the point is to be translated. The following AVDragTypes are used: ● kAVDragRect—snap to nearest grid intersection ● kAVDragSnapToTopLeft—snap to nearest grid intersection in top left direction ● kAVDragSnapToTop—snap to nearest grid line above this point; x is unchanged ● kAVDragSnapToTopRight—snap to nearest grid intersection in top right direction ● kAVDragSnapToRight—snap to nearest grid line right of this point; y is unchanged ● kAVDragSnapToBottomRight—snap to nearest grid intersection in bottom right direction ● kAVDragSnapToBottom—snap to nearest grid line below this point; x is unchanged ● kAVDragSnapToBottomLeft—snap to nearest grid intersection in bottom left direction ● kAVDragSnapToLeft—snap to nearest grid line left of this point; y is unchanged

Return Value

None

Header File

AVCalls.h



Related Methods

[AVPageViewDragOutNewRectSnapped](#)
[AVPageViewDragRectSnapped](#)
[AVPageViewSnapPoint](#)
[AVPageViewSnapRect](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

NOTE: The coordinate numeric types changed in **0x00060000**.

AVPageViewGetNextView

```
ASBool AVPageViewGetNextView (AVPageView pageView,  
PDDoc* pdDoc, PDPageNumber* pageNum, ASFixed* scale);
```

Description

Used by page caching code to determine the next page to cache (and at what scale). Allows clients to do their own page caching.

You can replace this method with your own version, using [HFTReplaceEntry](#).

Parameters

pageView	The page view.
pdDoc	(Filled by the method) The PDDoc containing the next page to cache.
pageNum	(Filled by the method) The next page to cache.
scale	(Filled by the method) The scale of the next page to cache.

Return Value

true if the next page was cached, false otherwise.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

AVPageViewGetPage

```
PDPPage AVPageViewGetPage (AVPageView pageView) ;
```

Description

Gets a **PDPPage** currently displayed in the specified page view . This does *not* acquire the page. Do not use this result across methods that might change the current page. To obtain a value that can be used across such calls, use **PDDocAcquirePage** instead.

Parameters

pageView	The page view whose PDPPage is obtained.

Return Value

PDPPage currently displayed in **pageView**, or **NULL** if there is not a valid **PDPPage** associated with **pageView**.

Header File

AVCalls.h

Related Methods

[PDDocAcquirePage](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVPageViewGetPageNum

```
PDPageNumber AVPageViewGetPageNum (AVPageView pageView);
```

Description

Gets the current page number for **pageView**.

NOTE: If more than one page may be visible, use [AVPageViewGetFirstVisiblePageNum](#), [AVPageViewGetLastVisiblePageNum](#), or [AVPageViewPageNumIsVisible](#) instead of this method.

Parameters

pageView	The page view whose current page number is obtained.

Return Value

Current page number, or **-1** if **pageView** is invalid. The first page in a document is page 0.

Header File

AVCalls.h

Related Methods

[AVPageViewGetFirstVisiblePageNum](#)
[AVPageViewGetLastVisiblePageNum](#)
[AVPageViewPageNumIsVisible](#)
[AVPageViewSetPageNum](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVPageViewGetPageToDevMatrix

```
void AVPageViewGetPageToDevMatrix (AVPageView pageView,  
ASFixedMatrix* pageToDevMatrix);
```

Description

Gets the matrix that transforms user space coordinates to device space coordinates for the specified page view.

Parameters

pageView	The page view whose transformation matrix is obtained.
pageToDevMatrix	<i>(Filled by the method)</i> Pointer to the transformation matrix.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewGetDevToPageMatrix](#)
[AVPageViewGetPageToDevScaling](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVPageViewGetPageToDevScaling

```
void AVPageViewGetPageToDevScaling (AVPageView pageView,  
float *xScale, float *yScale);
```

Description

Gets the point-to-pixel scaling factor that transforms page units to device space units for the specified page view. Although the method returns both x and y factors, these are normally the same.

NOTE: In previous releases, you could use the zoom factor to tranform page units (in points) to device units (in pixels). In Acrobat 6.0 and later, because a point is no longer assumed to be the same as a pixel, this is no longer accurate. It is recommended that you use the page-to-device matrix as much as possible, and use this method if you specifically need the point-to-pixel scaling factor.

Parameters

pageView	The page view whose scaling is obtained.
xScale	<i>(Filled by the method)</i> Pointer to the horizontal scaling factor.
yScale	<i>(Filled by the method)</i> Pointer to the vertical scaling factor.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewGetPageToDevMatrix](#)
[AVPageViewGetZoom](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVPageViewGetPixelInformationAtPoint

```
ASBool AVPageViewGetPixelInformationAtPoint
(AVPageView pageView, AVDevCoord x, AVDevCoord y,
AVInkValue *inkVector, ASUns32 *inkVectorLength);
```

Description

Gets the color-separation ink information for pixels at the cursor position in the specified page view. Ink previews must be turned on before calling this method.

Parameters

pageView	The page view whose pixel information is obtained.
x	The horizontal cursor coordinate of the point, as obtained by the AVPageViewCursorProc callback.
y	The vertical cursor coordinate of the point, as obtained by the AVPageViewCursorProc callback.
inkVector	<i>(Filled by the method)</i> An array of ink values for the specified point.
inkVectorLength	<i>(Filled by the method)</i> The size of the inkVector array.

Return Value

true if the information is successfully retrieved, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVPageViewGetVisibleInks](#)
[AVPageViewSetInkPreview](#)
[PDPageEnumInks](#)

Product

pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



AVPageViewGetSelectedAnnotPageNum

```
PDPageNumber AVPageViewGetSelectedAnnotPageNum
(AVPageView pageView);
```

Description

Returns the page number of the currently selected annotation or **-1** if no annotation is selected.

Parameters

pageView	The page view whose selected annotation page is obtained.

Return Value

Returns the page number of the currently selected annotation or **-1** if no annotation is selected.

Header File

AVCalls.h

Related Methods

[AVPageViewGetFirstVisiblePageNum](#)
[AVPageViewGetLastVisiblePageNum](#)
[AVPageViewPageNumIsVisible](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.



AVPageViewGetThreadIndex

```
AVTArraySize AVPageViewGetThreadIndex (AVPageView pageView);
```

Description

Gets the index of the currently active thread in a page view.

Parameters

pageView	The page view whose active thread index is obtained.

Return Value

The thread index of the current thread, or **-1** if there is no current thread.

Header File

AVCalls.h

Related Methods

[AVPageViewGetActiveBead](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVPageViewGetVisibleAnnotPage

```
PDPageNumber AVPageViewGetVisibleAnnotPage
(AVPageView pageView, PDAnnot annot);
```

Description

Gets the number of a page containing an annotation.

Parameters

pageView	The page view with the annotation.
annot	The annotation whose page number is obtained.

Return Value

The number of the page containing **annot**.

Header File

AVCalls.h

Related Methods

[AVPageViewGetFirstVisiblePageNum](#)
[AVPageViewGetLastVisiblePageNum](#)
[AVPageViewPageNumIsVisible](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

AVPageViewGetVisibleInks

```
void AVPageViewGetVisibleInks (AVPageView pageView,  
    PDPageInkRec *inks);
```

Description

Gets the set of inks to be displayed in a separations preview for a page view.

Parameters

pageView	The page view whose number of visible inks is obtained.
inks	<i>(Filled by the method)</i> An pointer to an array of structures containing the visible ink names for the page. The vector must be large enough to hold the visible inks.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewGetNumVisibleInks](#)
[AVPageViewSetInkPreview](#)
[AVPageViewSetVisibleInks](#)
[PDPageEnumInks](#)

Product

pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVPageViewGetZoom

ASFixed AVPageViewGetZoom (**AVPageView** pageView);

Description

Gets the current zoom for **pageView**. The zoom factor is point-to-point, not point-to-pixel. For example, a page that is 612 points wide at 100% zoom should be 612 points wide on the monitor, not 612 pixels.

NOTE: In previous releases, you could use the zoom factor to transform page units (in points) to device units (in pixels). In Acrobat 6.0 and later, because a point is not longer assumed to be the same as a pixel, this is no longer accurate. It is recommended that you use the page-to-device matrix as much as possible, and use the **AVPageViewGetPageToDevScaling** method if you specifically need the point-to-pixel scaling factor.

Parameters

pageView	The page view whose zoom is obtained.

Return Value

Current zoom, as a fixed number measured in units in which 1.0 is 100% zoom.

Header File

AVCalls.h

Related Methods

[AVPageViewGetPageToDevScaling](#)
[AVPageViewGetZoomType](#)
[AVPageViewZoomTo](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVPageViewGetZoomType

```
AVZoomType AVPageViewGetZoomType (AVPageView pageView);
```

Description

Gets the current zoom type.

Parameters

pageView	The page view whose zoom type is obtained.

Return Value

The current zoom type.

Header File

AVCalls.h

Related Methods

[AVPageViewGetZoom](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVPageViewGhostRectOutline

```
void AVPageViewGhostRectOutline (AVPageView pageView,  
    const AVDevRect* rect);
```

Description

Draws the specified rectangle in **pageView** using the ghost outline.

Parameters

pageView	The page view into which rect is drawn.
rect	The rectangle to draw.

Return Value

None

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVPageViewGoBack

```
void AVPageViewGoBack (AVPageView pageView);
```

Description

Goes to the previous view on the view stack, if a previous view exists. This might result in a different document being made active.

Parameters

pageView	The page view to change.

Return Value

None

Notifications

[AVPageViewDidChange](#)

Header File

AVCalls.h

Related Methods

[AVPageViewGoForward](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVPageViewGoForward

```
void AVPageViewGoForward (AVPageView pageView) ;
```

Description

Goes to the next view on the view stack, if a next view exists. This might result in a different document being made active.

Parameters

pageView	The page view to change.

Return Value

None

Notifications

[AVPageViewDidChange](#)

Header File

AVCalls.h

Related Methods

[AVPageViewGoBack](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVPageViewGoTo

```
void AVPageViewGoTo (AVPageView pageView,
                     PDPageNumber pageNum);
```

Description

Goes to specified page, retaining the current location on the page and the current zoom (either explicit or a variable). Invalidates the display, but does not perform an immediate redraw. This allows your client to call [AVPageViewZoomTo](#), [AVPageViewScrollTo](#), or both and get only a single redraw event. If you decide to do this, you should bracket the calls with [AVPageViewBeginOperation](#) and [AVPageViewEndOperation](#).

Parameters

pageView	The page view in which a different page is displayed.
pageNum	The page number of the destination page. The first page in a document is page 0.

Return Value

None

Notifications

[AVPageViewDidChange](#)

Header File

AVCalls.h

Related Methods

[AVPageViewGetPage](#)
[AVPageViewGoBack](#)
[AVPageViewGoForward](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVPageViewHighlightText

```
void AVPageViewHighlightText (AVPageView pageView,  
    PDTextSelect textSelect);
```

Description

Inverts the given text selection on the current page using the current **AVPageView** color.

Parameters

pageView	The page view.
textSelect	The words to highlight.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewTrackText](#)
[AVPageViewInvalidateText](#)
[AVPageViewPointInText](#)
[PDDocCreateTextSelect](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

AVPageViewInfoToDevice

```
void AVPageViewInfoToDevice (AVPageView pageView,  
    const ASFixedPoint* info, AVlCoord* x, AVlCoord* y);
```

Description

Translates the given point from info space to device space. See [AVPageViewDeviceToInfo](#) for a definition of info space.

Parameters

pageView	The page view holding the focus.
info	The point in info space.
x	(Filled by the method) The x-coordinate of info in device space.
y	(Filled by the method) The y-coordinate of info in device space.

Return Value

None

Header File

AVCalls.h

Related Methods

- [AVPageViewDeviceToInfo](#)
- [AVPageViewInfoToPoint](#)
- [AVPageViewPointToInfo](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewInfoToPoint

```
void AVPageViewInfoToPoint (AVPageView pageView,  
const ASFixedPoint* info ASFixedPoint* pt);
```

Description

Translates the given point from info space to user space. See [AVPageViewDeviceToInfo](#) for a definition of info space.

Parameters

pageView	The page view holding the focus.
info	The point in info space.
pt	<i>(Filled by the method)</i> The point in user space.

Return Value

None

Header File

AVCalls.h

Related Methods

- [AVPageViewDeviceToInfo](#)
- [AVPageViewInfoToDevice](#)
- [AVPageViewPointToInfo](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



AVPageViewInsetRect

```
void AVPageViewInsetRect (AVPageView pageView,
    const AVDevRect* rect, ASBool down);
```

Description

Draws a 3-D looking inset rectangle on the page view. Can be used to implement an “inset” style link where clicking the link causes the page to “push” into the screen.

Parameters

pageView	The page view on which to draw the rectangle.
rect	Rectangle to draw.
down	true to draw an inset rectangle, false to draw the rectangle in its normal state.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewDrawRect](#)
[AVPageViewDrawRectOutline](#)
[AVPageViewInvertRect](#)
[AVPageViewInvertRectOutline](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

AVPageViewInvalidateRect

```
void AVPageViewInvalidateRect (AVPageView pageView,
                               AVDevRect* area);
```

Description

Indicates that the specified area of **pageView** is invalid and should be redrawn. This adds the rectangle to the list of regions to redraw, but does not force an immediate redraw. Use [AVPageViewDrawNow](#) to force an immediate redraw.

Parameters

pageView	The AVPageView in which a region is invalidated.
area	Pointer to the rectangle to invalidate, specified in device space coordinates. Use AVPageViewRectToDevice to convert a user space rectangle to device space. Pass NULL to invalidate the entire currently visible portion of the page.

Return Value

None

Header File

AVCalls.h

Example

```
PDPPage page;
AVRect avRect;
ASInt32 annotNum;
PDAnnot anAnnot;

for (annotNum = 0; annotNum <
     PDPPageGetNumAnnots(page); annotNum++) {
    anAnnot = PDPPageGetAnnot (page, annotNum);
    if (PDAnnotGetSubtype(anAnnot) ==
        ASAtomFromString("Text")) {
        AVPageViewGetAnnotRect(pageView, anAnnot, &avRect);
        AVPageViewInvalidateRect(pageView, &avRect);
    }
}
PDPPageRelease(page);
AVPageViewDrawNow(pageView);
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

NOTE: The coordinate numeric types changed in **0x00060000**.

AVPageViewInvalidateText

```
void AVPageViewInvalidateText (AVPageView pageView,  
    PDTextSelect textSelect);
```

Description

Invalidates the bits that [AVPageViewHighlightText](#) touches.

Parameters

pageView	The page view.
textSelect	The words to invalidate.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewHighlightText](#)
[AVPageViewPointInText](#)
[AVPageViewTrackText](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

AVPageViewInvertQuad

```
void AVPageViewInvertQuad (AVPageView pageView,  
    const Quad* quad, ASBool highlight);
```

Description

Inverts the interior of a quad.

Parameters

pageView	The page view in which the inverted quad is drawn.
quad	Pointer to the quad to invert, specified in device space coordinates. Use AVPageViewPointToDevice to convert the coordinates of the four corners of the quad that is specified in user space.
highlight	If true, uses the highlight mode specified by avpHighlightMode in the Acrobat viewer's preferences file (see AVAppSetPreference). If false, uses a default highlighting mode.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewInvertRect](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

NOTE: The coordinate numeric types changed in **0x00060000**.

AVPageViewInvertRect

```
void AVPageViewInvertRect (AVPageView pageView,  
    const AVDevRect* rect, ASBool highlight);
```

Description

Inverts the interior of a rectangle.

Parameters

pageView	The page view in which the inverted rectangle is drawn.
rect	Pointer to the rectangle to invert, specified in device space coordinates. Use AVPageViewRectToDevice to convert the coordinates of a rectangle that is specified in user space.
highlight	If true , uses the highlight mode specified by avpHighlightMode in the Acrobat viewer's preferences file (see AVAppSetPreference). If false , uses a default highlighting mode.

Return Value

None

Header File

AVCalls.h

Related Methods

- [AVPageViewDrawRect](#)
- [AVPageViewDrawRectOutline](#)
- [AVPageViewInsetRect](#)
- [AVPageViewInvertRectOutline](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

NOTE: The coordinate numeric types changed in **0x00060000**.

AVPageViewInvertRectOutline

```
void AVPageViewInvertRectOutline (AVPageView pageView,  
const AVDevRect* rect);
```

Description

Inverts the specified rectangle's outline.

Parameters

pageView	The page view in which the inverted rectangle outline is drawn.
rect	Pointer to the rectangle whose outline is inverted, specified in device space coordinates. Use AVPageViewRectToDevice to convert the coordinates of a rectangle that is specified in user space coordinates.

Return Value

None

Header File

AVCalls.h

Related Methods

- [AVPageViewDrawRect](#)
- [AVPageViewDrawRectOutline](#)
- [AVPageViewInvertRect](#)



Example

```
while (AVSysMouseIsStillDown())
{
    AVPageViewGetMousePosition(pageView, &x, &y);
    xDelta = x - xOld;
    yDelta = y - yOld;
    if (xDelta != 0 || yDelta != 0) {
        /* Remove old rectangle. */
        AVPageViewInvertRectOutline(pageView, &rr);
        /* Draw new rect reflecting delta moved by the mouse. */
        rr.left += xDelta;
        rr.top += yDelta;
        rr.right += xDelta;
        rr.bottom += yDelta;
        AVPageViewInvertRectOutline(pageView, &rr);
        /* Prepare for next move. */
        xOld = x;
        yOld = y;
    }
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVPageViewIsAnnotAtPoint

```
ASBool AVPageViewIsAnnotAtPoint (AVPageView pageView,
AVDevCoord xHit, AVDevCoord yHit, PDAnnot* hitAnnot);
```

Description

Tests whether the specified point is within an annotation. This method is typically used by mouse-handling code, to pass clicks within an annotation to the appropriate annotation handler. For each annotation, this method calls the appropriate annotation handler's [AVAnnotHandlerPtInAnnotViewBBoxProc](#) to test whether the point is within the annotation.

Parameters

pageView	The page view for which the point to test.
xHit	The x-coordinate of the point to test.
yHit	The y-coordinate of the point to test.
hitAnnot	<i>(Filled by the method)</i> Pointer to the topmost annotation (if any) that was hit by the mouse click.

Return Value

true if the location specified by **xHit** and **yHit** is within an annotation, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVPageViewGetAnnotRect](#)
[AVPageViewGetMousePosition](#)
[AVPageViewGetSelectedAnnotPageNum](#)
[AVPageViewIsAnnotOfTypeAtPoint](#)
[AVPageViewGetSelectedAnnotPageNum](#)

Example

```
if (AVPageViewIsAnnotAtPoint(pageView, xHit, yHit, &anAnnot)){
    /* Pass click on to the annot to call the annot's draw function */
    ...
}
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewsAnnotOfTypeAtPoint

```
ASBool AVPageViewIsAnnotOfTypeAtPoint (AVPageView pageView,
AVDevCoord xHit, AVDevCoord yHit, ASAtom annotType,
ASBool belowOthers, PDAnnot* annot);
```

Description

Determines if an annotation of the specified type resides under the given point. If so, a handle to the annotation is returned.

NOTE: Most tools which author new annotations should ignore annotations of other types when performing hit tests. Use this routine instead of [AVPageViewIsAnnotAtPoint](#) to ignore annotations of other types and pass **true** for **belowOthers** so the user can click through annotations of other types.

Parameters

pageView	The page view.
xHit	x-coordinate of the point.
yHit	y-coordinate of the point.
annotType	The annotation type of interest.
belowOthers	If true , the search continues below the topmost annotation. Otherwise the search halts after a single annotation is found, irrespective of type.
annot	<i>(Filled by the method)</i> The uppermost annotation of the given type.

Return Value

true if an annotation was found, **false** otherwise.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.



AVPageViewIsBeadAtPoint

```
ASBool AVPageViewIsBeadAtPoint (AVPageView pageView,
AVDevCoord xHit, AVDevCoord yHit, PDBead* beadP);
```

Description

Tests whether the specified point is within a bead. Returns the bead if it is.

Parameters

pageView	The page view in which the point is tested.
xHit	The x-coordinate of the point to test, specified in device space coordinates.
yHit	The y-coordinate of the point to test, specified in device space coordinates.
beadP	<i>(Filled by the method)</i> Pointer to the bead in which the point is located. This is only filled if the point is within a bead.

Return Value

true if the point is within a bead, **false** otherwise. If the location is within a bead, the bead is returned in **beadP**.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewIsFocusAnnot

```
ASBool AVPageViewIsFocusAnnot (AVPageView pageView,  
PDAnnot annot);
```

Description

Used to determine if **annot** currently has focus.

Parameters

pageView	The page view in which the annotation resides.
annot	The annotation in question.

Return Value

true if **annot** has the focus, **false** otherwise.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVPageViewPageNumIsVisible

```
ASBool AVPageViewPageNumIsVisible (AVPageView pageView,
PDPageNumber pageNum);
```

Description

Determines if a given page number is visible.

Parameters

pageView	The page view.
pageNum	The page number corresponding to the view of interest.

Return Value

true if **pageNum** is visible, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVPageViewGetFirstVisiblePageNum](#)
[AVPageViewGetLastVisiblePageNum](#)
[AVPageViewGetPageNum](#)
[AVPageViewGetSelectedAnnotPageNum](#)
[AVPageViewSetPageNum](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

AVPageViewPointInText

```
ASBool AVPageViewPointInText (AVPageView pageView,
                               AVDevCoord xHit, AVDevCoord yHit, PDTextSelect pdText);
```

Description

Tests if the given point is in the **PDTextSelect**.

Parameters

pageView	The page view.
xHit	The x-coordinate to test.
yHit	The y-coordinate to test.
pdText	The text to hit-test upon.

Return Value

true if the point is in the **textSelect**, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVPageViewTrackText](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewPointToDevice

```
void AVPageViewPointToDevice (AVPageView pageView,
    const ASFixedPointP p, AVDevCoord* x, AVDevCoord* y);
```

Description

Transforms a point's coordinates from user space to device space.

Parameters

pageView	The page view for which the point's coordinates are transformed.
p	Pointer to the ASFixedPoint whose coordinates, specified in user space, are transformed.
x	<i>(Filled by the method)</i> x-coordinate of the device space point corresponding to p .
y	<i>(Filled by the method)</i> y-coordinate of the device space point corresponding to p .

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewDevicePointToPage](#)
[AVPageViewRectToDevice](#)
[AVPageViewDeviceRectToPage](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewPointToInfo

```
void AVPageViewPointToInfo (AVPageView pageView,  
    const ASFixedPoint* pt, ASFixedPoint* info);
```

Description

Translates the given point from user space to info space. See [AVPageViewDeviceToInfo](#) for a definition of info space.

Parameters

pageView	The page view holding the focus.
pt	The point in user space.
info	<i>(Filled by the method)</i> The point in info space.

Return Value

None

Header File

AVCalls.h

Related Methods

- [AVPageViewDeviceToInfo](#)
- [AVPageViewInfoToDevice](#)
- [AVPageViewInfoToPoint](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVPageViewReadPageDown

```
void AVPageViewReadPageDown (AVPageView pageView);
```

Description

Scrolls down through a document, as if the user hit the Enter key. The scrolling follows articles if Acrobat is currently in article-reading mode.

Parameters

pageView	The page view to scroll.

Return Value

None

Notifications

[AVPageViewDidChange](#)

Header File

AVCalls.h

Related Methods

[AVPageViewReadPageUp](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVPageViewReadPageUp

```
void AVPageViewReadPageUp (AVPageView pageView);
```

Description

Scrolls up through a document, as if the user hit the Enter key. The scrolling follows articles if Acrobat is currently in article-reading mode.

Parameters

pageView	The page view to scroll.

Return Value

None

Notifications

[AVPageViewDidChange](#)

Header File

AVCalls.h

Related Methods

[AVPageViewReadPageDown](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVPageViewRectToDevice

```
void AVPageViewRectToDevice (AVPageView pageView,  
const ASFixedRectP p, AVDevRect* rect);
```

Description

Transforms a rectangle’s coordinates from user space to device space. The resulting **AVRect** will be “normal,” that is, **left < right** and **top < bottom**.

Parameters

pageView	Page view for which the coordinates are transformed.
p	Pointer to the rectangle whose coordinates are transformed, specified in user space coordinates.
rect	<i>(Filled by the method)</i> Pointer to a rectangle containing the device space coordinates corresponding to p .

Return Value

None

Header File

AVCalls.h

Related Methods

- AVPageViewPointToDevice
- AVPageViewDevicePointToPage
- AVPageViewDeviceRectToPage

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.



AVPageViewReleaseMachinePort

```
void AVPageViewReleaseMachinePort (AVPageView pageView,  
void* port);
```

Description

Releases the platform-specific object needed to draw into Acrobat’s document window using a platform’s native graphics calls.

Parameters

pageView	The AVPageView whose platform-dependent port is released.
port	The platform-specific port to release.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewAcquireMachinePort](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVPageViewResumeOffscreenDrawing

```
void AVPageViewResumeOffscreenDrawing (AVPageView pageView);
```

Description

Clients that correctly use machine ports should work with the new off-screen drawing behavior introduced in Acrobat 5. For other clients, the **AVPageViewResumeOffscreenDrawing** and **AVPageViewSuspendOffscreenDrawing** are provided for temporarily disabling the off screen drawing behavior so that it acts more like Acrobat 4.0. The one restriction is that you cannot call these routines while the page is drawn. In other words, do not call these routines from within a drawing callback such as one passed to **AVAppRegisterForPageViewDrawing** or an annotation handler's **DoDraw** or **DoDrawEx** callback.

Parameters

pageView	The AVPageView being drawn.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewSuspendOffscreenDrawing](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVPageViewScrollTo

```
void AVPageViewScrollTo (AVPageView pageView,
AVDevCoord xOrigin, AVDevCoord yOrigin);
```

Description

Scrolls **pageView** to the location specified by **xOrigin** and **yOrigin**, within the limits imposed by the current zoom mode and the Acrobat viewer.

Parameters

pageView	The page view to scroll.
xOrigin	The x-coordinate to scroll to, specified in device space coordinates.
yOrigin	The y-coordinate to scroll to, specified in device space coordinates.

Return Value

None

Notifications

[AVPageViewDidChange](#)

Header File

AVCalls.h

Related Methods

[AVPageViewScrollToRect](#)
[AVPageViewScrollToAnnot](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewScrollToAnnot

```
void AVPageViewScrollToAnnot (AVPageView pageView,  
    PDAnnot annot);
```

Description

Scrolls the **pageView** to ensure that the specified **annot** is visible.

Parameters

pageView	The page view to scroll.
annot	The annotation to scroll to.

Return Value

None

Notifications

[AVPageViewDidChange](#)

Header File

AVCalls.h

Related Methods

[AVPageViewScrollToRect](#)

[AVPageViewScrollTo](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVPageViewScrollToRect

```
void AVPageViewScrollToRect (AVPageView pageView,
    const AVDevRect* rect, ASBool favorLeft, ASBool favorTop,
    AVDevSize margin);
```

Description

Attempts to scroll the page view as little as possible to make the specified rectangle completely visible. This method is handy for auto-scrolling the **AVPageView** in a natural way to bring some page object completely into view. It does not affect the zoom level or **AVZoomType**.

Parameters

pageView	The page view to scroll.
rect	Pointer to the rectangle that is completely visible.
favorLeft	Used when rect is wider than the window's aperture. If favorLeft is true , favors the left side. If false , favors the right side. Favoring a side means that the corresponding edge will appear within the aperture, even if the opposite edge will not.
favorTop	Used when rect is taller than the window's aperture. If favorTop is true , favors the top side. If false , favors the bottom side. Favoring a side means that the corresponding edge will appear within the aperture, even if the opposite edge will not.
margin	Number of pixels that rect should be from the nearest edges if it does not cause the rectangle to go from completely visible to partially obscured.

Notifications

[AVPageViewDidChange](#)

Header File

AVCalls.h

Related Methods

[AVPageViewScrollTo](#)
[AVPageViewScrollToAnnot](#)

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewSetAnnotLocation

```
void AVPageViewSetAnnotLocation (PDAnnot anAnnot,
AVPageView pageView, AVDevCoord x, AVDevCoord y);
```

Description

Sets an annotation's location, specified in device space coordinates.

Parameters

anAnnot	The annotation whose location is set.
pageView	The page view in which the annotation is displayed.
x	The annotation's new x-coordinate, specified in device space coordinates.
y	The annotation's new y-coordinate, specified in device space coordinates.

Return Value

None

Notifications

[PDAnnotWillChange](#)
[PDAnnotDidChange](#)

Header File

AVCalls.h

Related Methods

[AVPageViewGetAnnotRect](#)
[PDPageCreateAnnot](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewSetColor

```
void AVPageViewSetColor (AVPageView pageView,
    PDColorValue color);
```

Description

Sets the color that will be used for subsequent drawing by [AVPageViewDrawRect](#) and [AVPageViewDrawRectOutline](#).

Parameters

pageView	The page view whose drawing color is set.
color	The color to set.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

AVCalls.h

Related Methods

[AVPageViewGetColor](#)

Example

```
PDColorValueRec red;
/* Define red */
red.space = PDDeviceRGB;
red.value[0] = Int32ToFixed(1);
red.value[1] = 0;
red.value[2] = 0;

/* set the drawing color */
AVPageViewSetColor(currentPageView, &red);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVPageViewSetFocusAnnot

```
ASBool AVPageViewSetFocusAnnot (AVPageView pageView,  
PDAnnot focusAnnot, AVAnnotOpData opData);
```

Description

Attempts to set an annotation as the active annotation.

Parameters

pageView	The page view in which the annotation resides.
focusAnnot	The annotation in question.
opData	Can be NULL and probably should be in most cases. If specified it will be used for the kAVAnnotAcceptFocus operation.

Return Value

true if **annot** was given the focus, **false** otherwise.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVPageViewSetInkPreview

```
void AVPageViewSetInkPreview (AVPageView pageView,
                              ASBool inkPreview);
```

Description

Sets or clears the ink preview value for a page view, which allows you to disable the rendering of individual inks. This does not change the set of visible inks. Enabling ink preview also enables overprint preview (OPP).

Parameters

pageView	The page view whose number of visible inks is obtained.
inkPreview	Whether the page view has an ink preview.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewGetNumVisibleInks](#)
[AVPageViewGetPixelInformationAtPoint](#)
[AVPageViewGetVisibleInks](#)
[AVPageViewSetVisibleInks](#)

Product

pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVPageViewSetLayoutMode

```
void AVPageViewSetLayoutMode (AVPageView pageView,  
    PDLayoutMode mode) ;
```

Description

Sets the layout mode for a page view.

Parameters

pageView	The page view whose layout mode is set.
mode	The new layout mode for the page view.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewGetLayoutMode](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

AVPageViewSetPageNum

`PDPageNumber` AVPageViewSetPageNum (`AVPageView` pageView,
`PDPageNumber` pageNum) ;

Description

Sets the current logical page of the page view, which might not be the same as the current number indicated on the screen.

Client writers in general do not need this, but if you wish to perform some operation on a page other than the one returned by `AVPageViewGetPageNum`, you can use this call to temporarily set the page. You must restore the page number when you are done. You should avoid causing any major changes to the page view (such as scrolling) to ensure that you end up restoring the page to the correct value.

Parameters

<code>pageView</code>	The page view.
<code>pageNum</code>	The page number.

Return Value

Previous page number.

Header File

AVCalls.h

Related Methods

`AVPageViewGetFirstVisiblePageNum`
`AVPageViewGetLastVisiblePageNum`
`AVPageViewGetPageNum`
`AVPageViewGetSelectedAnnotPageNum`
`AVPageViewPageNumIsVisible`

Product

reader, base, pro

Availability

Available if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to `0x00020002` or higher.

AVPageViewSetVisibleInks

```
void AVPageViewSetVisibleInks (AVPageView pageView,
                               ASInt32 nInks, PDPageInkRec *inks);
```

Description

Sets the set of inks to be displayed in a separations preview for a page view. The ink preview must be turned on for this method to have an effect.

Parameters

pageView	The page view whose number of visible inks is obtained.
ninks	The size of the inks array. The maximum number of inks is 16.
inks	An array of the new visible inks for the page. Copies the inks into local storage, so the caller does not need to keep the array.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewGetNumVisibleInks](#)
[AVPageViewGetVisibleInks](#)
[AVPageViewSetInkPreview](#)

Product

pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



AVPageViewShowControl

```
void AVPageViewShowControl (AVPageView pageView,  
AVPageViewControlID controlID, ASBool show);
```

Description

Shows or hides the controls in the status area at the bottom of a page view.

Parameters

pageView	The page view whose controls are affected.
controlID	The controls affected.
show	true if the controls are displayed, false if not shown.

Return Value

None

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

AVPageViewSnapPoint

```
void AVPageViewSnapPoint (AVPageView pageView, AVDevCoord* x,
AVDevCoord* y, AVDragType direction);
```

Description

Snaps a point to the layout grid if the **avpSnapToGrid** preference is set.

NOTE: This method is superseded by **AVPageViewSnapPointEx** in Acrobat 6.0.

Parameters

pageView	The page view.
x	(Filled by the method) The x-coordinate of the point.
y	(Filled by the method) The y-coordinate of the point.
direction	An AVDragType indicating how the point is to be translated. Not all AVDragTypes are allowed—only the following AVDragTypes are used: ● kAVDragRect—snap to nearest grid intersection ● kAVDragSnapToTopLeft—snap to nearest grid intersection in top left direction ● kAVDragSnapToTop—snap to nearest grid line above this point; x is unchanged ● kAVDragSnapToTopRight—snap to nearest grid intersection in top right direction ● kAVDragSnapToRight—snap to nearest grid line right of this point; y is unchanged ● kAVDragSnapToBottomRight—snap to nearest grid intersection in bottom right direction ● kAVDragSnapToBottom—snap to nearest grid line below this point; x is unchanged ● kAVDragSnapToBottomLeft—snap to nearest grid intersection in bottom left direction ● kAVDragSnapToLeft—snap to nearest grid line left of this point; y is unchanged

Return Value

None

Header File

AVCalls.h

**Product**

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewSnapPointEx

```
void AVPageViewSnapPointEx (AVPageView pageView,
    ASFixedPoint* pt, AVDragType direction);
```

Description

Snaps a point to the layout grid if the **avpSnapToGrid** preference is set, using page-space coordinates.

NOTE: This method supersedes **AVPageViewSnapPoint** in Acrobat 6.0.

Parameters

pageView	The page view.
pt	<i>(Filled by the method)</i> The point that will be snapped to the grip, specified in page space.
direction	An AVDragType indicating how the point is to be translated. Not all AVDragTypes are allowed—only the following AVDragTypes are used: ● kAVDragRect—snap to nearest grid intersection ● kAVDragSnapToTopLeft—snap to nearest grid intersection in top left direction ● kAVDragSnapToTop—snap to nearest grid line above this point; pt.h is unchanged ● kAVDragSnapToTopRight—snap to nearest grid intersection in top right direction ● kAVDragSnapToRight—snap to nearest grid line right of this point; pt.v is unchanged ● kAVDragSnapToBottomRight—snap to nearest grid intersection in bottom right direction ● kAVDragSnapToBottom—snap to nearest grid line below this point; pt.h is unchanged ● kAVDragSnapToBottomLeft—snap to nearest grid intersection in bottom left direction ● kAVDragSnapToLeft—snap to nearest grid line left of this point; pt.v is unchanged

Return Value

None

Header File

AVCalls.h

**Related Methods**

[AVPageViewDragRectSnappedEx](#)
[AVPageViewSnapPoint](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVPageViewSnapRect

```
void AVPageViewSnapRect (AVPageView pageView,
    AVDevCoord xStart, AVDevCoord yStart, AVDevCoord xNow,
    AVDevCoord yNow, AVDevRect* startRect, AVDevRect* resultRect,
    ASInt32 handleType, ASUns32 modifiers, AVDevRect* extrema);
```

Description

Given a starting point, ending point, and starting rectangle, return a resulting rectangle which is snapped to the grid. The routine is designed for use within a custom mouse-drag loop, when the default drawing behavior provided by [AVPageViewDragRectSnapped](#) is insufficient.

Parameters

pageView	The page view.
xStart	The x-coordinate of the point where the user initially clicked, specified in device space coordinates.
yStart	The y-coordinate of the point where the user initially clicked, specified in device space coordinates.
xNow	The x-coordinate.
yNow	The y-coordinate.
startRect	The initial position of rectangle.
resultRect	<i>(Filled by the method)</i> The position of the rectangle at the end of the operation.
handleType	One of the AVRectHandleType enumerated values, typically obtained by calling AVRectHandleHitTest with the initial rectangle and starting coordinates. The handleType determines which point or edge of the rectangle should be snapped to the grid.

modifiers	Pass AVSysGetModifiers to pass in one of the Modifier Keys. If the key indicates that the Shift key is pressed, one of the following is done: ● If the handletype parameter is a resizing parameter (kAVRectHandleTopLeft through kAVRectHandleLeftMiddle) then we maintain the aspect ratio of the original rect. ● If the handletype parameter is kAVRectHandleNone, this means that the rect is being moved around rather than resized, and we will snap the result rect to the x- or y-axis of the start rect, whichever is closest.
extrema	Restricts the drag operation to a bounding rectangle. If NULL, the bounding rectangle of the page view is used.

Return Value

None

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.



AVPageViewStartReadingThread

```
void AVPageViewStartReadingThread (AVPageView pageView,  
PDThread thread);
```

Description

Puts the specified page view into “article thread-reading” mode.

Parameters

pageView	The page view to set to article thread-reading mode.
thread	The thread to read.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewGetActiveBead](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVPageViewSuspendOffscreenDrawing

```
void AVPageViewSuspendOffscreenDrawing (AVPageView pageView);
```

Description

Clients that correctly use machine ports should work with the new off-screen drawing behavior introduced in Acrobat 5.0. For other clients, the [AVPageViewResumeOffscreenDrawing](#) and [AVPageViewSuspendOffscreenDrawing](#) are provided for temporarily disabling the off screen drawing behavior so that it acts more like Acrobat 4.0. The one restriction is that you cannot call these routines while the page being is drawn. In other words, do not call these routines from within a drawing callback such as one passed to [AVAppRegisterForPageViewDrawing](#) or an annotation handler's **DoDraw** or **DoDrawEx** callback. Off-screen drawing should be suspended as rarely and briefly as possible (for example, only while a client that does not use machine ports correctly has a selection active).

Parameters

pageView	The page view whose drawing is to be suspended.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewResumeOffscreenDrawing](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVPageViewToDestInfo

```
AVDestInfo AVPageViewToDestInfo (AVPageView pageView,
ASAtom fitType);
```

Description

Creates a destination info object from a given **AVPageView** and **fitType**.

Parameters

pageView	The page view from whose current view the destination info is created.
fitType	The ASAtom specifying the fit type that the view destination will have. The string associated with fitType must be one of View Destination Fit Types .

Return Value

The newly created destination info.

Header File

AVCalls.h

Related Methods

[AVDestInfoDestroy](#)
[AVPageViewUseDestInfo](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020003** or higher.

AVPageViewToViewDest

```
PDViewDestination AVPageViewToViewDest (AVPageView pageView,  
ASAtom fitType, PDDoc srcPDDoc);
```

Description

Builds a PDViewDestination from the current zoom and position.

Parameters

pageView	The page view from whose current view the destination is created.
fitType	The ASAtom specifying the fit type that the view destination will have. The string associated with fitType must be one of View Destination Fit Types.
srcPDDoc	Document in which the view destination is used.

Return Value

The newly created view destination.

Header File

AVCalls.h

Related Methods

- PDViewDestCreate
- PDActionGetDest
- PDActionNewFromDest

Product

reader, base, pro

Availability

Available if PI_ACROVIEW_VERSION (in PIRequir.h) is set to 0x00020000 or higher.

AVPageViewTrackText

```
PDTextSelect AVPageViewTrackText (AVPageView pageView,
AVDevCoord xHit, AVDevCoord yHit, PDTextSelect current);
```

Description

Called in response to a mouse click to track a text selection on the screen. Uses the **AVPageView** current color, and leaves the screen with any highlights visible. Does not affect the current document selection.

Parameters

pageView	The page view.
xHit	The x-coordinate of the original click.
yHit	The y-coordinate of the original click.
current	Any existing selection that should be built upon.

Return Value

PDTextSelect containing the words selected.

Header File

AVCalls.h

Related Methods

[AVPageViewHighlightText](#)
[AVPageViewInvalidateText](#)
[AVPageViewPointInText](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVPageViewTransformRectRZ

```
void AVPageViewTransformRectRZ (AVPageView pv, ASInt32 flags,
ASFixedRect* ar, ASFixedMatrix* mr);
```

Description

Calculates where an annotation is drawn with no zoom or no rotation, as specified by the annotation flags. This method generalizes [AVPageViewGetAnnotRect](#).

Parameters

pv	The page view used for the transformation.
flags	Annotation flags obtained by PDAnnotGetFlags .
ar	Pointer to the annotation's bounding rectangle, specified in user space coordinates.
mr	<i>(Filled by the method)</i> The transformation matrix.

Return Value

None

Header File

AVCalls.h

Related Methods

- [AVPageViewAppearanceGetAVMatrix](#)
- [AVPageViewDeviceRectToPageRZ](#)
- [AVPageViewGetAnnotRect](#)

Example

```
ASFixedMatrix mr;
ASFixedRect ar, fr;
ASUns32 flags = PDAnnotGetFlags(annot);
PDAnnotGetRect(annot, &ar);
fr = ar;
AVPageViewTransformRectRZ(pageView, flags, &fr, &mr);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION**(in `PIRequir.h`) is set to **0x00040000** or higher.

AVPageViewUpdateInfoPanel

```
void AVPageViewUpdateInfoPanel (AVPageView pageView,
AVInfoPanelUpdateType updateType, void* data);
```

Description

Allow clients to control the Info panel output.

Parameters

pageView	The page view.
updateType	AVInfoPanelUpdateType constant. If kAVInfoPanelLock or kAVInfoPanelUnlock , data should be NULL and is ignored. If updateType is kAVInfoPanelRect , clients should pass an AVRect32P through data . It's also useful to note how the rectangle will be interpreted by the info panel. The info panel will display the following data: X: The left of the rectangle. Y: The top of the rectangle. W: The width of the rectangle. H: The height of the rectangle.
data	User-supplied data.

Return Value

None

Header File

AVCalls.h

Example

```
AVPageViewUpdateInfoPanel(pageView, kAVInfoPanelLock, NULL);
while (AVSysMouseIsStillDown()) {
    AVRect32 rect;
    ... // Set the rect members to something meaningful, then:
    AVPageViewUpdateInfoPanel(pageView, kAVInfoPanelRect, (void*)&rect);
    ... // Do other processing
}
AVPageViewUpdateInfoPanel(pageView, kAVInfoPanelUnlock, NULL);
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



AVPageViewUseDestInfo

```
void AVPageViewUseDestInfo (AVPageView pageView,  
AVDestInfo destInfo);
```

Description

Causes the given page view to change to the view given by an [AVDestInfo](#) object.

Parameters

pageView	The page view to change.
destInfo	The destination to use.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewToDestInfo](#)

[AVPageViewUseThisDestination](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020003** or higher.

AVPageViewUseThisDestination

```
void AVPageViewUseThisDestination (AVPageView pageView,  
PDViewDestination viewDest, ASFixed sourceZoom);
```

Description

Causes the given page view to change to the view given by **viewDest** and **sourceZoom**.

Parameters

pageView	The page view to change.
viewDest	The view destination.
sourceZoom	The zoom factor to use, unless viewDest specifies inheriting the current zoom, which is the zoom in effect when a link is clicked, for example. sourceZoom is used only if a) the view destination is of type XYZ (that is, specifies a point and a zoom) and b) the zoom is zero (meaning “inherit the current zoom”). In this case, sourceZoom is used as the zoom to inherit.

Return Value

None

Header File

AVCalls.h

Related Methods

[PDViewDestCreate](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

AVPageViewZoomTo

```
void AVPageViewZoomTo (AVPageView pageView,
                       AVZoomType zoomType, ASFixed scale);
```

Description

Sets the zoom factor and zoom type for the specified page view.

Parameters

pageView	The page view to zoom.
zoomType	The zoom type to set.
scale	Zoom factor, specified as a magnification factor (for example, 1.0 displays the document at actual size). scale is ignored unless zoomType is AVZoomNoVary . Use zero to inherit the zoom.

Return Value

None

Notifications

[AVPageViewDidChange](#)

Header File

AVCalls.h

Related Methods

[AVPageViewScrollTo](#)
[AVPageViewScrollToRect](#)

Example

```
AVPageViewZoomTo (pageView, AVZoomFitPage, fixedZero);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVSweetPea

AVSweetPeaGetBasicSuiteP

```
SPBasicSuite* AVSweetPeaGetBasicSuiteP (void);
```

Description

Used to implement the Adobe Dialog Manager (ADM). Accesses basic suite. For complete details, see [Adobe Dialog Manager Reference](#).

Parameters

None

Return Value

Pointer to the structure that contains the “SweetPea” functions.

Header File

AVSPCalls.h

Related Methods

[AVSweetPeaGetPluginRef](#)
[AVSweetPeaGetResourceAccess](#)
[AVSweetPeaIsADMAvailable](#)
[AVSweetPeaProcessADMEvent](#)
[AVSweetPeaSetResourceAccess](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVSweetPeaGetPluginRef

```
SPPluginRef AVSweetPeaGetPluginRef (void);
```

Description

Used to implement the Adobe Dialog Manager (ADM). This method is used to obtain a reference to the ADM client itself (not the client you are currently developing). For complete details, see [Adobe Dialog Manager Reference](#).

Parameters

None

Return Value

A reference to the ADM client itself.

Header File

AVSPCalls.h

Related Methods

[AVSweetPeaGetBasicSuiteP](#)
[AVSweetPeaGetResourceAccess](#)
[AVSweetPeaIsADMAvailable](#)
[AVSweetPeaProcessADMEvent](#)
[AVSweetPeaSetResourceAccess](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVSweetPeaGetResourceAccess

```
SPErr AVSweetPeaGetResourceAccess (SPPluginRef pluginRef,
    SPPlatformAccessRef* resourceAccess);
```

Description

Used to implement the Adobe Dialog Manager (ADM). Call this function to store away the current resource access, which you will restore after your dialog is closed. For complete details, see [Adobe Dialog Manager Reference](#).

Parameters

pluginRef	The value returned from the AVSweetPeaGetPluginRef method.
resourceAccess	Platform-specific resource information.

Return Value

ADM error codes.

Header File

AVSPCalls.h

Related Methods

- [AVSweetPeaGetBasicSuiteP](#)
- [AVSweetPeaGetPluginRef](#)
- [AVSweetPeaIsADMAvailable](#)
- [AVSweetPeaProcessADMEvent](#)
- [AVSweetPeaSetResourceAccess](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVSweetPealsADMAvailable

ASBool AVSweetPeaIsADMAvailable (void);

Description

Used to implement the Adobe Dialog Manager (ADM). This method is used to determine whether ADM is available. For complete details, see [Adobe Dialog Manager Reference](#).

Parameters

None

Return Value

true if ADM is available. **false** if not.

Header File

AVSPCalls.h

Related Methods

[AVSweetPeaGetBasicSuiteP](#)
[AVSweetPeaGetPluginRef](#)
[AVSweetPeaGetResourceAccess](#)
[AVSweetPeaProcessADMEvent](#)
[AVSweetPeaSetResourceAccess](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVSweetPeaProcessADMEvent

```
void AVSweetPeaProcessADMEvent (ASWindowRef window);
```

Description

Used to implement the Adobe Dialog Manager (ADM). Call this function to have a given **ASWindowRef** process events. This is useful for modeless dialogs. For complete details, see [Adobe Dialog Manager Reference](#).

Parameters

window	A platform-dependent window handle.

Return Value

None

Header File

AVSPCalls.h

Related Methods

[AVSweetPeaGetBasicSuiteP](#)
[AVSweetPeaGetPluginRef](#)
[AVSweetPeaGetResourceAccess](#)
[AVSweetPeaIsADMAvailable](#)
[AVSweetPeaSetResourceAccess](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



AVSweetPeaSetResourceAccess

```
SPErr AVSweetPeaSetResourceAccess (SPPluginRef pluginRef,
SPPlatformAccessRef resourceAccess);
```

Description

Used to implement the Adobe Dialog Manager (ADM). This function tells ADM which resource file contains your dialog resources. For complete details, see [Adobe Dialog Manager Reference](#).

Parameters

pluginRef	The value returned from the AVSweetPeaGetPluginRef method.
resourceAccess	Platform-specific resource information.

Return Value

Error code.

Header File

AVSPCalls.h

Related Methods

[AVSweetPeaGetBasicSuiteP](#)
[AVSweetPeaGetPluginRef](#)
[AVSweetPeaGetResourceAccess](#)
[AVSweetPeaIsADMAvailable](#)
[AVSweetPeaProcessADMEvent](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



AVSys

AVSysAllocTimeStringFromTimeRec

```
char* AVSysAllocTimeStringFromTimeRec (ASTimeRecP timeRec);
```

Description

Given an [ASTimeRecP](#), gets a string representing the date and time. This routine is locale-friendly.

The returned string is allocated using [ASmallloc](#) and must be freed by the caller with [ASfree](#).

Parameters

timeRec	A pointer to an ASTimeRec structure.

Return Value

String representing the date and time. Returns **NULL** if conversion fails.

Header File

[AVCalls.h](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in [PIRequir.h](#)) is set to **0x00040000** or higher.



AVSysBeep

```
void AVSysBeep (ASInt32 duration);
```

Description

Beeps.

Parameters

duration	Always pass 0.

Return Value

None

Header File

AVCalls.h

Example

```
AVSysBeep(0);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVSysGetCursor

[AVCursor](#) AVSysGetCursor (void);

Description

Gets the current cursor. Use this method when you want to change the cursor temporarily and be able to restore it to its current shape.

Parameters

None

Return Value

The current cursor.

Header File

AVCalls.h

Related Methods

[AVSysGetStandardCursor](#)

[AVSysSetCursor](#)

Example

```
oldCursor = AVSysGetCursor();
AVSysSetCursor (AVSysGetStandardCursor (
    WAIT_CURSOR));
/* do some work */
...
/* Return to old cursor */
AVSysSetCursor (oldCursor);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVSysGetIconFromFilename

```
AVIcon AVSysGetIconFromFilename (AStext fname,
AVSysIconType itype);
```

Description

Gets an icon of the specified type from the specified file.

Parameters

fname	The filename.
itype	The icon type, kAVSysSmallIcon or kAVSysLargeIcon .

Return Value

The icon object, or **NULL** if no icon of the given type was found.

Header File

AVCalls.h

Related Methods

[AVSysGetIconFromMimeType](#)
[AVSysGetIconFromTypeAndCreator](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVSysGetIconFromMimeType

```
AVIcon AVSysGetIconFromMimeType (const char *mimeType,
AVSysIconType itype);
```

Description

Gets an icon of the specified type from the specified Mime type.

Parameters

mimeType	The Mime type.
itype	The icon type, kAVSysSmallIcon or kAVSysLargeIcon .

Return Value

The icon object, or **NULL** if no icon of the given type was found.

Header File

AVCalls.h

Related Methods

[AVSysGetIconFromFilename](#)
[AVSysGetIconFromTypeAndCreator](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVSysGetIconFromTypeAndCreator

```
AVIcon AVSysGetIconFromTypeAndCreator (ASUns32 type,
ASUns32 creator, AVSysIconType itype);
```

Description

Gets an icon of the specified type from the specified type and creator (for Mac OS).

Parameters

type	The type value (which specifies a type such as PDF or GIF).
creator	The creator value (which specifies a creator such as Acrobat or Photoshop).
itype	The icon type, kAVSysSmallIcon or kAVSysLargeIcon .

Return Value

The icon object, or **NULL** if no icon of the given type was found.

Header File

AVCalls.h

Related Methods

[AVSysGetIconFromFilename](#)
[AVSysGetIconFromMimeType](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVSysGetModifiers

[AVFlagBits32](#) AVSysGetModifiers (void);

Description

Gets a flag indicating which modifier keys are currently being pressed.

Parameters

None

Return Value

An OR of the [Modifier Keys](#).

Header File

AVCalls.h

Related Methods

[AVSysMouseIsStillDown](#)

Example

```
ASBool shiftKeyIsDown =  
    ((AVSysGetModifiers() & AV_SHIFT) != 0);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVSysGetUsePenForInput

```
ASBool AVSysGetUsePenForInput (void);
```

Description

Gets the value of the **UsePenForInput** user-preference attribute, currently unused.

Parameters

None

Return Value

Currently, always **true**.

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVSysGetStandardCursor

[AVCursor](#) AVSysGetStandardCursor ([ASInt32](#) cursorID) ;

Description

Gets the specified cursor. The cursor can subsequently be displayed using [AVSysSetCursor](#).

Parameters

cursorID	The cursor to show. Must be one of the Predefined Cursors .

Return Value

The specified cursor.

Header File

AVCalls.h

Related Methods

[AVSysGetCursor](#)
[AVSysSetCursor](#)

Example

```
oldCursor = AVSysGetCursor();
AVSysSetCursor(AVSysGetStandardCursor(
    WAIT_CURSOR));
/* Do some work */
...
/* Return to old cursor */
AVSysSetCursor(oldCursor);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVSysMouseIsStillDown

ASBool AVSysMouseIsStillDown (void);

Description

Tests whether the mouse button is still being pressed.

Parameters

None

Return Value

true if the user is holding the mouse button down and has not released it since the last mouse-down event, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVSysGetModifiers](#)

Example

```
while (AVSysMouseIsStillDown())
{
    AVPageViewGetMousePosition(pageView, &x, &y);
    xDelta = x - xOld;
    yDelta = y - yOld;
    if (xDelta != 0 || yDelta != 0) {
        /* Remove old rectangle. */
        AVPageViewInvertRectOutline(pageView, &rr);
        /* Draw new rect reflecting delta moved by the mouse */
        rr.left += xDelta;
        rr.top += yDelta;
        rr.right += xDelta;
        rr.bottom += yDelta;
        AVPageViewInvertRectOutline(pageView, &rr);
        /* Prepare for next move. */
        xOld = x;
        yOld = y;
    }
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVSysSetCursor

```
void AVSysSetCursor (AVCursor cursor);
```

Description

Sets the cursor to the specified **AVCursor**.

Parameters

cursor	The cursor to display. Predefined platform-independent cursors can be obtained using AVSysGetStandardCursor. Clients can use their own custom cursors as follows: Macintosh users: Use the Macintosh Toolbox GetCursor call to retrieve your cursor from a resource. Cast the resulting CursHandle to an AVCursor and pass it to AVSysSetCursor. Windows users: Use the Windows API function LoadCursor to retrieve your cursor resource. Cast the resulting HCURSOR to an AVCursor and pass it to AVSysSetCursor.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVSysGetCursor](#)
[AVSysGetStandardCursor](#)
[AVSysSetWaitCursor](#)

Example

```
oldCursor = AVSysGetCursor();
AVSysSetCursor(AVSysGetStandardCursor(
    WAIT_CURSOR));
/* Do some work */
...
/* Return to old cursor */
AVSysSetCursor(oldCursor);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVSysSetWaitCursor

```
void AVSysSetWaitCursor (ASBool turnOn, ASUns32 timeLimit);
```

Description

Turns the wait cursor on or off. Unlike [AVSysGetCursor](#), [AVSysSetWaitCursor](#) ensures that the wait cursor stays on until another call to [AVSysSetWaitCursor](#) is made to turn it off, or the time limit is reached. When using this call to turn on the wait cursor, you must ensure that another call will be made to turn it back off.

Parameters

turnOn	true sets the wait cursor; false resets it.
timeLimit	Time limit in ticks (1/60th of a second). If the time limit is reached, the wait cursor will be turned off as if you called AVSysSetWaitCursor(false, ...) . This value is ignored if it is less than or equal to 0 or if the turnOn parameter is false. Set this to a reasonable nonzero value to ensure that the user wait cursor will go away.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVSysGetCursor](#)
[AVSysGetStandardCursor](#)
[AVSysSetCursor](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in [PIRequir.h](#)) is set to **0x00050000** or higher.



AVTool

AVToolGetType

```
ASAtom AVToolGetType (AVTool tool);
```

Description

Gets the currently active tool's type. See [Toolbar and Toolbar Button Names](#) for a list of the built-in tool types.

Parameters

tool	The tool whose type is obtained.

Return Value

The **ASAtom** returned can be converted to a string using [ASAtomGetString](#).

Header File

AVCalls.h

Related Methods

[AVAppEnumTools](#)
[AVAppGetActiveTool](#)
[AVAppGetDefaultTool](#)
[AVAppGetLastActiveTool](#)
[AVAppGetToolByName](#)
[AVToolIsPersistent](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVToolsPersistent

```
ASBool AVToolsPersistent (AVTool tool);
```

Description

Tests whether the specified tool is in a state keeping it active through multiple operations rather than only once, then restoring the previous tool. This method is called by another one-shot tool's **Activate** procedure. Two one-shot tools cannot cycle, because if the previous tool was not persistent, the second non-persistent tool reverts to the default tool.

Parameters

tool	The tool whose persistence flag is tested.

Return Value

true if the tool is persistent, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVAppEnumTools](#)
[AVAppGetActiveTool](#)
[AVAppGetDefaultTool](#)
[AVAppGetLastActiveTool](#)
[AVAppGetToolByName](#)
[AVToolGetType](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVToolBar

AVToolBarAddButton

```
void AVToolBarAddButton (AVToolBar toolBar,
    AVToolButton button, ASBool before, AVToolButton otherButton);
```

Description

Inserts a button into a toolbar. Call [AVToolBarUpdateButtonStates](#) after adding a button to update the toolbar.

Parameters

toolBar	The toolbar into which a button is added.
button	The button to add to the toolbar.
before	If true , button is added before otherButton ; if false, it is added after. If otherButton is NULL and before is true , the button is added to the beginning of the toolbar. If otherButton is NULL and before is false , the button is added to the end of the toolbar.
otherButton	A button relative to which the new button is added.

Return Value

None

Known Exceptions

[genErrNoMemory](#)

Header File

AVCalls.h

Related Methods

[AVToolButtonRemove](#)



Example

```
AVToolButton myButton;
AVToolBar mybar;

myButton = AVToolButtonNew(
    ASAtomFromString("Hello World"),
    GetResource('SICN', 600), FALSE, FALSE);
mybar = AVAppGetToolBar();
{
    AVToolButton oldButton =
        AVToolBarGetButtonByName(mybar,
            ASAtomFromString("EndDialogGroup"));
    AVToolBarAddButton(mybar, myButton,
        FALSE, oldButton);
}
AVToolBarGetNumButtons:
AVToolBar bar = AVAppGetToolBar();
if (AVToolBarGetNumButtons(bar)) {
    ...
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVToolBarEnumButtons

```
void AVToolBarEnumButtons (AVToolBar toolBar,
AVToolButtonEnumProc enumProc, void* clientData);
```

Description

Calls **enumProc** once for each toolbar button in the specified toolbar.

If a tool button has a flyout, this is a *separate* toolbar from the toolbar returned by [AVAppGetToolBar](#); enumerating the toolbar buttons on this main toolbar does *not* enumerate the toolbar buttons on any flyout. To enumerate the toolbar buttons on a button's flyout, call [AVToolButtonGetFlyout](#) to get its associated toolbar, then call **AVToolBarEnumButtons** with this toolbar.

NOTE: **AVToolBarEnumButtons** does not enumerate toolbar buttons that are marked as external by [AVToolButtonSetExternal](#).

Parameters

toolBar	The toolbar whose buttons are enumerated.
enumProc	User-supplied procedure to call once for each button. Enumeration ends if enumProc returns false .
clientData	Pointer to user-supplied data to pass to enumProc each time it is called.

Return Value

None

Header File

AVCalls.h

Related Methods

- [AVAppGetToolBar](#)
- [AVToolBarGetButtonByName](#)
- [AVToolButtonGetFlyout](#)
- [AVToolButtonSetExternal](#)



Example

```
/* Find the "Hand" toolbar button */
ACCB1 ASBool ACCB2 myButtonEnum
(AVToolButton button, void* clientData){
    if(AVToolButtonGetName(button) ==
        ASAtomFromString("Hand"))
        return false;
    return true;
}
AVToolBarEnumButtons(bar,
    ASCallbackCreateProto(
        AVToolButtonEnumProc, &myButtonEnum),
    NULL);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to 0x00020000 or higher.

AVToolBarGetButtonByName

```
AVToolButton AVToolBarGetButtonByName (AVToolBar toolBar,
ASAtom buttonName) ;
```

Description

Gets the toolbar button that has the specified name.

Parameters

toolBar	The toolbar in which the button is located.
buttonName	The ASAtom for the button to get. The character string representing buttonName can be converted to an ASAtom using ASAtomFromString . See Toolbar and Toolbar Button Names for a list of the names of the built-in buttons.

Return Value

The button with the specified name; if the name is not found, the return value is **NULL**.

Header File

AVCalls.h

Related Methods

[AVToolBarEnumButtons](#)

Example

```
/* Add a button just before the
   endToolsGroup separator */
AVToolBarAddButton (AVAppGetToolBar(),
myToolButton, true,
AVToolBarGetButtonByName (ToolBar,
ASAtomFromString ("endToolsGroup")));
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVToolBarGetFrame

```
void AVToolBarGetFrame (AVToolBar toolBar, AVRect* frame);
```

Description

Gets the coordinates of **toolBar**'s frame.

UNIX users: The toolbar frame is not needed or used, and this method returns a meaningless frame.

Parameters

toolBar	The toolbar whose frame is obtained.
frame	<i>(Filled by the method)</i> Pointer to a rectangle specifying the toolbar's frame, specified in device space coordinates.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppGetToolBar](#)
[AVToolButtonGetFlyout](#)

Example

```
AVRect frame;

AVToolBarGetFrame (AVAppGetToolBar(),
    frame);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

NOTE: This method was deprecated in version 5.0.

AVToolBarGetNumButtons

```
AVTArraySize AVToolBarGetNumButtons (AVToolBar toolbar);
```

Description

Gets the number of buttons in `toolbar`.

Parameters

<code>toolbar</code>	The toolbar whose button count is obtained.

Return Value

The number of buttons in `toolbar`.

Header File

AVCalls.h

Related Methods

[AVAppGetToolBar](#)
[AVToolButtonGetFlyout](#)

Product

reader, base, pro

Availability

Available if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

AVToolBarIsRoomFor

```
ASBool AVToolBarIsRoomFor (AVToolBar toolBar,  
ASInt16 nButtons, ASInt16 nSeparators);
```

Description

Tests whether there is room in a toolbar for an additional specified number of buttons and separators.

In Windows, this method assumes the application window has been maximized.

Parameters

toolBar	The toolbar to check.
nButtons	The number of buttons.
nSeparators	The number of separators.

Return Value

true if there is room in toolBar to add nButtons and nSeparators, false otherwise.

Header File

AVCalls.h

Related Methods

```
AVAppGetToolBar  
AVToolBarGetFrame  
AVToolBarGetNumButtons  
AVToolButtonGetFlyout  
AVToolButtonNew
```

Example

```
AVToolbar mybar = AVAppGetToolBar();  
freeSlot = AVToolBarIsRoomFor(mybar, 1, 0);  
if (freeSlot){  
    /* add new button */  
    ...  
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVToolBarNew

[AVToolBar](#) AVToolBarNew (const char* name, const char* title);

Description

Creates a new named toolbar. [AVAppRegisterToolBarPosition](#) must be called after creating the new toolbar to position it relative to other toolbars.

Parameters

name	The internal, language-independent name of the toolbar. May not be NULL .
title	The localized, user-friendly name of the toolbar. May not be NULL .

Return Value

The new **AVToolBar**.

Header File

AVCalls.h

Related Methods

[AVToolBarGetFrame](#)
[AVToolBarGetNumButtons](#)
[AVToolButtonGetFlyout](#)
[AVToolButtonNew](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVToolBarNewFlyout

```
AVToolBar AVToolBarNewFlyout (void);
```

Description

Creates a new sub-toolbar for use as a toolbar button *flyout*.

Flyouts are established by creating a new toolbar with **AVToolBarNewFlyout**, appending toolbar buttons to the new toolbar using **AVToolBar** calls, and attaching that toolbar to a tool button, known as the *anchor button*, with **AVToolButtonSetFlyout**.

This method creates a distinct toolbar from the toolbar returned by **AVAppGetToolBar**.

NOTE: The viewer makes a copy of the anchor button and attaches it to the front of the flyout to achieve the correct visual effect; the client does not need to do this.

Parameters

None

Return Value

The newly created toolbar to use for a flyout.

Header File

AVCalls.h

Related Methods

[AVToolButtonGetFlyout](#)

[AVToolButtonSetFlyout](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

AVToolBarUpdateButtonStates

```
void AVToolBarUpdateButtonStates (AVToolBar toolbar);
```

Description

Forces a redraw of **toolbar**. Call this method when a toolbar button is added or removed or one of the buttons changes state.

Parameters

toolbar	The toolbar to redraw.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVAppGetToolBar](#)
[AVToolBarGetFrame](#)
[AVToolBarIsRoomFor](#)
[AVToolButtonGetFlyout](#)

Example

```
AVToolBarUpdateButtonStates (
    AVAppGetToolBar () );
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVToolButton

AVToolButtonDestroy

```
void AVToolButtonDestroy (AVToolButton toolButton);
```

Description

Removes the specified button from the toolbar and destroys the button. Call [AVToolBarUpdateButtonStates](#) after removing a button to update the toolbar.

Parameters

toolButton	The button to destroy.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVToolButtonNew](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVToolButtonExecute

```
void AVToolButtonExecute (AVToolButton button);
```

Description

Executes the [AVExecuteProc](#) associated with **button**, if it exists. This [AVExecuteProc](#) is set by [AVToolButtonSetExecuteProc](#). Does nothing if [AVToolButtonIsEnabled](#) for the button returns **false**.

Parameters

button	The button whose execute proc is executed.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVToolButtonIsEnabled](#)
[AVToolButtonNew](#)
[AVToolButtonSetExecuteProc](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVToolButtonGetFlyout

```
AVToolBar AVToolButtonGetFlyout (AVToolButton button);
```

Description

Gets the *flyout* attached to a toolbar button. A flyout is a sub-toolbar attached to a toolbar button.

This method gets a *different* toolbar from the toolbar returned by [AVAppGetToolBar](#).

Parameters

button	The toolbar button whose flyout is obtained.

Return Value

The flyout associated with **button**. Returns **NULL** if there is no flyout associated with **button**.

Header File

AVCalls.h

Related Methods

[AVToolBarNewFlyout](#)
[AVToolButtonSetFlyout](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.



AVToolButtonGetIcon

```
AVIcon AVToolButtonGetIcon (AVToolButton button);
```

Description

Gets the icon associated with the specified **AVToolButton**.

Parameters

button	The button whose icon is obtained.

Return Value

The icon associated with **button**.

Header File

AVCalls.h

Related Methods

[AVToolButtonSetIcon](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

AVToolButtonGetLabelText

```
ASBool AVToolButtonGetLabelText (AVToolButton button,
ASText resultText, AVToolButtonLabelPriority *priority);
```

Description

Gets the label text associated with the specified **AVToolButton** and its priority value. The priority determines the preference order in which labels are shown when a toolbar is too short to hold all of the button labels. If the priority is less than **kAVButtonPriorityOnExtraLow** the label text is not shown at all unless the user forces all labels to be shown using the General preference panel.

Parameters

button	The button whose label text is obtained.
resultText	<i>(Filled by the method)</i> The label text.
priority	<i>(Filled by the method)</i> The label text's priority value.

Return Value

true if the label text string and priority are successfully obtained, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVToolButtonSetLabelText](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVToolButtonGetMenu

```
AVMenu AVToolButtonGetMenu (AVToolButton button);
```

Description

Gets the menu attached to a toolbar button.

Parameters

button	The toolbar button to which a menu is attached.

Return Value

The menu attached to **button**.

Header File

AVCalls.h

Related Methods

[AVPageViewDoPopupMenu](#)

[AVToolButtonSetMenu](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

AVToolButtonGetName

```
ASAtom AVToolButtonGetName (AVToolButton button);
```

Description

Gets the **ASAtom** corresponding to the name of the specified toolbar button.

Parameters

button	The toolbar button whose name is obtained.

Return Value

The **ASAtom** corresponding to the toolbar button's name. **ASAtom** can be converted to a character string using [ASAtomGetString](#). See [Toolbar and Toolbar Button Names](#) for a list of the built-in button names.

Header File

AVCalls.h

Related Methods

[AVToolBarGetButtonByName](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVToolButtonIsEnabled

```
ASBool AVToolButtonIsEnabled (AVToolButton button);
```

Description

Tests whether a toolbar button is enabled.

Parameters

button	The button to test.

Return Value

true if **button**'s [AVComputeEnabledProc](#) returns **true**, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVToolButtonSetComputeEnabledProc](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVToolButtonIsMarked

```
ASBool AVToolButtonIsMarked (AVToolButton button);
```

Description

Tests whether the specified button is marked.

Parameters

button	The button to test.

Return Value

true if **button**'s [AVComputeMarkedProc](#) returns **true**, **false** if **button** is not marked or does not have an [AVComputeMarkedProc](#).

Header File

[AVCalls.h](#)

Related Methods

[AVToolButtonSetComputeMarkedProc](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in [PIRequir.h](#)) is set to **0x00020000** or higher.

AVToolButtonIsVisible

```
ASBool AVToolButtonIsVisible (AVToolButton button) ;
```

Description

Tests whether the specified button is visible when its parent menu is opened.

Parameters

button	The button to test.

Return Value

true if **button**'s **AVComputeVisibleProc** returns **true**, or if it does not have an **AVComputeVisibleProc**.

false if **button**'s **AVComputeVisibleProc** returns **false**.

Header File

AVCalls.h

Related Methods

[AVToolButtonSetComputeVisibleProc](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVToolButtonIsSeparator

```
ASBool AVToolButtonIsSeparator (AVToolButton toolButton);
```

Description

Tests whether a toolbar button is a separator or a normal button.

Parameters

button	The button to test.

Return Value

true if the button is a separator, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVToolButtonNew](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVToolButtonNew

```
AVToolButton AVToolButtonNew (ASAtom name, AVIcon icon,
ASBool longOnly, ASBool isSeparator);
```

Description

Creates a toolbar button with the specified name, icon and long-menus state. Can also be used to create a separator with the specified name.

Parameters

name	The ASAtom corresponding to the button's name. The character string for name can be converted to an ASAtom using ASAtomFromString .
icon	The icon to use for this button. In Mac OS, icon must be a handle to a standard SICN resource. In Windows, icon is an 18×18 icon with a light gray background (that is, RGB values of 192,192,192).
longOnly	<i>(Ignored in Acrobat 3.0 or later)</i> If true , the button is shown only when the user selects "Full menus" in the Acrobat viewer. If false , shows in both "Full menu" and "Short menu" modes.
isSeparator	If true , the new button is a separator used to leave space between groups of related buttons. For separators, icon , the button's AVExecuteProc , AVComputeEnabledProc , and AVComputeMarkedProc are ignored. In addition, separators are not click-able. If false , the button is a normal toolbar button.

Return Value

The newly created button.

Known Exceptions

[genErrNoMemory](#)

Header File

AVCalls.h

Related Methods

[AVToolButtonDestroy](#)
[AVToolButtonSetExecuteProc](#)
[AVToolButtonSetComputeEnabledProc](#)
[AVToolButtonSetComputeMarkedProc](#)



Example

```
AVToolButton myButton;
AVToolBar mybar;

myButton = AVToolButtonNew(
    ASAtomFromString("Hello World"),
    GetResource('SICN', 600), FALSE, FALSE);
mybar = AVAppGetToolBar();
freeslots = AVToolBarIsRoomFor(mybar, 1,
    0);

if (freeslots){
    /* add new button after all others */
    AVToolButton oldButton =
        AVToolBarGetButtonByName(mybar,
            ASAtomFromString("EndDialogGroup"));
    AVToolBarAddButton(mybar, myButton,
        FALSE, oldButton);
    AVToolButtonSetExecuteProc(myButton,
        doHelloAlertCallback, NULL);
    AVToolButtonSetComputeEnabledProc(
        myButton, DocExistEnableCallback,
        NULL);
    AVToolButtonSetComputeMarkedProc(
        myButton, markedProcCallback, NULL);
}
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVToolButtonRemove

```
void AVToolButtonRemove (AVToolButton toolButton);
```

Description

Removes the specified button from the toolbar, but does not destroy the button. Call [AVToolBarUpdateButtonStates](#) after removing a button to update the toolbar.

Parameters

toolButton	The button to remove.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVToolBarAddButton](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVToolButtonSetComputeEnabledProc

```
void AVToolButtonSetComputeEnabledProc (AVToolButton button,
AVComputeEnabledProc proc, void* clientData);
```

Description

Sets the [AVComputeEnabledProc](#) associated with a toolbar button. This routine determines whether the button can be selected.

Parameters

button	The button whose AVComputeEnabledProc is set.
proc	User-supplied procedure to call whenever the Acrobat viewer needs to know whether button should be enabled.
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVToolButtonIsEnabled](#)

Example

```
AVToolButton oldButton =
    AVToolBarGetButtonByName (mybar,
        ASAtomFromString ("EndDialogGroup"));
AVToolBarAddButton (mybar, myButton, FALSE,
    oldButton);
AVToolButtonSetExecuteProc (myButton,
    doHelloAlertCallback, NULL);
AVToolButtonSetComputeEnabledProc (myButton,
    DocExistEnableCallback, NULL);
AVToolButtonSetComputeMarkedProc (myButton,
    markedProcCallback, NULL);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVToolButtonSetComputeMarkedProc

```
void AVToolButtonSetComputeMarkedProc (AVToolButton button,
AVComputeMarkedProc proc, void* clientData);
```

Description

Sets the [AVComputeMarkedProc](#) associated with a toolbar button. A marked button appears pressed on the screen.

Parameters

button	The button whose AVComputeMarkedProc is set.
proc	User-supplied callback to call whenever the Acrobat viewer needs to know whether the specified toolbar button should be marked.
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVToolButtonIsMarked](#)
[AVToolButtonNew](#)
[AVToolButtonSetComputeEnabledProc](#)
[AVToolButtonSetExecuteProc](#)

Example

```
AVToolButton oldButton =
    AVToolBarGetButtonByName(mybar,
        ASAtomFromString("EndDialogGroup"));
AVToolBarAddButton(mybar, myButton, FALSE, oldButton);
AVToolButtonSetExecuteProc(myButton,
    doHelloAlertCallback, NULL);
AVToolButtonSetComputeEnabledProc(myButton,
    DocExistEnableCallback, NULL);
AVToolButtonSetComputeMarkedProc(myButton,
    markedProcCallback, NULL);
```

Product

reader, base, pro

**Availability**

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVToolButtonSetComputeVisibleProc

```
void AVToolButtonSetComputeVisibleProc (AVToolButton button,  
AVComputeVisibleProc proc, void* clientData);
```

Description

Sets the [AVComputeVisibleProc](#) associated with a toolbar button. This routine determines whether the button is visible when its parent toolbar is visible.

Parameters

button	The toolbar button whose visibility procedure is set.
proc	User-supplied procedure to call when the Acrobat viewer needs to know if the button is visible.
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVToolButtonIsVisible](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVToolButtonSetExecuteProc

```
void AVToolButtonSetExecuteProc (AVToolButton button,
AVExecuteProc proc, void* clientData);
```

Description

Sets the user-supplied procedure to call to actually “do” whatever the button does.

Parameters

button	The button whose AVExecuteProc is set.
proc	User-supplied procedure to call when button is executed.
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVToolButtonNew](#)
[AVToolButtonSetComputeEnabledProc](#)
[AVToolButtonSetComputeMarkedProc](#)

Example

```
AVToolButton oldButton =
    AVToolBarGetButtonByName(mybar,
        ASAtomFromString("EndDialogGroup"));
AVToolBarAddButton(mybar, myButton, FALSE,
    oldButton);
AVToolButtonSetExecuteProc(myButton,
    doHelloAlertCallback, NULL);
AVToolButtonSetComputeEnabledProc(myButton,
    DocExistEnableCallback, NULL);
AVToolButtonSetComputeMarkedProc(myButton,
    markedProcCallback, NULL);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVToolButtonSetExternal

```
void AVToolButtonSetExternal (AVToolButton button,
ASUns16 external);
```

Description

Indicates that the specified toolbar button should be displayed in toolbars contained in external windows, such as in a Web browser.

NOTE: Call **AVToolButtonSetExternal** before adding the toolbutton with **AVToolBarAddButton**. If you want to change the toolbutton's location after it has been added, call **AVToolButtonSetExternal** and re-add the button with **AVToolBarAddButton**.

Parameters

button	The button.
external	Indicates whether to show the button in external windows. Must be one of Tool Button Flags . NOTE: To enable a toolbar button in an external window, first remove the button from the toolbar, then turn on the TOOLBUTTON_EXTERNAL flag and add the button back on to the toolbar.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVToolButtonNew](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.



AVToolButtonSetFlyout

```
void AVToolButtonSetFlyout (AVToolButton button,
AVToolBar bar);
```

Description

Attaches a sub-toolbar or *flyout* to a toolbar button. A copy of the button is attached to the front of the toolbar. Click-hold pops up the *flyout* and allow the user to select a different button.

Flyouts are established by creating a new toolbar with [AVToolBarNewFlyout](#), appending toolbar buttons to the new toolbar using [AVToolBar](#) calls, and attaching that toolbar to a tool button, known as the *anchor button*, with [AVToolButtonSetFlyout](#).

Parameters

button	The toolbar button to attach to the flyout.
bar	The flyout to which button is attached.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVToolBarNewFlyout](#)
[AVToolButtonGetFlyout](#)

Product

reader, base, pro

Availability

Available if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.

AVToolButtonSetHelpText

```
void AVToolButtonSetHelpText (AVToolButton button,
    const char* text);
```

Description

Sets the text to show in “tooltips.” This text is shown when the cursor is held over a toolbar button for period of time.

In Acrobat 4.0 and later, toolbar buttons can also have single key shortcuts assigned to them. (Conceptually, tools have shortcuts but we attach the shortcuts to the buttons.) This is done by appending a vertical bar character “|” followed by the shortcut to the button’s tooltip text. For example, here’s the tooltip text for the Hand tool:

“Hand Tool (H)|h”

The trailing “|h” portion indicates that the Hand tool uses the ‘H’ key as the shortcut. This portion is stripped off before the tooltip is displayed.

This behavior only applies to tooltips where the “|” is the second-to-last character.

Appending the shortcut to the tooltip text allows it to localize at the same time as the tooltip text.

Parameters

button	The toolbar button to which a tooltip is added.
text	The text to show.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVToolButtonNew](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION**(in `PIRequir.h`) is set to **0x00020002** or higher.

AVToolButtonSetIcon

```
void AVToolButtonSetIcon (AVToolButton button, AVIcon icon);
```

Description

Sets a new icon for a toolbar button.

A tool button's icon can change dynamically. This allows multi-state buttons, such as the one that opens and closes the splitter bar, to change appearance appropriately. This should allow multiple buttons to collapse into one.

Most other aspects of a tool button, such as execute proc and tooltip text, can be changed on the fly using other **AVToolButton** methods.

Parameters

button	The toolbar button whose icon is set.
icon	The icon to place on button .

Return Value

None

Header File

AVCalls.h

Related Methods

[AVToolButtonGetIcon](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

AVToolButtonSetLabelText

```
void AVToolButtonSetLabelText (AVToolButton button,
                               AText resultText, AVToolButtonLabelPriority priority);
```

Description

Sets the label text associated with the specified **AVToolButton** and its priority value. The priority determines the preference order in which labels are shown when a toolbar is too short to hold all of the button labels. If the priority is less than **kAVButtonPriorityOnExtraLow** the label text is not shown at all unless the user forces all labels to be shown using the General preference panel.

Parameters

button	The button whose label text is obtained.
resultText	The label text.
priority	The label text's priority value.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVToolButtonGetLabelText](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVToolButtonSetMenu

```
void AVToolButtonSetMenu (AVToolButton button, AVMenu menu);
```

Description

Attaches a menu to a toolbar button.

If a tool button has no execute proc, the menu pops up when the tool button is clicked. If the tool button does have an execute proc, the user must click and hold on the button for some time to display the menu. Simply clicking on the button invokes the button's execute proc as usual.

Parameters

button	The toolbar button to which a menu is attached.
menu	The menu to attach to button .

Return Value

None

Header File

AVCalls.h

Related Methods

[AVPageViewDoPopupMenu](#)

[AVToolButtonGetMenu](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

AVToolButtonSetNotifyTooltipProc

```
void AVToolButtonSetNotifyTooltipProc (AVToolButton button,  
AVNotifyTooltipProc proc, void* clientData);
```

Description

Sets the [AVNotifyTooltipProc](#) associated with a toolbar button. This routine is called before text is displayed in the tooltip.

Parameters

button	The toolbar button whose tooltip procedure is set.
proc	User-supplied procedure to call when the Acrobat viewer needs to display tooltip text, as when the cursor moves over button .
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



AVUndo

AVUndoGetAVDoc

[AVDoc](#) AVUndoGetAVDoc ([AVUndo](#) undo)

Description

Gets the document whose undo list contains the undo record.

Parameters

undo	The undo record whose document is obtained.

Return Value

The [AVDoc](#) object.

Header File

AVCalls.h

Related Methods

[AVDocBeginUndoOperation](#)
[AVDocClearUndos](#)
[AVDocEndUndoOperation](#)
[AVDocGetTopUndo](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



AVUndoGetData

[AVUndoHandlerData](#) AVUndoGetData ([AVUndo](#) undo)

Description

Gets the client-defined private data for the undo record, as specified on creation. The handler data can be accessed by any of the callbacks listed in the [AVUndoHandler](#).

Parameters

<code>undo</code>	The undo record whose handler data is obtained.

Return Value

The handler data.

Header File

`AVCalls.h`

Related Methods

[AVUndoNew](#)

Product

reader, base, pro

Availability

Available if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

AVUndoGetType

```
const char* AVUndoGetType (AVUndo undo)
```

Description

Gets the type of the undo record, as specified in the [AVUndoHandler](#). The type is a client-defined string that can be matched for retrieval by [AVDocGetTopUndo](#).

Parameters

undo	The undo record whose type is obtained.

Return Value

The client-defined type string for the undo record.

Header File

AVCalls.h

Related Methods

[AVUndoNew](#)
[AVDocGetTopUndo](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVUndoNew

[AVUndo](#) AVUndoNew ([AVDoc](#) doc, [AVUndoHandler](#) handler, [AVUndoHandlerData](#) undoData)

Description

Creates a new [AVUndo](#) record for a document's undo list.

Parameters

doc	The document whose undo list contains this undo record.
handler	The handler structure containing callbacks that perform the undo and redo operations.
undoData	Any private data needed by the handler callbacks.

Return Value

The new [AVUndo](#) object.

Header File

AVCalls.h

Related Methods

[AVUndoGetAVDoc](#)
[AVUndoGetData](#)
[AVUndoGetType](#)
[AVUndoSetData](#)

[AVDocBeginUndoOperation](#)
[AVDocClearUndos](#)
[AVDocEndUndoOperation](#)
[AVDocGetTopUndo](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



AVUndoSetData

```
void AVUndoSetData (AVUndo undo,
AVUndoHandlerData handlerData)
```

Description

Sets or replaces the client-defined private data for the undo record. The handler data can be accessed by any of the callbacks listed in the [AVUndoHandler](#).

Parameters

undo	The undo record whose handler data is set.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVUndoNew](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



AVWindow

AVWindowBecomeKey

```
void AVWindowBecomeKey (AVWindow win);
```

Description

Makes **win** the key window (regardless of the setting of its **WantsKey** flag) if the window is visible.

UNIX users: This method is a no-op.

Parameters

win	The window that is to become the key window.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVWindowSetWantsKey](#)

[AVWindowIsVisible](#)

[AVWindowResignKey](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVWindowBringToFront

```
void AVWindowBringToFront (AVWindow win);
```

Description

Brings the specified window to the front.

Parameters

win	The window to bring to the front.

Return Value

None

Notifications

[AVAppFrontDocDidChange](#)

Header File

AVCalls.h

Related Methods

[AVWindowBecomeKey](#)

[AVWindowIsKey](#)

[AVWindowSetWantsKey](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



AVWindowCenter

```
void AVWindowCenter (AVWindow win);
```

Description

Centers the window within its parent window or the desktop if it has no parent.

Parameters

win	The window to center.

Return Value

None

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



AVWindowDestroy

```
void AVWindowDestroy (AVWindow win);
```

Description

Destroys the specified window and all associated memory. Closes the window without calling the window handler's [AVWindowWillCloseProc](#) (that is, this operation cannot be rejected).

Parameters

win	The window to destroy.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVWindowNew](#)

[AVWindowNewFromPlatformThing](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVWindowDoModal

```
ASBool AVWindowDoModal (AVWindow win) ;
```

Description

Makes the specified window modal, so that no other window can take the keyboard focus when this window is active.

Parameters

win	The window to make modal.

Return Value

true if successful, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVWindowEndModal](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



AVWindowDrawNow

```
void AVWindowDrawNow (AVWindow win);
```

Description

Redraws the invalid regions of the specified window .

Parameters

win	The window to draw .

Return Value

None

Header File

AVCalls.h

Related Methods

[AVWindowInvalidateRect](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVWindowEndModal

[ASBool](#) AVWindowEndModal ([AVWindow](#) win) ;

Description

Stops the specified window from being modal, so that other windows can take the keyboard focus when this window is active.

Parameters

win	The window to stop from being modal.

Return Value

true if successful, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVWindowDoModal](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVWindowEnsureInBounds

```
void AVWindowEnsureInBounds (AVWindow win);
```

Description

If the **win** is completely outside the desktop region (that is, off-screen), it is moved so that it will be within the desktop region (that is, on-screen). No action is taken if a draggable edge of the window is within the desktop region. The “desktop region” does not include the task bar (Windows) or menubar (Macintosh).

Parameters

win	The window to ensure is on-screen .

Return Value

None

Header File

AVCalls.h

Related Methods

[AVWindowGetDesktopBounds](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

AVWindowGetBorderWidths

```
void AVWindowGetBorderWidths (AVWindow win, AVRect* rect);
```

Description

Fills a rectangle structure with the left, top, right, and bottom distances between inner window rectangle (the frame rectangle) and outer window rectangle. These distances include borders, title bars, and so on.

Parameters

win	The window whose border width is obtained.
rect	<i>(Filled by the method)</i> Pointer to a rectangle structure in which to return the window's border width rectangle, specified in global screen coordinates.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVWindowGetFrame](#)

[AVWindowGetInterior](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVWindowGetDesktopBounds

```
ASBool AVWindowGetDesktopBounds (AVWindow win,
const AVRect *frame, AVRect *bounds);
```

Description

Gets the desktop boundary for the monitor on which the window would appear if its frame was set to the specified **frame**. This boundary describes in global coordinates the visible area of the desktop (that is, without the menu bar, task bar, docked toolbars, and so on).

Parameters

win	The window whose desktop bounds are obtained.
frame	Pointer to a rectangle structure for a potential window frame.
bounds	<i>(Filled by the method)</i> Pointer to a structure in which to return the window's desktop bounds rectangle, specified in global screen coordinates.

Return Value

true if successful, **false** if the method cannot provide the information. In this case, the bounds are set to (0,0,0,0).

Header File

AVCalls.h

Related Methods

[AVWindowEnsureInBounds](#)
[AVWindowGetFrame](#)
[AVWindowGetInterior](#)
[AVWindowSetFrame](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

AVWindowGetFrame

```
void AVWindowGetFrame (AVWindow win, AVScreenRect* rect);
```

Description

Gets the window's frame, which specifies its size and location on the screen.

NOTE: In Mac OS, this method may change the current port, thus altering the Macintosh graphics state. It sets the port to that specified by win, but fails to restore the previous port before exiting.

Parameters

win	The window whose frame is obtained.
rect	<i>(Filled by the method)</i> Pointer to a rectangle specifying the window's frame rectangle, specified in global screen coordinates. In Mac OS, the frame includes only the window's content region. In Windows, it includes the entire window (content region, title bar, and so forth).

Return Value

None

Header File

AVCalls.h

Related Methods

[AVWindowSetFrame](#)
[AVWindowGetInterior](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVWindowGetInterior

```
void AVWindowGetInterior (AVWindow win, AVWindowRect* rect);
```

Description

Gets the interior rectangle of the window.

Parameters

win	The window whose interior is obtained.
rect	<i>(Filled by the method)</i> Pointer to a rectangle specifying the window's interior, specified as local window coordinates.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVWindowGetFrame](#)

[AVWindowSetFrame](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVWindowGetMinMaxSize

```
void AVWindowGetMinMaxSize (AVWindow win, AVRect* rect);
```

Description

Gets the minimum and maximum size of the window.

Top and **left** in **rect** are for minimum size; **bottom** and **right** are for maximum size.

Parameters

win	The window whose size is obtained.
rect	<i>(Filled by the method)</i> Pointer to a rectangle specifying the window's size, specified in device space coordinates. If this method is not supported on a particular platform, the minimum size returned is 0, 0 and the maximum size is ASMAXInt16, ASMAXInt16 .

Return Value

None

Header File

AVCalls.h

Related Methods

[AVWindowSetMinMaxSize](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00040005** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVWindowGetOwnerData

```
void* AVWindowGetOwnerData (AVWindow win);
```

Description

Gets a window's owner data. The owner data is private data for the use of the window's creator. For example, if a client uses its own class library, it might use the owner data field to store a pointer to the object owning the **AVWindow**.

Parameters

win	The window whose owner data is obtained.

Return Value

Pointer to owner data for **win**.

Header File

AVCalls.h

Related Methods

[AVWindowSetOwnerData](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVWindowGetPlatformThing

```
void* AVWindowGetPlatformThing (AVWindow win);
```

Description

Returns a pointer to the platform-specific thing associated with the window. Do not confuse this with the owner data (see also [AVWindowGetOwnerData](#)).

Parameters

win	Platform-dependent window thing: a WindowPtr in Mac OS, an HWND in Windows, and a Widget in UNIX.

Return Value

The platform-dependent thing for the window. **NULL** if the window is associated with an **AVDoc** that has been set dead by [AVDocSetDead](#).

Known Exceptions

[AVDocWantsToDie](#) is broadcast after [AVDocSetDead](#) has been called, which would warn that the **AVWindow** associated with that window is **NULL**.

Header File

AVCalls.h

Related Methods

[AVDocGetAVWindow](#)
[AVWindowNewFromPlatformThing](#)
[AVWindowNew](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVWindowGetTitle

```
void AVWindowGetTitle (AVWindow win, AText title);
```

Description

Gets the title to display in the specified window's title bar.

Parameters

win	The window whose title is obtained.
title	<i>(Filled by the method)</i> The window title as an AText object.

Return Value

None

Known Exceptions

None

Header File

AVCalls.h

Related Methods

[AVWindowSetTitle](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

NOTE: The parameters changed in number and type and the return type changed in **0x00060000**.



AVWindowHandlePlatformEvent

```
ASBool AVWindowHandlePlatformEvent (AVWindow win,
void* platformEvent);
```

Description

Handles a platform-specific event. Use this method to dispatch a platform-specific event structure to an **AVWindow**.

Parameters

win	The window with the event to handle.
platformEvent	A pointer to a platform-specific event structure.

Return Value

true if the event was handled, **false** otherwise.

Known Exceptions

May raise exceptions, depending on the event.

Header File

AVCalls.h

Related Methods

[AVAppHandleAppleEvent](#)
[AVAppHandlePlatformEvent](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.



AVWindowHide

```
void AVWindowHide (AVWindow win);
```

Description

Hides the specified window. Hiding a window makes it invisible, it does not minimize or icon-ize it.

In Windows, a document window can be minimized using code based on the following:

```
theAVWindow = AVDocGetAVWindow(theAVDoc);
theThing = (HWND)AVWindowGetPlatformThing(
    theAVWindow);
wasVisible = ShowWindow(theThing,
    SW_MINIMIZED);
```

Parameters

win	The window to hide.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVWindowShow](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVWindowInvalidateRect

```
void AVWindowInvalidateRect (AVWindow win,  
    const AVWindowRect* rect);
```

Description

Invalidates the specified rectangular region, for eventual redraw. This is the preferred method for refreshing a portion of an **AVWindow**. Use [AVWindowDrawNow](#) to force a window to redraw its invalid regions.

Parameters

win	The window in which to invalidate a region.
rect	Pointer to a rectangle specifying the region to invalidate.

Return Value

None

Header File

AVCalls.h

Related Methods

- [AVWindowDrawNow](#)
- [AVWindowGetInterior](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVWindowIsKey

```
ASBool AVWindowIsKey (AVWindow win);
```

Description

Tests whether the specified window is the key window. The key window is the window that receives mouse clicks.

UNIX users: This method is a no-op.

Parameters

win	The window to test.

Return Value

true if **win** is the key window, **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVWindowBecomeKey](#)
[AVWindowBringToFront](#)
[AVWindowSetWantsKey](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVWindowIsVisible

```
ASBool AVWindowIsVisible (AVWindow win);
```

Description

Tests whether a window is displayed on the screen.

Parameters

win	The window whose visibility is tested.

Return Value

true if the window is displayed on the screen (even if it is currently obscured by another window), **false** otherwise.

Header File

AVCalls.h

Related Methods

[AVWindowBecomeKey](#)
[AVWindowHide](#)
[AVWindowShow](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVWindowMaximize

```
void AVWindowMaximize (AVWindow win, ASBool maximize);
```

Description

Maximizes the specified window. In Mac OS, this corresponds to calling the **ZoomWindow** Toolbox function.

Parameters

win	The window to maximize.
maximize	true to maximize the window, false to return it to its non-maximized state.

Return Value

None

Header File

AVCalls.h

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVWindowNew

```
AVWindow AVWindowNew (AVWindowLayer layer, AVFlagBits32 flags,
AVWindowHandler handler, ASExtension owner);
```

Description

Creates a new window and registers it with the Acrobat viewer.

Because Windows and UNIX use the platform's native window handling instead of the Acrobat viewer's [AVWindowHandler](#) mechanism (that is, the [AVWindowHandler](#)'s callbacks are never called on those platforms), there is no advantage to using [AVWindowNew](#). Client developers on those platforms should use [AVWindowNewFromPlatformThing](#) instead of this method.

Parameters

layer	The layer in which the window resides. In Mac OS, must be one of the AVWindowLayer constants. In Windows, all AVWindows are of type AVWLfloating , and layer is ignored.
flags	An OR of the values listed in AVWindow Flags .
handler	A structure containing the window handler's callback functions. Pass NULL in Windows and UNIX, because the Window handler's callbacks are unused on those platforms.
owner	The gExtensionID extension registering the window.

Return Value

The newly created window.

Known Exceptions

[genErrNoMemory](#)

Header File

AVCalls.h

Related Methods

[AVWindowNewFromPlatformThing](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in [PIRequir.h](#)) is set to **0x00020000** or higher.



NOTE: The flags numeric type changed in version 0x00060000.

AVWindowNewFromPlatformThing

```
AVWindow AVWindowNewFromPlatformThing (AVWindowLayer layer,
AVFlagBits32 flags, AVWindowHandler handler,
ASExtension owner, void* platformThing);
```

Description

Creates a new window from a platform-specific structure and registers it with the Acrobat viewer.

If a user creates an **HWND** or a **WindowPtr** and does not register it with the viewer via **AVWindowNewFromPlatformThing**, it should not be allowed to persist across event loops (that is, in Mac OS, it should be system-modal).

Windows and UNIX use the platform's native window handling instead of the Acrobat viewer's **AVWindowHandler** mechanism (that is, the **AVWindowHandler**'s callbacks are *never* called on those platforms). Client developers on those platforms should use **AVWindowNewFromPlatformThing** instead of **AVWindowNew**, since they have to create the window using platform-specific code anyway.

Parameters

layer	One of the AVWindowLayer constants, specifying the layer in which the window is located. For Mac OS, see AVWindowNew . layer is ignored in Windows, because the equivalent information was specified when the platform thing was created.
flags	For Mac OS, an OR of the AVWindow Flags . It is the responsibility of the Macintosh client developer to ensure that flags matches any attributes of the "platform-thing" window; the Acrobat viewer cannot determine flags from the window itself. flags is ignored in Windows and UNIX, because the equivalent information was specified when the platform thing was created.
handler	A structure containing the window handler's callback functions. Pass NULL in Windows and UNIX, because the Window handler's callbacks are unused on those platforms.
owner	The gExtensionID extension registering the window.
platformThing	A platform-specific object (WindowPtr in Mac OS, an HWND in Windows, and a Widget in UNIX) that will be used for this window.

Return Value

The newly created window.

Known Exceptions

[genErrNoMemory](#)

Header File

AVCalls.h

Related Methods

[AVWindowNew](#)

[AVWindowGetPlatformThing](#)

Example

```
/* Under Windows, to do a modal dialog correctly, you want to add the
following to the WM_INITDIALOG portion of your dialog. */
avw = AVWindowNewFromPlatformThing(AVWLmodal, 0, NULL, gExtensionID,
hWnd);
AVAppBeginModal(avw);

/* You'll also want to add the following to the WM_DESTROY portion of
the dialog. */
AVAppEndModal();
AVWindowDestroy(avw);
```

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

NOTE: The flags numeric type changed in version **0x00060000**.

AVWindowResignKey

```
void AVWindowResignKey (AVWindow win);
```

Description

Use this method when you want **win** to resign its key window status. Another window might pick up key window status as a result. You must first call [AVWindowSetWantsKey](#) with a value of **false** for the **wantsKey** parameter or your window may immediately become the key window again.

UNIX users: This method is a no-op.

Parameters

win	The window resigning key window status.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVWindowBecomeKey](#)
[AVWindowSetWantsKey](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVWindowSetFrame

```
void AVWindowSetFrame (AVWindow win,  
const AVScreenRect* rect);
```

Description

Sets the window’s frame, which specifies its size and location on the screen.

Parameters

win	The window whose frame is set.
rect	Pointer to a rectangle specifying the window’s frame rectangle, specified in global screen coordinates. In Mac OS, the frame includes only the window’s content region. In Windows, it includes the entire window (content region, title bar, and so forth).

Return Value

None

Header File

AVCalls.h

Related Methods

- AVWindowGetFrame
- AVWindowGetInterior

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.

AVWindowSetMinMaxSize

```
void AVWindowSetMinMaxSize (AVWindow win,  
const AVRect* rect);
```

Description

Sets the minimum and maximum size of the window.

Top and **left** in **rect** are for minimum size; **bottom** and **right** are for maximum size.

Parameters

win	The window whose size is set.
rect	Pointer to a rectangle specifying the window's size, specified in device space coordinates.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVWindowGetMinMaxSize](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00040005** or higher.

NOTE: The coordinate numeric type changed in **0x00060000**.



AVWindowSetOwnerData

```
void AVWindowSetOwnerData (AVWindow win, void* newData);
```

Description

Sets a window's owner data. The owner data is private data for the use of the window's creator. For example, if a client uses its own class library, it might use the owner data field to store a pointer to the object owning the **AVWindow**.

Parameters

win	The window whose owner data is set.
newData	Pointer to the new owner data for win .

Return Value

None

Header File

AVCalls.h

Related Methods

[AVWindowGetOwnerData](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

AVWindowSetTitle

```
void AVWindowSetTitle (AVWindow win, AText newTitle);
```

Description

Sets the title to display in the specified window's title bar. This method cannot be used to set the title of the window in which the Acrobat viewer normally opens a PDF file; it can only be used to set the title of an **AVWindow** created by a client. To set the title of a window in which the Acrobat viewer opens a PDF file, you must replace [AVDocOpenFromASFileWithParams](#) and pass the window title in **tempTitle**.

Parameters

win	The window whose title is set.
newTitle	The title to set for win .

Return Value

None

Known Exceptions

[genErrNoMemory](#)

Header File

AVCalls.h

Related Methods

[AVWindowGetTitle](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

NOTE: The title parameter type changed in **0x00060000**.

AVWindowSetWantsKey

```
void AVWindowSetWantsKey (AVWindow win, ASBool wantsKey);
```

Description

Sets or clears a flag indicating that **win** wants to become the key window. This corresponds to the **AVWIN_WANTSKEY** flag that can be specified when the window is created using [AVWindowNew](#).

Once this flag is set, the Acrobat viewer may, at any time, make this window the key window. This occurs, for example, if the current key window closes and this window moves to the front. If **WantsKey** is **false**, this window will not become the key window. If the window is already the key window, however, simply invoking **AVWindowSetWantsKey** with **wantsKey** set to **false** will not cause it to resign key window status; to do that, you must call [AVWindowResignKey](#).

Parameters

win	The window that wants to become the key window.
wantsKey	If true , win is willing to become the key window, false otherwise.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVWindowBecomeKey](#)
[AVWindowResignKey](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVWindowShow

```
void AVWindowShow (AVWindow win);
```

Description

Shows the specified window.

Parameters

win	The window to show.

Return Value

None

Header File

AVCalls.h

Related Methods

[AVWindowHide](#)
[AVWindowIsVisible](#)

Product

reader, base, pro

Availability

Available if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

AVWindowUserClose

ASBool AVWindowUserClose (AVWindow win, ASBool quitting);

Description

Simulates a user’s click on a window’s box. This calls the AVWindowWillCloseProc of win’s AVWindowHandler.

Parameters

win	The window to close.
quitting	If true, assume the application is trying to quit. If false, assume the user clicked on the window’s close box.

Return Value

true if the window was closed, false if the window handler’s AVWindowWillCloseProc aborted the close by returning false.

Header File

AVCalls.h

Related Methods

AVWindowCenter
AVWindowHide

Product

reader, base, pro

Availability

Available if PI_ACROVIEW_VERSION (in PIRequir.h) is set to 0x00020000 or higher.



Cos Layer Methods

General

CosArray

CosBoolean

CosDict

CosDoc

CosFixed

CosInteger

CosName

CosNull

CosObj

CosObjCollection

CosStream

CosString

Encryption/Decryption



General

CosSetMaxDocStorages

```
void CosSetMaxDocStorage (ASInt32 maxMemory) ;
```

Description

Puts a limit on the amount of memory (RAM) that can be used to store Cos objects. The default limit is 2MB, and this method can be used only to increase the limit. Beyond the limit, Cos objects may be stored on disk.

NOTE: The limit applies only to fixed-size data in Cos objects, not to variable data stored in strings, arrays, dictionaries and streams. In some cases, objects may need to stay in memory, even if the limit is exceeded.

Parameters

maxMemory	The maximum amount of RAM (in bytes) that will be used to store fixed-size Cos objects.

Return Value

None

Header File

CosCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



CosArray

CosArrayGet

```
CosObj CosArrayGet (CosObj array, ASArraySize index);
```

Description

Gets the specified element from an array.

Parameters

array	The array from which an element is obtained.
index	The array element to obtain. The first element in an array has an index of zero.

Return Value

The Cos object occupying the **index**th element of **array**. Returns a null Cos object if **index** is outside the array bounds.

Header File

CosCalls.h

Related Methods

[CosArrayLength](#)

[CosArrayPut](#)

[CosArrayInsert](#)

Example

```
ASInt32 i;  
CosObj array, arrayEntry;  
ASInt32 n = CosArrayLength (array);  
for (i = 0 ; i <n; i++)  
{  
    arrayEntry = CosArrayGet (array, i);  
    /*do something with arrayEntry */  
    ...  
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

CosArrayInsert

```
void CosArrayInsert (CosObj array, ASTArraySize pos,  
                    CosObj obj);
```

Description

Inserts an object into an array.

Parameters

array	The array into which the object is inserted.
pos	The location in the array to insert the object. The object is inserted before the specified location. The first element in an array has a pos of zero. If pos $\geq$ CosArrayLength (array), obj is added at the end of the array. The length of the array always increases by 1.
obj	The object to insert.

Return Value

None

Known Exceptions

Raises an exception if object to insert is a direct object that is already contained in another object or if object to insert belongs to another document.

Header File

`CosCalls.h`

Related Methods

[CosArrayLength](#)
[CosArrayRemove](#)
[CosArrayGet](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



CosArrayLength

```
ASTArraySize CosArrayLength (CosObj array);
```

Description

Gets the number of elements in **array**.

Parameters

array	The array for which the number of elements is determined.

Return Value

The number of elements in **array**.

Known Exceptions

File access, memory, object type.

Header File

CosCalls.h

Related Methods

Numerous

Example

```
if (CosArrayLength(cosArray) > MAXLEN)
    AAlertNote("Array too long");
```

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

CosArrayPut

```
void CosArrayPut (CosObj array, AStArraySize index,
                 CosObj obj);
```

Description

Puts the specified object into the specified location in an array. The array is extended as much as necessary and null objects are stored in empty slots. Sets the **PDDocNeedsSave** flag (see **PDDocSetFlags**) flag of **array**'s **CosDoc** if **array** is indirect or is a direct object with an indirect composite object at the root of its container chain.

Parameters

array	The array in which obj is stored.
index	The location in array to store obj . The first element of an array has an index of zero.
obj	The Cos object to insert into array .

Return Value

None

Known Exceptions

Raises an exception if object to insert is a direct object that is already contained in another object or if object to insert belongs to another document.

Header File

CosCalls.h

Related Methods

[CosArrayLength](#)
[CosArrayGet](#)
[CosArrayInsert](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

CosArrayRemove

```
void CosArrayRemove (CosObj array, CosObj obj);
```

Description

Finds the first element, if any, equal to the specified object and removes it from the array. [CosObjEqual](#) is used to determine whether an array element is equal to the specified object.

The array is compressed after removing the element. The compression is accomplished by moving each element following the deleted element to the slot with the next smaller index and decrementing the **array**'s length by 1.

Because the array is automatically compressed when an element is removed, it is not safe to call [CosArrayRemove](#) within your callback procedure for [CosObjEnum](#).

Parameters

array	The array from which obj is removed.
obj	The object to remove.

Return Value

None

Known Exceptions

File access, memory, object type.

Header File

CosCalls.h

Related Methods

[CosArrayInsert](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

CosArrayRemoveNth

```
void CosArrayRemoveNth (CosObj array, ASTArraySize pos);
```

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. Sets the “dirty” flag of array’s CosDoc.

Parameters

array	The CosArray to remove the member from.
pos	The index for the array member to remove. Array indices start at 0.

Return Value

None

Header File

CosCalls.h

Related Methods

[CosArrayRemove](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in PIREquir.h) is set to **0x00040000** or higher.



CosNewArray

```
CosObj CosNewArray (CosDoc dP, ASBool indirect,
ASArraySize nElements);
```

Description

Creates and returns a new array Cos object.

Parameters

dP	The document in which the array is used.
indirect	If true , creates the array as an indirect Cos object, and sets the document's PDDocNeedsSave flag (see PDDocSetFlags). If false , creates the array as a direct object.
nElements	The number of elements that will be in the array. nElements is only a hint; Cos arrays grow dynamically as needed.

Return Value

The newly created array Cos object.

Header File

CosCalls.h

Related Methods

[CosObjDestroy](#)
[CosArrayGet](#)
[CosArrayInsert](#)
[CosArrayLength](#)
[CosArrayPut](#)
[CosArrayRemove](#)

Example

```
CosObj cosArray = CosNewArray(NULL, false, 10);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



CosBoolean

CosBooleanValue

```
ASBool CosBooleanValue (CosObj obj);
```

Description

Gets the value of the specified boolean object.

Parameters

<code>obj</code>	The boolean Cos object whose value is obtained.

Return Value

The value of `obj`.

Known Exceptions

Raises an exception if `obj` has the wrong Cos type.

Header File

`CosCalls.h`

Related Methods

[CosNewBoolean](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_COS_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.



CosNewBoolean

```
CosObj CosNewBoolean (CosDoc dP, ASBool indirect,
    ASBool value);
```

Description

Creates a new boolean object associated with the specified document and having the specified value.

Parameters

dP	The document in which the boolean is used.
indirect	If true , creates the boolean object as an indirect object, and sets the document dP 's PDDocNeedsSave flag (see PDDocFlags). If false , creates the boolean as a direct object.
value	The value the new boolean will have.

Return Value

A Cos boolean object.

Header File

CosCalls.h

Related Methods

```
CosBooleanValue
CosObjDestroy
```

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



CosDict

CosDictGet

```
CosObj CosDictGet (CosObj dict, ASAtom key);
```

Description

Gets the value of the specified key in the specified dictionary. If 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.

NOTE: Use [CosObjEnum](#) to list all key-value pairs in a dictionary.

Parameters

dict	The dictionary or stream from which a value is obtained.
key	The key whose value is obtained. See the PDF Reference to obtain the names of keys in dictionary objects that are part of standard PDF, such as annotations or page objects. NOTE: Use ASAtomFromString to convert the key name string to an ASAtom . Even though PDF dictionary keys are name objects and names begin with the slash character (/), the string passed to ASAtomFromString should not start with "/"—unless the key actually begins with "/".

Return Value

The object associated with the specified key. If **key** is not present or if its value is **null**, returns a Cos object of type **CosNull**.

Header File

CosCalls.h

Related Methods

[CosDictPut](#)
[CosDictKnown](#)
[CosStreamDict](#)

Example

```
CosObj theDict, intObj;

intObj = CosDictGet(theDict,
    ASAtomFromString ("Key1"));
```

**Product**

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

CosDictGetXAPMetadata

```
ASBool CosDictGetXAPMetadata (CosObj obj,
                              AText* metadataAText);
```

Description

Gets the XMP metadata associated with a Cos dictionary or stream.

NOTE: The term “XAP” refers to an early internal code name for Adobe’s Extensible Metadata Platform (XMP).

If there is XMP metadata, it is returned as an **AText** in the output parameter **metadataAText**. The **AText** returned becomes the property of the client, who is free to alter or destroy it.

NOTE: **CosDictGetXAPMetadata** will not attempt to verify that **obj** is one of the objects that is specified in the [PDF Reference](#) to allow XMP metadata.

Parameters

obj	A dictionary or stream CosObj .
metadataAText	<i>(Filled by the method)</i> The AText object from which the XMP metadata will be obtained.

Return Value

True if **obj** has associated XMP metadata, false if it does not. Also returns false if **obj** is not a dictionary or stream. Returns **true** exactly when the Cos object **obj** has XMP metadata.

Known Exceptions

[ErrSysPModel](#)
[pdMetadataErrCouldntCreateMetaXAP](#)

Header File

PDMetadataCalls.h

Related Methods

[CosDictSetXAPMetadata](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMETADATA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

CosDictKnown

```
ASBool CosDictKnown (CosObj dict, ASAtom key);
```

Description

Tests whether a specific key is found in the specified dictionary. Calling this method is equivalent to checking if the value returned from [CosDictGet](#) is a null Cos object.

Use [CosObjEnum](#) to obtain a list of all key-value pairs in a dictionary.

Parameters

dict	The dictionary in which to look for key . May be a dictionary or stream.
key	The key to find. See the PDF Reference to obtain the names of keys in dictionary objects that are part of standard PDF, such as annotations or page objects. NOTE: Use ASAtomFromString to convert the key name string to an ASAtom . Even though PDF dictionary keys are name objects and names begin with the slash character (/), the string passed to ASAtomFromString should not start with "/"—unless the key actually begins with "/".

Return Value

true if the value of a key is known (exists and is not null) in dict; **false** otherwise.

Known Exceptions

[cosErrExpectedDict](#)

Header File

CosCalls.h

Related Methods

[CosDictGet](#)

[CosDictPut](#)

**Example**

```
CosObj myDict;

if (CosDictKnown(myDict, ASAtomFromString ("Optional"))
{
    /* The key is in the dictionary. */
    ...
}
else
{
    /* That key is not in the dictionary. */
    ...
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

CosDictPut

```
void CosDictPut (CosObj dict, ASAtom key, CosObj val);
```

Description

Sets the value of a dictionary key, adding the key to the dictionary if it is not already present. Sets the **PDDocNeedsSave** flag (see [PDDocSetFlags](#)) of **dict**'s **CosDoc** if **dict** is indirect or is a direct object with an indirect composite object at the root of its container chain.

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.

NOTE: A dictionary entry whose value is **null** is equivalent to an absent entry; using **CosDictPut** to put a **null** value in a dictionary has the same effect as calling **CosDictRemove** to remove it from the dictionary.

Parameters

dict	The dictionary or stream in which a value is set.
key	The key whose value is set. See the PDF Reference to obtain the names of keys in dictionary objects that are part of standard PDF, such as annotations or page objects. NOTE: Use ASAtomFromString to convert the key name string to an ASAtom. Even though PDF dictionary keys are name objects and names begin with the slash character (/), the string passed to ASAtomFromString should not start with "/"—unless the key actually begins with "/".
val	The value to set.

Return Value

None

Known Exceptions

Raises an exception if object to insert is a direct object that is already contained in another object or if object to insert belongs to another document.

Header File

CosCalls.h

Related Methods

[CosDictGet](#)
[CosDictKnown](#)
[CosStreamDict](#)

**Example**

```
CosObj dict, intObj;  
CosDictPut(dict, ASAtomFromString("Key1"), intObj);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

CosDictRemove

```
void CosDictRemove (CosObj dict, ASAtom key);
```

Description

Removes a key–value pair from a dictionary. Sets the **PDDocNeedsSave** flag (see [PDDocSetFlags](#)) of **dict**'s **CosDoc** if the dictionary is indirect or has an indirect composite object at the root of its container chain.

Because of the way in which key–value pairs are represented in memory, it is not safe to call [CosDictRemove](#) within your callback procedure for [CosObjEnum](#).

Parameters

dict	The dictionary from which the key–value pair is removed.
key	The key to remove. See the PDF Reference to obtain the names of keys in dictionary objects that are part of standard PDF, such as annotations or page objects. NOTE: Use ASAtomFromString to convert the key name string to an ASAtom . Even though PDF dictionary keys are name objects and names begin with the slash character (/), the string passed to ASAtomFromString should not start with "/"—unless the key actually begins with "/".

Return Value

None

Header File

CosCalls.h

Related Methods

[CosDictGet](#)
[CosDictPut](#)

Example

```
CosObj dict;  
CosDictRemove(dict, ASAtomFromString("MyKey"));
```

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in [PIRequir.h](#)) is set to **0x00020000** or higher.

CosDictSetXAPMetadata

```
void CosDictSetXAPMetadata (CosObj obj,
    AStext metadataAStext) ;
```

Description

Sets the XMP metadata associated with a Cos dictionary or stream.

NOTE: The term “XAP” refers to an early internal code name for Adobe’s Extensible Metadata Platform (XMP).

Replaces the XMP metadata associated with the Cos object `obj` with the XMP metadata stored in `metadataAStext`.

The contents of `metadataAStext` must be well-formed XML and Resource Description Format (RDF) as defined by the W3C (see <http://www.w3.org/RDF>) that also forms valid XMP. `CosDictSetXAPMetadata` will not destroy `metadataAStext` or alter its text.

NOTE: `CosDictSetXAPMetadata` will raise an exception if the user does not have permission to change the document.

NOTE: `CosDictSetXAPMetadata` will not attempt to verify that `obj` is one of the objects that is specified in the [PDF Reference](#) to allow XMP metadata.

Parameters

<code>obj</code>	The dictionary or stream Cos object whose associated XMP metadata is to be set.
<code>metadataAStext</code>	The <code>AStext</code> object containing the metadata to be associated with <code>obj</code> .

Return Value

None

Known Exceptions

[ErrSysPModelError](#)

Header File

`PDFMetadataCalls.h`

Related Methods

[CosDictGetXAPMetadata](#)

Product

base, pro, pdf

**Availability**

Available if **PI_PDMETADATA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



CosNewDict

```
CosObj CosNewDict (CosDoc dP, ASBool indirect,
                  ASArraySize nEntries);
```

Description

Creates a new dictionary.

See the [PDF Reference](#) for information on dictionary objects that are part of standard PDF, such as annotations or page objects.

Parameters

dP	The document in which the dictionary is used.
indirect	If true , creates the dictionary as an indirect Cos object, and sets dP's PDDocNeedsSave flag (see PDDocFlags). If false , creates the dictionary as a direct object.
nEntries	Number of entries in the dictionary. This value is only a hint—Cos dictionaries grow dynamically as needed.

Return Value

The newly created dictionary Cos object.

Header File

CosCalls.h

Related Methods

[CosDictGet](#)
[CosDictKnown](#)
[CosDictPut](#)
[CosDictRemove](#)
[CosObjDestroy](#)

Example

```
PDDoc d;
CosDoc cd;
CosObj dict;
cd = PDDocGetCosDoc(d);
dict = CosNewDict(cd, false, 2);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



CosDoc

CosDocClose

```
void CosDocClose (CosDoc cosDoc);
```

Description

Closes a Cos document. You should only call this method with a document obtained via [CosDocOpenWithParams](#) to release resources used by the Cos document.

Parameters

cosDoc	The document to close.

Return Value

None

Header File

CosCalls.h

Related Methods

[CosDocOpenWithParams](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.



CosDocCreate

```
CosDoc CosDocCreate (ASFlagBits createFlags);
```

Description

Creates an empty Cos document.

Parameters

createFlags	An inclusive OR of bit flags that specify the attributes of a CosDoc when created by CosDocCreate . The only flag currently defined is: • cosDocCreateInfoDict (0x01): Creates an Info dictionary for the document.

Return Value

An empty Cos document.

Header File

CosCalls.h

Related Methods

[CosDocSaveToFile](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

CosDocEnumEOFs

```
ASBool CosDocEnumEOFs (CosDoc cosDoc, CosDocEnumEOFsProc proc,
void* clientData);
```

Description

Calls the specified procedure for each EOF in a given **CosDoc**, where the EOF is a position in a PDF file after a **%%EOF** keyword that marks the end of either a main cross-reference section, or an update cross-reference section that corresponds to an incremental save. Not every **%%EOF** keyword fits these criteria. For example, the first **%%EOF** in a linearized file does not, so its position is not be passed to **proc**.

Of **cosDoc** was created in memory (using **CosDocCreate**), or if it was damaged and needed to be repaired, the procedure is not called at all.

Parameters

cosDoc	The CosDoc in which the EOF's are enumerated.
proc	The CosDocEnumEOFsProc to call for each EOF.
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

true if all of the calls to **proc** return **true**. **false** as soon as a call to **proc** returns **false**.

Header File

CosCalls.h

Related Methods

[CosDocEnumIndirect](#)

Example

```
static ACCB1 ASBool ACCB2 myCosObjEnumDictProc(CosDoc cosDoc,
ASInt32 fileOffset, void* clientData);
{
    ...process this portion of the CosDoc...
    return true;
}
...
PDDoc myPdDoc;

CosDocEnumEOFsProc myEnumCB =
ASCallbackCreateProto(CosDocEnumEOFsProc, &myCosObjEnumDictProc);
CosDocEnumEOFs(PDDocGetCosDoc(myPdDoc), myEnumCB, &clientData);
```

**Product**

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

CosDocEnumIndirect

```
ASBool CosDocEnumIndirect (CosDoc dP, CosObjEnumProc proc,
    void* clientData);
```

Description

Enumerates all the indirect objects of a given **CosDoc**.
The objects are enumerated in no particular order. Successive enumerations of the same Cos document are not guaranteed to enumerate objects in the same order.
This method does not enumerate *invalid* objects; this includes objects that are defined as **null**, objects that are not defined at all (that is, have no cross-reference entry), and objects that are on the free list (see the [PDF Reference](#)).

Parameters

dP	The CosDoc whose indirect objects are enumerated.
proc	User-supplied callback to call for each indirect object in dP . Enumeration ends when proc returns false or all indirect objects have been enumerated. The value parameter returned in proc is always the null Cos object.
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

true if all of the calls to **proc** returned **true**. **false** as soon as a call to **proc** returns **false**.

Known Exceptions

Re-raises any exception that **proc** raises.

Header File

CosCalls.h

Related Methods

[CosObjEnum](#)



Example

```
static ACCB1 ASBool ACCB2
myCosObjEnumProc(CosObj obj,
CosObj value, void* clientData)
{
    ...process the indirect object...
    return true;
}
...
PDDoc myPdDoc;

CosObjEnumProc myEnumCB =
    ASCallbackCreateProto(CosObjEnumProc,
        &myCosObjEnumProc);
CosDocEnumIndirect(
    PDDocGetCosDoc(myPdDoc), myEnumCB,
    &clientData);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

CosDocGetID

```
ASBool CosDocGetID (CosDoc dP, CosByte** pInstanceID,  
    CosByte** pPermaID, ASTCount* instIDLength,  
    ASTCount* permIDLength) ;
```

Description

Returns two ID byte arrays identifying the **CosDoc**. The client should copy these arrays before making the next call to Acrobat.

Parameters

dP	The CosDoc whose ID byte arrays are returned.
pInstanceID	<i>(Filled by the method)</i> The instance ID.
pPermaID	<i>(Filled by the method)</i> The permanent ID.
instIDLength	The length of pInstanceID , in bytes.
permIDLength	The length of pPermaID , in bytes.

Return Value

true if the ID is returned, **false** otherwise.

Header File

CosCalls.h

Product

reader, base, pro, pdfi

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



CosDocGetInfoDict

```
CosObj CosDocGetInfoDict (CosDoc dP) ;
```

Description

Gets the specified document's Info dictionary. In general, access the document's Info dictionary using [PDDocGetInfo](#) and [PDDocSetInfo](#) wherever possible.

Parameters

dP	The document whose Info dictionary is obtained.

Return Value

The document's Info dictionary Cos object.

Header File

`CosCalls.h`

Related Methods

[CosDocGetRoot](#)
[PDDocGetInfo](#)
[PDDocSetInfo](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



CosDocGetObjByID

```
CosObj CosDocGetObjByID (CosDoc dP, ASInt32 objNum) ;
```

Description

Gets the indirect **CosObj** with the latest generation number.

Parameters

dP	The CosDoc to search for the matching Cos object.
objNum	The local master index for the indirect Cos object to return.

Return Value

The **CosObj** with the latest generation number whose ID (object number) equals **objNum**, or the **null** object if there is no object with this ID.

Header File

CosCalls.h

Related Methods

[CosObjGetID](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



CosDocGetRoot

```
CosObj CosDocGetRoot (CosDoc dP) ;
```

Description

Gets the Catalog (the root object) for the specified document. See Section 3.6.1 in the [PDF Reference](#) for a description of the Catalog.

Parameters

dP	The document whose Catalog is obtained.

Return Value

The document's Catalog dictionary Cos object.

Header File

CosCalls.h

Related Methods

[CosDocGetInfoDict](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

CosDocHasFullCompression

ASBool CosDocHasFullCompression (CosDoc dP);

Description

Tests whether the Cos document is fully compressed. In a fully compressed document, most objects are stored in *object streams*, which are normally Flate-encoded to reduce the size of the PDF file. Cross-reference information for these objects is stored in *cross-reference streams*, which are also normally Flate-encoded. See the [PDF Reference](#).

NOTE: Fully compressed files are not compatible with PDF 1.4 and earlier viewers.

Parameters

dP	The document whose compression is checked.

Return Value

true if the document is fully compressed, **false** otherwise.

Header File

CosCalls.h

Related Methods

[CosDocHasPartialCompression](#)

Product

reader, base, pro

Availability

Available if **PI_COS_VERSION** (in PIREquir.h) is set to **0x00060000** or higher.

CosDocHasPartialCompression

[ASBool](#) CosDocHasPartialCompression ([CosDoc](#) dP);

Description

Tests whether the Cos document is partially compressed. In a partially compressed file, the size of the logical structure information is reduced; however, this information is unavailable to pre-PDF 1.5 viewers, while the document can still be viewed and printed. PDF 1.5 viewers (such as Acrobat 6) have full access to the structure information.

In a partially compressed document, objects related to logical structure are stored in *object streams*, which are normally Flate-encoded to compress the document. Their cross-reference information is stored twice: in a *cross-reference stream*, to which there is a reference in the trailer of an update section; and in the main cross-reference table, which indicates that the objects are on the free list. See the [PDF Reference](#).

Parameters

dP	The document whose compression is checked.

Return Value

true if the document is partially compressed, **false** otherwise.

Header File

CosCalls.h

Related Methods

[CosDocHasFullCompression](#)

Product

reader, base, pro

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

CosDocOpenWithParams

CosDoc CosDocOpenWithParams (**CosDocOpenParams** params);

Description

Opens a Cos document. The document does not need to be a PDF document. In **params**, the client specifies a **fileSys** and **pathName** from which to open the document. The client may also specify a header string other than "%PDF-". For example, a client might want to open a private file type, for example, "%FDF-".

If the **doRepair** flag is set in the open flags, a minimal document can be opened. A minimal document contains the header string and a trailer dictionary. It may contain indirect objects before the trailer dictionary, and the trailer dictionary can refer to those objects. For example:

```
%FDF-1.0
1 0 obj
<< /Version /1.5
    /FDF << /F 20 0 R /JavaScript 5 0 R >>
>>
trailer
<<
/Root 1 0 R
>>
```

Parameters

params	Specifies how to open the document.

Return Value

A Cos document.

Known Exceptions

Various

Header File

CosCalls.h

Related Methods

[CosDocClose](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

CosDocSaveToFile

```
void CosDocSaveToFile (CosDoc cosDoc, ASFile asFile,
    CosDocSaveFlags saveFlags, CosDocSaveParams saveParams);
```

Description

Saves a Cos document to a file. **CosDocSaveToFile** will not generate a cross-reference index (table or stream) in the saved file. If you want the index to be generated, then you have to use **CosDocSaveWithParams**, which generates it by default.

Parameters

cosDoc	Document to save.
asFile	File to which the document is written; must be open in write mode. This file is not necessarily positionable.
saveFlags	An OR of the CosDocSaveFlags bit flag values specifying how to save the document.
saveParams	Optional parameters for use when saving a document, as described in CosDocSaveParams .

Return Value

None

Known Exceptions

cosErrAfterSave
cosErrNeedFullSave
genErrBadParm

Header File

CosCalls.h

Related Methods

CosDocCreate
CosDocSaveWithParams

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

CosDocSaveWithParams

```
void CosDocSaveWithParams (CosDoc cosDoc, ASFile asFile,
    CosDocSaveFlags saveFlags, CosDocSaveParams saveParams);
```

Description

Saves a Cos document, optionally to a new file. Generates a cross-reference index (table or stream) by default.

Parameters

cosDoc	The CosDoc for the document to save.
asFile	The file to which the document will be written. This file must already be open in write mode. If you pass NULL , cosDoc is saved to the file with which it was originally associated.
saveFlags	An OR of the CosDocSaveFlags bit flag values specifying how to save the document.
saveParams	CosDocSaveParams parameters for use when saving the CosDoc document.

Return Value

None

Known Exceptions

cosErrAfterSave
cosErrNeedFullSave
genErrBadParm

Header File

CosCalls.h

Related Methods

CosDocCreate
CosDocSaveToFile

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

CosDocSetDirty

```
void CosDocSetDirty (CosDoc cosDoc, ASBool isDirty);
```

Description

Sets a Cos document's dirty flag to a given boolean value. If this flag is **true** when the document is closed, it indicates that the document must be saved to preserve changes.

Parameters

cosDoc	The Cos document whose dirty flag is set.
isDirty	true if dirty, false otherwise.

Return Value

None

Header File

CosCalls.h

Related Methods

[CosDocSaveToFile](#)

[CosDocSaveWithParams](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.



CosFixed

CosFixedValue

```
ASFixed CosFixedValue (CosObj obj);
```

Description

Gets the value of `obj` as a real number.

Parameters

<code>obj</code>	The object whose value is obtained. It must have type CosInteger or CosFixed . The result is undefined if the real value is outside the range of ASFixed numbers.

Return Value

The numeric value of `obj`.

Known Exceptions

Raises an exception if the given object has the wrong Cos type.

Header File

`CosCalls.h`

Related Methods

[CosIntegerValue](#)
[CosNewFixed](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_COS_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

CosNewFixed

```
CosObj CosNewFixed (CosDoc dP, ASBool indirect,  
                   ASFixed value);
```

Description

Creates a new fixed number object, associated with the specified document and having the specified **value**.

Parameters

dP	The document in which the fixed number is used.
indirect	If true , creates the fixed number object as an indirect object, and sets the document dP 's PDDocNeedsSave flag (see PDDocFlags). If false , creates the fixed number as a direct object.
value	The numeric value the new fixed number.

Return Value

A fixed Cos object.

Header File

CosCalls.h

Related Methods

[CosFixedValue](#)
[CosNewInteger](#)
[CosObjDestroy](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



CosInteger

CosIntegerValue

```
ASInt32 CosIntegerValue (CosObj obj);
```

Description

Gets the integer value of a specified number object.

Parameters

obj	The object whose integer value is obtained. It must have type CosInteger or CosFixed . If it is CosFixed , its value is rounded to the nearest integer. The result is undefined if the real value is outside the range of ASInt32 numbers.

Return Value

The integer value of **obj**.

Known Exceptions

Raises an exception if the given object has the wrong Cos type.

Header File

`CosCalls.h`

Related Methods

[CosFixedValue](#)
[CosNewFixed](#)
[CosNewInteger](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



CosNewInteger

```
CosObj CosNewInteger (CosDoc dP, ASBool indirect,
                     ASInt32 value);
```

Description

Creates a new integer object associated with the specified document and having the specified value.

Parameters

dP	The document in which the integer is used.
indirect	If true , creates the integer object as an indirect object, and sets the document dP's PDDocNeedsSave flag (see PDDocFlags). If false , creates the integer as a direct object.
value	The value the new integer will have.

Return Value

An integer Cos object.

Header File

CosCalls.h

Related Methods

- CosIntegerValue
- CosNewFixed
- CosObjDestroy

Product

reader, base, pro, pdf

Availability

Available if PI_COS_VERSION (in PIRequir.h) is set to 0x00020000 or higher.



CosName

CosNameValue

```
ASAtom CosNameValue (CosObj obj);
```

Description

Gets the value of a name object.

Parameters

obj	The object of type CosName whose value is obtained.

Return Value

The **ASAtom** corresponding to the specified name object. **ASAtom** can be converted to a string using **ASAtomGetString**.

Known Exceptions

Raises an exception if **obj** has the wrong type, if storage is exhausted, or if file access fails.

Header File

CosCalls.h

Related Methods

CosNewName

Example

```
ASAtom nameAtom = CosNameValue(cosObj);
if (nameAtom == ASAtomFromString("Roger"))
{
    ...
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

CosNewName

[CosObj](#) CosNewName ([CosDoc](#) dP, [ASBool](#) indirect, [ASAtom](#) name);

Description

Creates a new name object associated with the specified document and having the specified value.

Parameters

dP	The document in which the new name is used.
indirect	If true , creates the name as an indirect object, and sets the document's PDDocNeedsSave flag (see PDDocFlags) flag. If false , creates the name as a direct object.
name	The ASAtom corresponding to the name to create. A C string can be converted to an ASAtom using ASAtomFromString .

Return Value

The newly created name Cos object.

Header File

CosCalls.h

Related Methods

[CosNameValue](#)
[CosObjDestroy](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in [PIRequir.h](#)) is set to **0x00020000** or higher.



CosNull

CosNewNull

```
CosObj CosNewNull (void);
```

Description

Returns a direct object of type **CosNull**. This **null** object is said to be invalid. You can compare an object to **null** using either of the following methods:

```
CosObjEqual (obj, CosNewNull());
CosObjGetType(obj) == CosNull;
```

In general, use **CosNewNull** only to initialize a local variable or pass a parameter. Null objects may be stored as array elements, but not as dictionary values. The following statements are equivalent:

```
CosDictPut (dict, key, CosNewNull());
CosDictRemove (dict, key);
```

Parameters

None.

Return Value

A **null** Cos object.

Header File

CosCalls.h

Related Methods

[CosObjGetType](#)

Examples

```
CosArrayPut (arrayObj, CosNewNull(), 6);

CosStreamSetData (streamObj, stm, 0, false, CosNewNull(), 0);

dObj = CosDictGet (dictObj, ASAtomFromString ("D"));
if (CosObjGetType (dObj) == CosNull)
    AVAlertNote("Expected a non-NULL value");
```

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



CosObj

CosDocObjIsWithinRange

```
ASBool CosDocObjIsWithinRange (CosObj obj,
                               ASInt32 byteRanges[], ASInt32 numEntries);
```

Description

Tests whether the definition of a specified Cos object, in the file associated with the object's [CosDoc](#), begins within any of a set of byte ranges. The test is inclusive; that is the object may begin at the first or last byte of a range.

Parameters

obj	The Cos object (must be indirect).
byteRanges	An array containing pairs of byte offsets within the document. Each pair is a start and end offset from the beginning of the document.
numEntries	The number of byte offsets (not pairs) in the byteRanges array.

Return Value

true if the object begins within any of the given ranges and has not been modified, **false** otherwise.

Known Exceptions

Raises an exception if **obj** is a direct object or **numEntries** is an odd number.

Header File

CosCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

CosObjAddToCollection

```
ASBool CosObjAddToCollection (CosObjCollection coll,
                              CosObj obj);
```

Description

Adds a Cos object to a collection; see [CosObjCollection](#) for requirements of these collections. This method sets the dirty flag of the collection's Cos document.

Parameters

<code>coll</code>	The Cos object collection.
<code>obj</code>	The object to add.

Return Value

true if `obj` was successfully added to the collection, **false** otherwise.

Known Exceptions

Raises an exception if the collection and the object belong to different Cos documents.

Header File

`CosCalls.h`

Related Methods

[CosObjGetCompressibility](#)
[CosObjIsCompressed](#)
[CosObjRemoveFromCollection](#)
[CosObjSetCompressibility](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

CosObjCmp

```
ASInt32 CosObjCmp (CosObj obj1, CosObj obj2);
```

Description

Compares the two **CosObjs**. The result is 0 if and only if **CosObjEqual(obj1, obj2)** is **true**. Otherwise, the result is either -1 or 1. The result is useful for ordering or sorting Cos objects. No other significance should be attached to the result. In particular, a nonzero result indicates nothing about the type of either object.

The result is valid only within a single instance of the document. That is, if **CosObjCmp** returns a nonzero value and the document is closed and then reopened, there is no guarantee that it will return the same nonzero value for those same objects.

The following conditions apply:

- If **CosObjCmp (a, b) == 0**, then **CosObjCmp (b, a) == 0**.
- If **CosObjCmp (a, b) > 0**, then **CosObjCmp (b, a) < 0**.
- If **CosObjCmp (a, b) < 0**, then **CosObjCmp (b, a) > 0**.
- If **CosObjCmp (a, b) == 0**, and **CosObjCmp (b, c) == 0**, then **CosObjCmp (a, c) == 0**.
- If **CosObjCmp (a, b) > 0**, and **CosObjCmp (b, c) > 0**, then **CosObjCmp (a, c) > 0**.
- If **CosObjCmp (a, b) < 0**, and **CosObjCmp (b, c) < 0**, then **CosObjCmp (a, c) < 0**.

Parameters

obj1	The first CosObj to compare.
obj2	The second CosObj to compare.

Return Value

Returns zero if the two objects are equal, -1 if **obj1** is less than **obj2**, 1 if **obj1** is greater than **obj2**.

Header File

```
CosCalls.h
```

Related Methods

```
CosObjEqual
```

Product

```
reader, base, pro, pdf
```

**Availability**

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



CosObjCopy

```
CosObj CosObjCopy (CosObj srcObj, CosDoc destDoc,  
    ASBool copyIndirect);
```

Description

Copies a **CosObj** from one document to another (or the same document).

Parameters

srcObj	The CosObj to copy.
destDoc	The CosDoc for the document into which the CosObj is copied.
copyIndirect	true if all indirectly referenced objects from srcObj are copied to destDoc , false otherwise.

Return Value

The **CosObj** which has been copied to the destination document.

Header File

CosCalls.h

Related Methods

[CosObjEqual](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

CosObjDestroy

```
void CosObjDestroy (CosObj obj);
```

Description

Destroys a Cos object. This method does nothing if **obj** is a direct scalar object, such as the **null** object.

If a composite object (array, dictionary or stream) is destroyed:

- all the direct objects in it are automatically destroyed
- the indirect objects in it are *not* destroyed

Parameters

obj	The object to destroy.

Return Value

None

Header File

CosCalls.h

Related Methods

[CosNewArray](#)
[CosNewBoolean](#)
[CosNewDict](#)
[CosNewFixed](#)
[CosNewInteger](#)
[CosNewName](#)
[CosNewStream](#)
[CosNewString](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

CosObjEnum

```
ASBool CosObjEnum (CosObj obj, CosObjEnumProc proc,
void* clientData);
```

Description

Enumerates the elements of a Cos object by calling a user-supplied procedure for each component of the object.

Parameters

obj	The object whose elements are enumerated. - For scalars or strings: returns true. - For dictionaries, proc is called for each key-value pair. - For arrays, proc is called for each element. - For streams, proc is called for each key-value pair in the stream dictionary
proc	User-supplied callback to call for each element of obj . Enumeration ends if proc returns false . Array elements are enumerated in ascending order of index. The order in which dictionary key-value pairs are enumerated is undefined. proc (as well as any routine called by proc) must not modify obj . Doing so can produce undefined results or errors. For example, if obj is an array, proc must not call CosArrayRemove ; if obj is a dictionary, proc must not call CosDictPut .
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

Returns the value that **proc** returned (that is, returns **true** if all the elements of the object were enumerated, **false** if enumeration was terminated at the request of **proc**).

Header File

CosCalls.h

Related Methods

[CosArrayGet](#)
[CosDictGet](#)
[CosDocEnumEOFs](#)
[CosDocEnumIndirect](#)



Example

```
static ACCB1 ASBool ACCB2
myCosObjEnumDictProc(CosObj key,
CosObj value, void* clientData)
{
    ...process the key-value pair...
    return true;
}

ASCallback myEnumCB =
    ASCallbackCreateProto(CosObjEnumProc,
        &myCosObjEnumDictProc);
CosObjEnum(cosobjDict, myEnumCB,
    &clientData);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



CosObjEqual

```
ASBool CosObjEqual (CosObj obj1, CosObj obj2);
```

Description

Tests whether two Cos objects are equal. Cos objects are equal when all of the following conditions are true:

- They are either both direct or both indirect.
- They have the same type.
- If they are indirect, they have the same generation number.
- If they are scalars, they have the same value. (Two **null** objects are equal.)
- If they are nonscalar, they reference the same value.

The last condition implies that the comparison is "shallow." For example:

```
CosObj a, b, c;
a = CosNewString (doc, "XYZ");
b = CosNewString (doc, "XYZ");
c = b;
```

In this case, **CosObjEqual (a,b)** is false, but **CosObjEqual (b,c)** is true.

Parameters

obj1, obj2	Objects to compare.

Return Value

true if **obj1** and **obj2** are equal, **false** otherwise.

Header File

CosCalls.h

Related Methods

[CosObjCmp](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

CosObjGetCollection

```
CosObjCollection CosObjGetCollection (CosObj obj);
```

Description

Gets the **CosObjCollection** containing the specified object. If the object is not in a collection, the method raises an exception.

Parameters

obj	The object whose CosObjCollection is obtained.

Return Value

The **CosObjCollection** to which the object belongs.

Known Exceptions

Raises an error if **obj** is not in a collection..

Header File

CosCalls.h

Related Methods

[CosObjAddToCollection](#)
[CosObjIsCompressed](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

CosObjGetCompressibility

```
ASBool CosObjGetCompressibility (CosObj obj);
```

Description

Tests whether an object is compressible. A compressible object can be added to a [CosObjCollection](#). An object is compressible only if all of the following conditions are true:

- It is indirect.
- It has a generation number of zero.
- It is not a stream.
- It has not been marked as incompressible by [CosObjSetCompressibility](#).

Parameters

<code>obj</code>	The object to test.

Return Value

true if `obj` is compressible, **false** otherwise.

Header File

`CosCalls.h`

Related Methods

[CosObjIsCompressed](#)
[CosObjSetCompressibility](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



CosObjGetDoc

```
CosDoc CosObjGetDoc (CosObj obj);
```

Description

Gets the **CosDoc** containing the specified object. This is defined *only* for indirect or non-scalar objects.

Parameters

obj	The object whose CosDoc is obtained.

Return Value

The object's **CosDoc**.

Known Exceptions

Raises **cosErrInvalidObj** if the object is a direct scalar object.

Header File

CosCalls.h

Related Methods

[PDDocGetCosDoc](#)

Example

```
CosDoc myCosDoc = CosObjGetDoc(cosObj);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

CosObjGetGeneration

```
ASUns16 CosObjGetGeneration (CosObj obj);
```

Description

Gets the generation number of an indirect Cos object. See Section 3.2.9 in the [PDF Reference](#) for more information.

Parameters

obj	The indirect CosObj for which the generation number is obtained. A CosObj can be determined to be indirect using CosObjIsIndirect .

Return Value

The generation number of **cosObj**.

Known Exceptions

Raises [cosErrInvalidObj](#) if the object is not valid or is not indirect.

Header File

CosCalls.h

Related Methods

[CosObjGetID](#)
[CosObjIsIndirect](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



CosObjGetID

```
ASInt32 CosObjGetID (CosObj obj);
```

Description

Gets the local master index for an indirect object. For indirect objects, the local master index is the same as the indirect object index that appears in the PDF file.

Parameters

obj	The indirect CosObj for which the ID is obtained. A CosObj can be determined to be indirect using CosObjIsIndirect .

Return Value

The ID of **obj**.

Known Exceptions

Raises **cosErrInvalidObj** if the object is not valid or is not indirect.

Header File

CosCalls.h

Related Methods

[CosDocGetObjByID](#)
[CosObjGetGeneration](#)
[CosObjIsIndirect](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



CosObjGetType

```
CosType CosObjGetType (CosObj obj);
```

Description

Gets an object's type.

Parameters

obj	The object whose type is obtained.

Return Value

The object's type.

Header File

CosCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



CosObjHash

```
ASInt32 CosObjHash (CosObj obj);
```

Description

Gets a 32-bit hash-code for the given **CosObj**.

Two **CosObjects** with equal hash-codes are not necessarily equal, however. Use **CosObjEqual** to determine the equality of Cos objects.

Parameters

obj	The CosObj for which to obtain a hash-code.

Return Value

32-bit hash-code for the given **CosObj**, or **CosNewNull** if there is no object with this ID.

Header File

`CosCalls.h`

Related Methods

[CosObjEqual](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

CosObjIsCompressed

```
ASBool CosObjIsCompressed (CosObj obj);
```

Description

Tests whether an object is compressed—that is, part of a [CosObjCollection](#).

Parameters

<code>obj</code>	The object to test.

Return Value

true if `obj` is compressed, **false** otherwise.

Header File

`CosCalls.h`

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



CosObjsIndirect

```
ASBool CosObjIsIndirect (CosObj obj);
```

Description

Tests whether an object is indirect.

Parameters

obj	The object to test.

Return Value

true if **obj** is indirect, **false** if **obj** is direct.

Header File

CosCalls.h

Example

```
if(CosObjIsIndirect(cosObj)) {  
    /* Process indirect objects only */  
    ...  
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

CosObjRefreshAfterLinearizedSave

```
void CosObjRefreshAfterLinearizedSave (CosObj *obj,
    CosDoc doc);
```

Description

Updates an indirect Cos object after a linearized save operation. Linearizing renumbers all indirect objects; this function returns the new renumbered Cos object, which should be used from this point on. This call is only valid from within notification callbacks responding to the **PDDocDidSave** notification. If called from outside this context, or if the passed Cos object is direct, the function does not modify the object.

Parameters

obj	A pointer to the object to refresh. The object is updated by the method.
doc	The document that was saved.

Return Value

None

Header File

CosCalls.h

Product

reader, base, pro

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

CosObjRemoveFromCollection

```
void CosObjRemoveFromCollection (CosObjCollection coll,  
    CosObj obj);
```

Description

Removes a Cos object from the [CosObjCollection](#) to which it belongs.

Parameters

<code>coll</code>	The Cos object collection.
<code>obj</code>	The object to remove.

Return Value

None

Known Exceptions

Raises an exception if the object is not in the collection.

Header File

`CosCalls.h`

Related Methods

[CosObjAddToCollection](#)
[CosObjGetCompressibility](#)
[CosObjIsCompressed](#)
[CosObjSetCompressibility](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_COS_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

CosObjSetCompressibility

```
void CosObjSetCompressibility (CosObj obj,
                               ASBool compressible);
```

Description

Controls whether a Cos object can be compressed. A compressible object can be added to a [CosObjCollection](#).

If you set the compressibility to **false**, calling [CosObjAddToCollection](#) on that object has no effect. If the object is already compressed, it is removed from the object collection to which it belongs and then marked as incompressible.

This method does nothing if applied to a direct object, a stream, or an object whose generation number is not zero. Objects of these types are never compressible.

Parameters

obj	The object whose compressibility is set.
compressible	true if the object can be made part of a CosObjCollection , false otherwise.

Return Value

true if obj is marked as compressible, **false** otherwise.

Header File

CosCalls.h

Related Methods

- [CosObjAddToCollection](#)
- [CosObjGetCompressibility](#)
- [CosObjIsCompressed](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

CosObjCollection

CosObjCollectionEnum

```
ASBool CosObjCollectionEnum (CosObjCollection coll,
                             CosObjEnumProc proc, void* clientData);
```

Description

Enumerates the members of a Cos object collection, calling a user-supplied procedure for each member object. The order in which the objects are enumerated is undefined.

Parameters

coll	The object collection whose members are enumerated.
proc	User-supplied callback to call for each member object of coll . Enumeration ends if proc returns false . The callback must not modify the collection (by adding or removing objects, for example). Doing so produces undefined results or errors.
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

Returns the value that **proc** returned (that is, returns **true** if all the member objects were enumerated, **false** if enumeration was terminated at the request of **proc**).

Header File

CosCalls.h

Related Methods

[CosObjGetCollection](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



CosObjCollectionEqual

```
ASBool CosObjCollectionEqual (CosObjCollection c1,
                              CosObjCollection c2);
```

Description

Tests whether two Cos object collections are the same collection. Two **null** collections are always equal. (A **null** collection is not associated with a document and cannot store objects; it is generally used only as an initial value for a variable of type [CosObjCollection](#).)

Parameters

c1, c2	Object collections to compare.

Return Value

true if **c1** and **c2** are the same collection, **false** otherwise.

Header File

CosCalls.h

Related Methods

[CosNewObjCollection](#)
[CosObjGetCollection](#)
[CosObjCollectionIsNull](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

CosObjCollectionIsNull

```
ASBool CosObjCollectionIsNull (CosObjCollection coll);
```

Description

Tests whether an object collection is **null**. A **null** collection is not associated with a document and cannot store objects; it is generally used only as an initial value for a variable of type [CosObjCollection](#).

Parameters

<code>coll</code>	The object collection to test.

Return Value

true if `coll` is null, **false** otherwise.

Header File

`CosCalls.h`

Related Methods

[CosNewObjCollection](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

CosObjCollectionSize

```
ASInt32 CosObjCollectionSize (CosObjCollection coll);
```

Description

Gets the number of objects in an object collection. The size of a **null** collection is zero.

Parameters

coll	The object collection whose size is obtained.

Return Value

The number of objects in the collection.

Header File

CosCalls.h

Related Methods

[CosObjAddToCollection](#)
[CosObjRemoveFromCollection](#)
[CosObjCollectionEnum](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

CosNewObjCollection

```
CosObjCollection CosNewObjCollection (CosDoc dP);
```

Description

Creates a new object collection for objects in a document.

Parameters

dP	The document whose objects are collected, or NULL to create a null collection. (A null collection is not associated with a document and cannot store objects; it is generally used only as an initial value for a variable of type CosObjCollection .)

Return Value

The newly created Cos object collection.

Header File

CosCalls.h

Related Methods

[CosObjAddToCollection](#)
[CosObjCollectionEnum](#)
[CosObjGetCollection](#)
[CosObjCollectionIsNull](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher

CosStream

CosNewStream

```
CosObj CosNewStream (CosDoc dP, ASBool indirect, ASStm source,
    ASInt32 sourceStart, ASBool encodeTheSourceData,
    CosObj attributesDict, CosObj encodeParms,
    ASInt32 sourceLength);
```

Description

Creates a new Cos stream, using data from an existing **ASStm**. The data is copied, so the source stream may be closed after **CosNewStream** returns.

This method creates a Cos stream object by writing its PDF representation to an intermediate file, in the format specified by [PDF Reference](#):

```
<</Length ... /Filter ... /DecodeParms ...>>
stream
... data, possibly encoded ...
endstream
```

This occurs in four steps:

1. Writing the attribute dictionary.

If **attributesDict** is a valid Cos dictionary, the method writes that dictionary to the intermediate file. Otherwise, it creates a new direct dictionary, determining a **Length** key according to the **sourceLength** value:

- If **sourceLength** is negative, or if the source data is to be encoded (see below), the value of the **Length** key is a reference to a new indirect object, whose value will be set in Step 4.
- Otherwise, **Length** is a direct scalar representing **sourceLength**.

The dictionary that is written becomes the new stream's attribute dictionary.

2. Reading the data.

sourceStart determines where in the source stream to begin reading, and whether the source is seekable.

- If **sourceStart** is a negative number, the source is assumed to be non-seekable but positioned at the point where reading should start.
- Otherwise, the source is assumed to be seekable, and reading starts at the position indicated by **sourceStart**. If **sourceStart** is zero, data is read from the beginning of the source stream. Positive values for **sourceStart** may be used, for instance, to skip over initial data in the stream.



3. Encoding the data

- If **attributesDict** is a valid Cos dictionary, *and* it contains a **Filter** key, *and* **encodeTheSourceData** is **true**, the method encodes the data after reading it from the source stream and before writing it to the intermediate file.
The **attributesDict** is used as the new stream’s dictionary. The **Filter** entry in this dictionary indicates how the data in the resulting Cos stream object will be subsequently decoded; the value may be the name of a decoding filter or an array of such names. Specify multiple filters in the order they should be applied to decode the data. (If parameters are needed to decode the data, they are specified as the value of the **DecodeParms** key in **attributesDict**. See the [PDF Reference](#) for details.) For each decoding filter, there is a corresponding encoding filter, which the method applies to the source data during this step.
If parameters are needed to encode the data, they must be specified in the call by **encodeParms**. (Often, the encoding parameters are different from the decoding parameters.) The **encodeParms** parameter is optional for all encoding filters except **DCTDecode** and **JBIG2Decode**. See the **encodeParms** field of [PDEFilterSpec](#).
If an array of filters is supplied, and at least one of them requires encoding parameters, then a corresponding array of encoding parameters is also required. Use the null object to represent default parameters for filters that have defaults.
- In any other case, the method copies the source data directly into the Cos stream with no encoding. If **sourceLength** is negative, it reads bytes until the source reaches its EOF. Otherwise, **sourceLength** indicates how many bytes to read from the source, and an exception is raised if the source reaches EOF before that.

4. Writing the data

After the data is written, if the value of the **Length** key in the attributes dictionary was an indirect reference (either because it was supplied that way in **attributesDict**, or because it was created that way in Step 1) the value of that indirect object is set to the number of bytes actually written (that is, the encoded length if the data was encoded). An indirect **Length** key is useful for one-pass writing, when the size of the written data is not known in advance, either because the data was to be encoded, or because there was no way to know how much data there would be before the source reached its EOF.

- NOTE:** **CosNewStream** sets the document **PDDocNeedsSave** flag (see [PDDocFlags](#)).
- NOTE:** You cannot call [CosStreamPos](#) on a stream created with **CosNewStream** until the file has been saved.

Parameters

dP	The Cos document in which the newly created stream will be used.

indirect	Must always be true , specifying that the Cos stream is created as an indirect object (all streams are indirect). This also sets the document's PDDocNeedsSave flag (see PDDocFlags).
stm	The source stream containing the data to copy into the new stream. The caller is responsible for closing stm after CosNewStream returns. The source stream can be any readable ASStm. Typical sources are: • From files (ASFileStmRdOpen) or from memory (ASMemStmRdOpen). These streams are always seekable. • From arbitrary procedures (ASProcStmRdOpen or ASProcStmRdOpenEx), or from other Cos streams (CosStreamOpenStm). These streams are always non-seekable.
sourceStart	The byte offset into stm from which data reading starts for a seekable stream. If the value is negative, specifies that the stream is not seekable.
encodeTheSourceData	Whether the data in stm should be encoded using filters specified in attributesDict before it is written to the Cos stream. See the description of the encoding step above. If attributesDict is a null object or if the dictionary has no Filter key, this value is ignored.
attributesDict	Either the null Cos object, or a direct Cos dictionary containing stream attributes, such as the length of the Cos stream data and a list of decoding filters and parameters to apply to the data, as defined in Section 3.2.7 in the PDF Reference. See the encoding step in the description above.
encodeParms	The parameters to be used by the filters if the source data is encoded before it is written to the file. The parameters follow the structure for the value of the DecodeParms stream attribute described in Table 3.4 in the PDF Reference. See the encoding step in the description above. If no encoding parameters are needed, this value is ignored.
sourceLength	The amount of data to be read from the source. If negative (typically -1), data is read from the source until it reaches its EOF. See Step 1 in the description above.



Return Value

The newly created stream Cos object.

Known Exceptions

Raises an exception if **attributesDict** is neither the **null** object nor a direct Cos dictionary.

Raises an exception if **sourceStart** is nonnegative but the source is not seekable.

Raises an exception if **sourceLength** is nonnegative but the source stream reaches EOF before that many bytes have been read.

Header File

`CosCalls.h`

Related Methods

[CosObjDestroy](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



CosStreamDict

```
CosObj CosStreamDict (CosObj stream);
```

Description

Gets a stream's attributes dictionary.

Parameters

stream	The stream whose attributes dictionary is obtained.

Return Value

The stream's attributes dictionary Cos object.

Header File

CosCalls.h

Related Methods

[CosStreamLength](#)
[CosStreamPos](#)
[CosDictGet](#)
[CosDictPut](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

CosStreamLength

```
ASTArraySize CosStreamLength (CosObj stream);
```

Description

Gets the length of a Cos stream from the **Length** key in the stream's attributes dictionary. This specifies the length of the *undecoded* data; that is, the number of bytes in the stream before the **Filter** (if any) is applied.

Parameters

stream	The stream whose length is obtained.

Return Value

The length of the stream.

Known Exceptions

Raises an exception if the **Length** key is not found in the attributes dictionary or its value is not an integer.

Header File

CosCalls.h

Related Methods

[CosStreamDict](#)
[CosStreamPos](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

CosStreamOpenStm

```
ASStm CosStreamOpenStm (CosObj stream,
CosStreamOpenMode mode) ;
```

Description

Creates a new, non-seekable **ASStm** for reading data from a Cos stream. The data in the Cos stream may be filtered and encrypted. After opening the Cos stream, data can be read from it into memory using **ASStmRead**. When reading is completed, close the stream using **ASStmClose**.

Parameters

stream	The Cos stream object for which an ASStm is opened.
mode	Must be one of the CosStreamOpenMode values.

Return Value

The newly-opened **ASStm**.

Header File

CosCalls.h

Related Methods

ASStmRead
ASStmWrite
CosNewStream

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

CosStreamPos

ASTCount CosStreamPos (**CosObj** stream);

Description

Gets the byte offset of the start of a Cos stream's data in the PDF file (that is, the byte offset of the beginning of the line following the "stream" token). Use this method to obtain the file location of any private data in a stream that you need to read directly rather than letting it pass through the normal Cos mechanisms. For example, a QuickTime video embedded in a PDF file.

CosStreamPos is only valid when called on a stream that is already stored in a PDF document. If the stream was created using **CosNewStream**, the new stream is stored in the document's temp file, and you *cannot* invoke **CosStreamPos** on it. After the file has been saved, you can use **CosStreamPos** on the stream.

Parameters

stream	The stream whose current position is obtained.

Return Value

The byte offset of the start of the Cos stream's data in the PDF file.

Known Exceptions

Raises **cosErrInvalidObj** if the stream object has not yet been saved to the PDF file. In other words, before you can call **CosStreamPos** on a newly created stream, you must first save the PDF file.

Header File

CosCalls.h

Related Methods

CosStreamDict
CosStreamLength

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

CosString

CosCopyStringValue

```
char* CosCopyStringValue (CosObj obj, ASTCount* nBytes);
```

Description

Returns a newly allocated buffer containing a copy of the Cos object’s string value. Upon return **nBytes** contains the number of bytes in the original Cos string. **CosCopyStringValue** never returns **NULL**; it raises an exception if the allocation fails. The client is responsible for freeing the result by calling [ASfree](#).

CosCopyStringValue allocates extra memory past the end of the string and writes zeros into these extra bytes to ensure that the string is null-terminated whether viewed as a UTF-16 (Unicode) string or as a C string. (These bytes are not included in the number returned in **nBytes**). If the Cos string has 0 length, **nBytes** will be 0, and a pointer to newly allocated memory containing some zero bytes is returned (that is, **CosCopyStringValue** still returns a null-terminated string but with zero length).

NOTE: In general, the returned value is not a null-terminated C string. Cos string objects are binary and can contain arbitrary byte sequences, including **NULL** characters. Standard C string-handling functions may not work as expected.

Parameters

obj	The Cos object whose string value is copied and returned.
nBytes	<i>(Filled by the method)</i> The length of the original Cos string in bytes. Can be NULL if you don't care how many bytes were in the original string.

Return Value

A copy of the Cos object’s string value.

Known Exceptions

Raises an out-of-memory exception if insufficient memory is available. Can also raise any exception that **CosStringValue** can raise.

Header File

CosCalls.h

Related Methods

[CosStringValueSafe](#)

**Product**

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

CosNewString

```
CosObj CosNewString (CosDoc dP, ASBool indirect, char* str,
                     ASArraySize nBytes);
```

Description

Creates and returns a new Cos string object.

Parameters

dP	The document in which the string is used.
indirect	If true , creates the string as an indirect object, and sets the document dP 's PDDocNeedsSave flag (see PDDocFlags). If false , creates the string as a direct object.
str	The value that the new string will have. It is <i>not</i> a C string, since Cos strings can contain NULL characters. The data in str is copied, that is, if str was dynamically allocated, it can be freed after this call.
nBytes	The length of str .

Return Value

The newly created string Cos object.

Header File

CosCalls.h

Related Methods

[CosStringValue](#)
[CosObjDestroy](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

CosStringGetHexFlag

```
ASBool CosStringGetHexFlag (CosObj cosObj);
```

Description

Gets the hex flag of the **CosString**. The hex flag specifies whether the **CosString** should be written out as hex when writing the Cos Object to file.

Parameters

<code>cosObj</code>	The CosString for which the hex flag is obtained.

Return Value

The current value of the flag.

Known Exceptions

[cosErrExpectedString](#)

Header File

`CosCalls.h`

Related Methods

[CosStringSetHexFlag](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

CosStringSetHexFlag

```
ASBool CosStringSetHexFlag (CosObj cosObj, ASBool setHex);
```

Description

Sets the hex flag of the **CosString**. The hex flag specifies whether the **CosString** should be written out as hex when writing the Cos Object to file.

Parameters

cosObj	The CosString for which the hex flag is set.
setHex	The value to set for the flag.

Return Value

The value of **setHex**.

Known Exceptions

[cosErrExpectedString](#)

Header File

CosCalls.h

Related Methods

[CosStringGetHexFlag](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

CosStringValue

```
char* CosStringValue (CosObj obj, ASTCount* nBytes);
```

Description

Gets the value of string Cos object, and the string’s length.

NOTE: The pointer returned from this method is not guaranteed to remain valid if **CosStringValue** is called again. It is recommended that you use **CosStringValueSafe** or **CosCopyStringValue** instead; these methods place the string into a user-allocated buffer.

NOTE: The caller *must* immediately copy the returned string. The memory pointed to be the return value may become invalid if *any* memory-allocating calls are made. In particular, consider the sequence:

```
str1 = CosStringValue(...);  
str2 = CosStringValue(...);
```

In this case, the contents of **str1** may be invalid by the time the second **CosStringValue** call returns.

NOTE: The returned value is *not* a C-style string. Cos string objects can contain **NULL** bytes. Standard C string-handling functions may not work as expected.

Parameters

obj	The object whose value is obtained.
nBytes	(Filled by the method) The length of the string, in bytes. Must be a non- NULL pointer.

Return Value

The value of **obj**.

Known Exceptions

Raises an exception if the type of **obj** is not **CosString**.

Header File

CosCalls.h

Related Methods

[CosNewString](#)
[CosCopyStringValue](#)

**Example**

```
ASInt32 len;  
char* p;  
  
p = CosStringValue(strCos, &len);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

CosStringValueSafe

```
char* CosStringValueSafe(CosObj obj, char* buffer,
    AStArraySize bufferSize, AStCount *nBytes);
```

Description

Copies at most **bufferLen** bytes from **obj**'s string value into **buffer**, and stores the actual length of the Cos string in ***nBytes**. If **bufferLen** is greater than the length of the Cos string, the remaining bytes in **buffer** have undefined values upon return.

NOTE: In general, the returned value is not a null-terminated C string. Cos string objects are binary data and can contain any arbitrary byte sequence, including embedded **NULL** characters. Standard C string-handling functions may not work as expected.

Parameters

obj	The Cos object whose string value is copied.
buffer	The buffer into which the Cos string value is copied or NULL .
bufferLen	The length of buffer or 0.
nBytes	<i>(Filled by the method)</i> The length of the original Cos string in bytes (which may be more than bufferLen). Must be a non- NULL pointer.

Return Value

A copy of the Cos string value or an exception. Will never return **NULL**.

Known Exceptions

Will raise a bad-parameter exception if **bufferLen** is less than 0, or **nBytes** is **NULL**. Can also raise any exception that **CosStringValue** can raise.

Header File

CosCalls.h

Related Methods

[CosCopyStringValue](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



Encryption/Decryption

CosCryptGetVersion

```
ASTVersion CosCryptGetVersion ();
```

Description

Gets the current version number of the encryption algorithm supported.

Return Value

The current version number of encryption supported.

Header File

CosCalls.h

Related Methods

[CosDecryptGetMaxKeyBytes](#)

[CosEncryptGetMaxKeyBytes](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00040005** or higher.



CosDecryptData

```
void CosDecryptData (void* src, ASTArraySize len, void* dst,
    char* cryptData, ASTArraySize cryptDataLen);
```

Description

Decrypts data in a buffer using the specified encryption key. The standard Acrobat viewer encryption/decryption algorithm (RC4 from RSA Data Security, Inc.) is used.

Parameters

src	The buffer containing the data to decrypt.
len	The number of bytes in src .
dst	<i>(Filled by the method)</i> The buffer into which the decrypted data will be placed. This may point to the same location as src .
cryptdata	The encryption key.
cryptDataLen	Length of the encryption key, in bytes. It cannot be greater than 5.

Return Value

None

Known Exceptions

Raises an exception if encryption encounters an internal error.

Header File

CosCalls.h

Related Methods

[CosEncryptData](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

CosDecryptGetMaxKeyBytes

```
CosByteMax CosDecryptGetMaxKeyBytes (ASTVersion cryptVersion);
```

Description

Gets the maximum number of the decryption key length, in bytes, for the specified **cryptVersion**.

Parameters

cryptVersion	The Cos crypt version—the version of the algorithm that is used to encrypt and decrypt document data. cryptVersion equal to 0 is treated as cryptVersion equal to 1 to maintain backward compatibility.

Return Value

The maximum number of key length, in bytes, for the specified **cryptVersion**. If **cryptVersion** is not currently supported, returns -1.

Header File

CosCalls.h

Related Methods

[CosCryptGetVersion](#)
[CosEncryptGetMaxKeyBytes](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00040005** or higher.

CosEncryptData

```
void CosEncryptData (void* src, ASTArraySize len, void* dst,
    char* cryptData, ASTArraySize cryptDataLen);
```

Description

Encrypts data in a buffer using the specified encryption key. The standard Acrobat viewer encryption/decryption algorithm (RC4 from RSA Data Security, Inc.) is used.

Parameters

src	The buffer containing the data to encrypt.
len	The number of bytes in src .
dst	<i>(Filled by the method)</i> The buffer into which the encrypted data will be placed. This may point to the same location as src .
cryptdata	The encryption key.
cryptDataLen	Length of the encryption key, in bytes. It cannot be greater than 5.

Return Value

None

Known Exceptions

Raises an exception if encryption encounters an internal error.

Header File

CosCalls.h

Related Methods

[CosDecryptData](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

CosEncryptGetMaxKeyBytes

```
CosByteMax CosEncryptGetMaxKeyBytes (ASTVersion cryptVersion);
```

Description

Gets the maximum number of the encryption key length, in bytes, for the specified **cryptVersion**.

Parameters

cryptVersion	The Cos crypt version—the version of the algorithm that is used to encrypt and decrypt document data. cryptVersion equal to 0 is treated as cryptVersion equal to 1 to maintain backward compatibility.

Return Value

The maximum number of key length, in bytes, for the specified **cryptVersion**. If **cryptVersion** is not currently supported, it returns -1.

Header File

CosCalls.h

Related Methods

[CosCryptGetVersion](#)
[CosDecryptGetMaxKeyBytes](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_COS_VERSION** (in **PIRequir.h**) is set to **0x00040005** or higher.



PD Layer Methods

General

PDAction

PDAnnot

PDAnnotHandler

PDBead

PDBookmark

PDCharProc

PDDoc

PDFFileSpec

PDFont

PDForm

PDGraphic

PDImage

PDInlineImage

PDLinkAnnot

PDNameTree

PDNumTree

PDOCConfig

PDOCContext

PDOCG

PDOCMD

PDPage

PDPageLabel

PDPath

PDStyle

PDText

PDTextAnnot

PDTextSelect

PDThread

PDTrans



PDViewDest
PDWord
PDWordFinder
PDXObject



General

PDApplyFunction

```
void PDApplyFunction (CosObj funcDict, const float inVals[],  
float outVals[]);
```

Description

Given a **CosObj** that represents a function, applies the function to the supplied values.

Parameters

funcDict	CosObj representing a function.
inVals	Input values.
outVals	Output values.

Return Value

None

Known Exceptions

Raises **genErrBadParm** if the **CosObj** is not a function dictionary.

Raises an error if the **CosObj** is malformed.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDDrawCosObjToWindow

```
void PDDrawCosObjToWindow (CosObj cosObj, void* window,
void* displayContext, ASBool isDPS, ASFixedMatrix* matrix,
ASFixedRect* updateRect, CancelProc cancelProc,
void* cancelProcClientData);
```

Description

Draws the specified stream of PDF marking operators into the specified window. This method is used for platform-independent drawing of graphics and text.

This method changes the current page's CTM and zoom factor, and dirties the file.

Parameters

cosObj	The stream Cos object to draw into window. This stream can be created using CosNewStream. The stream's dictionary must contain a Resources key whose value is a dictionary specifying all the resources needed to draw the Cos object (including a ProcSet entry). Its structure and contents are the same as for the Resources dictionary for a Page object. See Section 3.7.2 in the PDF Reference for a description of a Page object's Resources dictionary. The stream's data is a sequence of PDF marking operators. See Appendix A in the PDF Reference for a description of these operators. A pseudocode example of the stream object is: <pre><< /Length 1000 /Filter [...filters...] /Resources << /ProcSet [/PDF /Text] /Font <</F5 6 0 R /F9 12 0 R>> >> >> stream ...stream data... endstream</pre>
window	Pointer to a platform-dependent window object (WindowPtr or CWindowPtr in Mac OS, HWND in Windows). In Mac OS, to draw into an off screen GWorld, pass NULL in window and pass the GWorldPtr in displayContext. In Windows, to draw into an offscreen DC, pass NULL for window.

displayContext	Pointer to a platform-dependent display context structure (GWorldPtr in Mac OS, hDC in Windows). In Mac OS, displayContext is ignored if window is non- NULL .
isDps	Currently unused. Always set to false .
matrix	Pointer to a matrix to concatenate onto the default page matrix. It is useful for scaling and for converting from page to window coordinates.
updateRect	Pointer to a rectangle, specified in user space coordinates. Any objects outside of updateRect will not be drawn. All objects are drawn if updateRect is NULL .
cancelProc	Procedure called periodically to check for user cancel of the drawing operation. The default cancel proc can be obtained using AVAppGetCancelProc . May be NULL , in which case no cancel proc is used.
cancelProcClientData	Pointer to user-supplied data to pass to cancelProc each time it is called. Should be NULL if cancelProc is NULL .

Return Value

None

Header File

PDCalls.h

Related Methods[CosNewStream](#)[PDPageDrawContentsToWindow](#)[PDPageDrawContentsToWindowEx](#)**Product**

reader, base, pro, pdf

AvailabilityAvailable if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020001** or higher.

PDDrawCosObjWithParams

```
void PDDrawCosObjWithParams (CosObj cosObj,
PDDrawParams params)
```

Description

Provides control over the rendering of contents, including both those parameters you would pass to [PDDrawCosObjWithParams](#), and an optional-content context that determines which contents are visible.

Parameters

cosObj	The object to draw.
params	The parameters with which to draw the object, including the optional-content context to use for content visibility.

Return Value

None

Known Exceptions

[pdPErrUnableToCreateRasterPort](#)

Header File

PDCalls.h

Related Methods

[PDDrawCosObjToWindow](#)
[PDPageEnumContents](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDEnumDocs

```
void PDEnumDocs (PDDocEnumProc proc, void* clientData);
```

Description

Enumerates the **PDDocs** that are currently open, calling a user-supplied procedure for each open document.

Parameters

proc	User-supplied callback to call for each open PDDoc . Enumeration halts if proc returns false .
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Header File

PDCalls.h

Related Methods

[AVAppEnumDocs](#)

[PDDocOpen](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDGetHostEncoding

```
void* PDGetHostEncoding (void);
```

Description

Indicates what kind of host encoding a system uses. Allows you to determine whether a system is Roman or non-Roman. (Non-Roman is also known as CJK-capable, that is, capable of handling multi-byte character sets, such as Chinese, Japanese, or Korean.)

Host encoding is a platform-dependent encoding for the host machine. For non-UNIX Roman systems, it is **MacRomanEncoding** in Mac OS and **WinAnsiEncoding** in Windows. In UNIX (except HP-UX) Roman systems, it is ISO8859-1 (ISO Latin-1); for HP-UX, it is **HP-ROMAN8**. See Appendix D in the [PDF Reference](#) for descriptions of **MacRomanEncoding**, **WinAnsiEncoding**, and **PDFDocEncoding**.

For non-Roman systems, the host encoding may be a variety of encodings, which are defined by a **CMap** (character map). See Section 5.6.4 in the [PDF Reference](#) for a list of predefined **CMaps**.

Use [PDGetHostEncoding](#) to determine if a system's host encoding is Roman or not.

Parameters

None

Return Value

0 for a Roman system; nonzero for a non-Roman system (a structure that depends on the host encoding). Users should simply test whether this value is 0 or not.

Header File

PDCalls.h

Related Methods

[PDGetPDFDocEncoding](#)
[AVAppGetLanguageEncoding](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020003** or higher.

PDGetPDFDocEncoding

```
ASUns8** PDGetPDFDocEncoding (void);
```

Description

Gets an array describing the differences between the platform's host encoding and **PDFDocEncoding**.

Host encoding is a platform-dependent encoding for the host machine. For non-UNIX Roman systems, it is **MacRomanEncoding** in Mac OS and **WinAnsiEncoding** in Windows. In UNIX (except HP-UX) Roman systems, it is ISO8859-1 (ISO Latin-1); for HP-UX, it is **HP-ROMAN8**. See Appendix D in the [PDF Reference](#) for descriptions of **MacRomanEncoding**, **WinAnsiEncoding**, and **PDFDocEncoding**.

For non-Roman systems, the host encoding may be a variety of encodings, which are defined by a **CMap** (character map). See Section 5.6.4 in the [PDF Reference](#) for a list of predefined **CMaps**.

Use [PDGetHostEncoding](#) to determine if a system's host encoding is Roman or not.

Parameters

None

Return Value

An array containing 256 elements. Each element corresponds to a code point in the **PDFDocEncoding**, and is either **NULL** or a pointer to a string.

If the element is **NULL**, the code point refers to the same glyph in both host encoding and **PDFDocEncoding**. If the element is non-**NULL**, it points to a string containing the glyph name for the code point.

Header File

PDCalls.h

Related Methods

[PDGetHostEncoding](#)
[AVAppGetLanguageEncoding](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDHostMBLen

ASInt32 PDHostMBLen (const char* cp);

Description

Gets the number of additional bytes required for the multi-byte character pointed to by **cp**. If **cp** points to a single-byte character, 0 is returned. This method allows determining the length of multi-byte character strings to allocate space for them.

This function is similar to the ANSI-C code:

```
mblen(cp, MB_LEN_MAX) - 1
```

or the Windows function:

```
IsDBCSLeadByte(cp)
```

Parameters

cp	The character to examine.

Return Value

The number of bytes in the multi-byte character.

Header File

PDCalls.h

Example

```
char* strchrMB(char* cp, char* ch) {
    ASInt32 ch_len = PDHostMBLen(ch);

    while (*cp != '\0') {
        ASInt32 cp_len = PDHostMBLen(cp);

        if (ch_len == cp_len) {
            ASInt32 i;

            for (i = 0; i < ch_len; i++)
                if (cp[i] != ch[i])
                    break;

            if (i == ch_len)
                return cp;
        }

        cp += (cp_len + 1);
    }

    return NULL;
}
```


**Product**

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020003** or higher.



PDPrefGetColorCal

```
ASBool PDPrefGetColorCal (PDColorCalP colorCal);
```

Description

Gets the values to use for displaying calibrated color and gray scale. These values are the chromaticity and gammas of the phosphors in the monitor.

These values are used for rendering if calibrated color is enabled by the preferences file item `avpDoCalibratedColor` (see [AVAppGetPreference](#)).

Parameters

colorCal	<i>(Filled by the method)</i> Pointer to a structure that contains the color calibration information. For RGB devices, the red, green, and blue chromaticity and gammas are used; for gray scale, <code>whiteChrom</code> and <code>greenGamma</code> are used. You must allocate storage for the <code>colorCal</code> data structure and pass a pointer to the memory you allocated.

Return Value

Always returns `true`.

Header File

PDCalls.h

Related Methods

[PDPrefSetColorCal](#)
[AVAppGetPreference](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.



PDPrefSetColorCal

```
ASBool PDPrefSetColorCal (PDColorCalP colorCal);
```

Description

Sets the values to use for displaying calibrated color and gray scale. These values are the chromaticities and gammas of the phosphors in the monitor.

These values do not necessarily correspond to the monitor being used; it is the responsibility of the client that sets these values to provide the correct values.

These values are used for rendering if calibrated color is enabled by the preferences file item **avpDoCalibratedColor** (see [AVAppSetPreference](#)).

Parameters

colorCal	Pointer to a structure that contains the color calibration information. For RGB devices, the red, green, and blue chromaticities and gammas are used; for gray scale, whiteChrom and greenGamma are used.

Return Value

Returns **true** if the values were successfully set and installed for the currently active display device, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDPrefGetColorCal](#)
[AVAppSetPreference](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDRegisterCryptHandler

```
void PDRegisterCryptHandler (PDCryptHandler handler,
    const char* pdfName, const char* userName);
```

Description

Registers a new security handler with the Acrobat viewer.

Parameters

handler	Pointer to a structure that contains the security handler's callback functions. This structure <i>must not</i> be freed after calling PDRegisterCryptHandler .
pdfName	The name of the security handler as it will appear in the PDF file. This name is also used by PDDocSetNewCryptHandler . Storage for this name can be freed after PDRegisterCryptHandler has been called.
userName	The name of the security handler as it will be shown in menus. This name can be localized into different languages. Storage for this name can be freed after PDRegisterCryptHandler has been called.

Return Value

None

Known Exceptions

Raises **genErrBadParm** if the security handler's **size** field is incorrect.

Raises **ErrSysPDSEdit** if either **pdfName** or **userName** are already in use by a registered security handler.

Raises **genErrNoMemory** if memory is exhausted.

May raise other exceptions.

Header File

PDCalls.h

Related Methods

[PDDocGetNewCryptHandler](#)
[PDDocPermRequest](#)
[PDDocSetNewCryptHandler](#)
[PDDocSetNewCryptHandlerEx](#)
[PDRegisterCryptHandlerEx](#)

Example

```

PDCryptHandler handler;

handler = (PDCryptHandler) ASmallloc(
    sizeof(PDCryptHandlerRec));
handler->size = sizeof(PDCryptHandlerRec);
handler->NewAuthData = 0;
handler->GetAuthData = 0;
handler->Authorize =
    ASCallbackCreateProto(
        PDCryptAuthorizeProc, &Authorize);
handler->NewSecurityData =
    ASCallbackCreateProto(
        PDCryptNewSecurityDataProc,
        &NewSecurityData);
handler->ValidateSecurityData =
    ASCallbackCreateProto(
        PDCryptValidateSecurityDataProc,
        &ValidateSecurityData);
handler->UpdateSecurityData =
    ASCallbackCreateProto(
        PDCryptUpdateSecurityDataProc,
        &UpdateSecurityData);
handler->NewCryptData =
    ASCallbackCreateProto(
        PDCryptNewCryptDataProc, &NewCryptData);
handler->FillEncryptDict =
    ASCallbackCreateProto(
        PDCryptFillEncryptDictProc,
        &FillEncryptDict);
handler->GetSecurityInfo =
    ASCallbackCreateProto(
        PDCryptGetSecurityInfoProc,
        &GetSecurityInfo);
DURING
    PDRegisterCryptHandler(handler,
        "Subscription", "Subscript");
HANDLER
    ...
END_HANDLER

```

Product

reader, base, pro, pdfi

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDRegisterCryptHandlerEx

```
void PDRegisterCryptHandlerEx (PDCryptHandler handler,
    const char* pdfName, const char* userName, void* clientData);
```

Description

Registers a new security handler with the Acrobat viewer. Same as [PDRegisterCryptHandler](#) except that it accepts a client data parameter.

Parameters

handler	Pointer to a structure that contains the security handler's callback functions. This structure must <i>not</i> be freed after calling PDRegisterCryptHandlerEx .
pdfName	The name of the security handler as it will appear in the PDF file. This name is also used by PDDocSetNewCryptHandler . Storage for this name can be freed after PDRegisterCryptHandlerEx has been called.
userName	The name of the security handler as it will be shown in menus. This name can be localized to different languages. Storage for this name can be freed after PDRegisterCryptHandlerEx has been called.
clientData	Pointer to user-supplied data to store with the handler.

Return Value

None

Known Exceptions

Raises [genErrBadParm](#) if the security handler's **size** field is incorrect.

Raises [ErrSysPDSEdit](#) if either **pdfName** or **userName** are already in use by a registered security handler.

Raises [genErrNoMemory](#) if memory is exhausted.

May raise other exceptions.

Header File

PDCalls.h

Related Methods

[PDDocSetNewCryptHandler](#)
[PDDocPermRequest](#)
[PDRegisterCryptHandler](#)

Example

```

PDCryptHandler handler;

handler = (PDCryptHandler) ASmallloc(
    sizeof(PDCryptHandlerRec));
handler->size = sizeof(PDCryptHandlerRec);
handler->NewAuthData = 0;
handler->GetAuthData = 0;
handler->Authorize =
ASCallbackCreateProto(
    PDCryptAuthorizeProc, &Authorize);
handler->NewSecurityData =
ASCallbackCreateProto(
    PDCryptNewSecurityDataProc,
    &NewSecurityData);
handler->ValidateSecurityData =
ASCallbackCreateProto(
    PDCryptValidateSecurityDataProc,
    &ValidateSecurityData);
handler->UpdateSecurityData =
ASCallbackCreateProto(
    PDCryptUpdateSecurityDataProc,
    &UpdateSecurityData);
handler->NewCryptData =
ASCallbackCreateProto(
    PDCryptNewCryptDataProc, &NewCryptData);
handler->FillEncryptDict =
ASCallbackCreateProto(
    PDCryptFillEncryptDictProc,
    &FillEncryptDict);
handler->GetSecurityInfo =
ASCallbackCreateProto(
    PDCryptGetSecurityInfoProc,
    &GetSecurityInfo);
DURING
    PDRegisterCryptHandler(handler,
        "Subscription", "Subscript");
HANDLER
    ...
END_HANDLER

```

Product

reader, base, pro, pdfi

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

PDRegisterFileSpecHandler

```
void PDRegisterFileSpecHandler (ASFileSys contextFileSys,
                                PDFileSpecHandler fileSpecHandler, void* fileSpecHandlerObj);
```

Description

Registers a new file specification handler with the Acrobat viewer. In version 3.0 and later of the Acrobat viewer, use the [PDRegisterFileSpecHandlerByName](#) method instead.

Parameters

contextFileSys	The file system that specifies the context in which the file specification handler is used. This is the file system on which the PDF document resides. It is sometimes necessary to use different file specification handlers depending on the file system in which the document is open. For example, when a document is opened in a Web browser, the Acrobat viewer may use the browser's http stack when it needs to use http. When a document is opened outside of the browser, however, the Acrobat viewer must use a different http stack.
fileSpecHandler	Pointer to a structure that contains the handler's callbacks. This structure <i>must</i> not be freed after calling PDRegisterFileSpecHandler .
fileSpecHandlerObj	Pointer to user-supplied data to pass to the file specification handler's callbacks each time they are called.

Return Value

None

Known Exceptions

[genErrNoMemory](#)

Header File

PDCalls.h

Related Methods

[PDRegisterFileSpecHandlerByName](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDRegisterFileSpecHandlerByName

```
void PDRegisterFileSpecHandlerByName (ASAtom specSysName,
ASFileSys contextFileSys, PDFFileSpecHandler fileSpecHandler,
void* fileSpecHandlerObj);
```

Description

Registers a new file specification handler with the Acrobat viewer. The viewer calls the appropriate file specification handler when it encounters a file specification in a PDF file. The appropriate file specification handler is the one whose:

- **specSysName** matches the value of the **FS** key in the file specification,
- **contextFileSys** matches the file system on which the PDF file resides.

The file specification handler's file system, (passed as the **fileSys** field of **fileSpecHandler**), is used to obtain data from, or write data to, the file referred to by the file specification.

Parameters

specSysName	The name (as an ASAtom) of a file system with which this file specification works.
contextFileSys	The file system that specifies the context in which the file specification handler is used. This is the file system on which the PDF document resides. It is sometimes necessary to use different file specification handlers depending on the file system in which the document is open. For example, when a document is opened in a Web browser, the Acrobat viewer may use the browser's http stack when it needs to use http. When a document is opened outside of the browser, however, the Acrobat viewer must use a different http stack.
fileSpecHandler	Pointer to a structure that contains the handler's callbacks. This structure <i>must</i> not be freed after calling PDRegisterFileSpecHandlerByName .
fileSpecHandlerObj	Pointer to user-supplied data to pass to the file specification handler's callbacks each time they are called.

Return Value

None

Known Exceptions

[genErrNoMemory](#)

**Header File**

PDCalls.h

Related Methods

[PDRegisterFileSpecHandler](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

PDXlateToHost

```
void PDXlateToHost (char* in, char* out, ASInt32 numBytes);
```

Description

Translates a string from **PDFDocEncoding** to host encoding. This method is useful when setting or retrieving displayed text that *must* be in **PDFDocEncoding** (or Unicode), such as text that appears in a text annotation or bookmark.

A character that cannot be converted to the destination encoding is replaced with a space.

Host encoding is a platform-dependent encoding for the host machine. For non-UNIX Roman systems, it is **MacRomanEncoding** in Mac OS and **WinAnsiEncoding** in Windows. In UNIX (except HP-UX) Roman systems, it is ISO8859-1 (ISO Latin-1); for HP-UX, it is **HP-ROMAN8**. See Appendix D in the [PDF Reference](#) for descriptions of **MacRomanEncoding**, **WinAnsiEncoding**, and **PDFDocEncoding**.

For non-Roman systems, the host encoding may be a variety of encodings, which are defined by a **CMap** (character map). See Section 5.6.4 in the [PDF Reference](#) for a list of predefined **CMaps**.

Use [PDGetHostEncoding](#) to determine if a system's host encoding is Roman or not. For non-Roman systems, use [PDXlateToHostEx](#).

In general, [PDXlateToHostEx](#) can be called instead of **PDXlateToHost** since [PDXlateToHostEx](#) works for any host encoding.

Parameters

in	The string to translate (may point to the same memory as out , allowing strings to translate in place).
out	<i>(Filled by the method)</i> The translated string (may point to the same memory as in).
numBytes	Number of bytes in the string to translate.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDGetHostEncoding](#)
[PDXlateToHostEx](#)
[PDXlateToPDFDocEnc](#)
[PDXlateToPDFDocEncEx](#)

**Example**

```
ASUns16 attr;
attr = PDWordGetAttr(myWord);
wlen = PDWordGetLength(myWord);
if(attr & WXE_HAS_NONALPHANUM)/* handle
    Mac quotes */
    PDXlateToPDFDocEnc(&msg[0], &msg[0],
        strlen(msg));
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDXlateToHostEx

```
ASInt32 PDXlateToHostEx (const char* inPdfStr,
ASInt32 inPdfStrSize, char* outHostStr,
ASInt32 outHostStrSize);
```

Description

Translates a string from Unicode or **PDFDocEncoding** to host encoding. This method is useful when setting or retrieving displayed text that might be in Unicode, such as text that appears in a text annotation or bookmark.

A character that cannot be converted to the destination encoding is replaced with a space.

Host encoding is a platform-dependent encoding for the host machine. For non-UNIX Roman systems, it is **MacRomanEncoding** in Mac OS and **WinAnsiEncoding** in Windows. In UNIX (except HP-UX) Roman systems, it is ISO8859-1 (ISO Latin-1); for HP-UX, it is **HP-ROMAN8**. See Appendix D in the [PDF Reference](#) for descriptions of **MacRomanEncoding**, **WinAnsiEncoding**, and **PDFDocEncoding**.

For non-Roman systems, the host encoding may be a variety of encodings, which are defined by a **CMap** (character map). See Section 5.6.4 in the [PDF Reference](#) for information on **CMaps**.

For non-Roman systems, use **PDXlatetoHostEx**. Use **PDGetHostEncoding** to determine whether the host encoding for a system is Roman or not.

In general, **PDXlatetoHostEx** operates the same as **PDXlateToHost** but requires an extra argument, since the sizes of the input and translated strings may differ. This method can be called instead of **PDXlateToHost**—and *must* be called for multi-byte character set systems.

Parameters

inPdfStr	Pointer to the string to translate (may point to the same memory as outHostStr , allowing strings to translate in place).
inPdfStrSize	The length of inPdfStr , in bytes.
outHostStr	<i>(Filled by the method)</i> Pointer to the translated string (may point to the same memory as inPdfStr).
outHostStrSize	The length of the outHostStr buffer, in bytes.

Return Value

Number of bytes in the translated string **outHostStr**.

Header File

PDCalls.h

**Related Methods**

[PDFontXlateToHost](#)
[PDFontXlateToUCS](#)
[PDGetHostEncoding](#)
[PDXlateToHost](#)
[PDXlateToPDFDocEnc](#)
[PDXlateToPDFDocEncEx](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020003` or higher.

PDXlateToPDFDocEnc

```
void PDXlateToPDFDocEnc (char* in, char* out,
    ASInt32 numBytes);
```

Description

Translates a string from host encoding to **PDFDocEncoding**. This method is useful when setting or retrieving displayed text that *must* be in **PDFDocEncoding** (or Unicode), such as text that appears in a text annotation or bookmark.

A character that cannot be converted to the destination encoding is replaced with a space. For example, **PDXlateToPDFDocEnc** converts `\n` to a space character (`\r` is present in **PDFDocEncoding** and is left unchanged).

Host encoding is a platform-dependent encoding for the host machine. For non-UNIX Roman systems, it is **MacRomanEncoding** in Mac OS and **WinAnsiEncoding** in Windows. In UNIX (except HP-UX) Roman systems, it is ISO8859-1 (ISO Latin-1); for HP-UX, it is **HP-ROMAN8**. See Appendix D in the [PDF Reference](#) for descriptions of **MacRomanEncoding**, **WinAnsiEncoding**, and **PDFDocEncoding**.

For non-Roman systems, the host encoding may be a variety of encodings, which are defined by a **CMap** (character map). See Section 5.6.4 in the [PDF Reference](#) for a list of predefined **CMaps**. Use [PDGetHostEncoding](#) to determine if a system's host encoding is Roman or not. For non-Roman systems, use [PDXlateToPDFDocEncEx](#).

In general, [PDXlateToPDFDocEncEx](#) can be called instead of **PDXlateToPDFDocEnc** since [PDXlateToPDFDocEncEx](#) works for **PDFDocEncoding** or Unicode.

Parameters

in	The string to translate (may point to the same memory as out , allowing strings to translate in place).
out	<i>(Filled by the method)</i> The translated string (may point to the same memory as in).
numBytes	Number of bytes in the string to translate.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDGetHostEncoding](#)
[PDXlateToHost](#)
[PDXlateToHostEx](#)

PDXlateToPDFDocEncEx

Example

```
ASUns16 attr;
attr = PDWordGetAttr(myWord);
wlen = PDWordGetLength(myWord);
if(attr & WXE_HAS_NONALPHANUM)/* handle
    Mac quotes */
    PDXlateToPDFDocEnc(&msg[0], &msg[0],
        strlen(msg));
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDXlateToPDFDocEncEx

```
ASInt32 PDXlateToPDFDocEncEx (ASBool bUseUnicode,
    const char* inHostStr, ASInt32 inHostStrSize, char* outPDFStr,
    ASInt32 outPDFStrSize);
```

Description

Translates a string from host encoding to **PDFDocEncoding** or Unicode. This method is useful when using text that must be in **PDFDocEncoding** or Unicode, such as text in a text annotation, bookmark, or article title.

A character that cannot be converted to the destination encoding is replaced with a space. For example, **PDXlateToPDFDocEncEx** converts `\n` to a space character (`\r` is present in **PDFDocEncoding** and is left unchanged).

Host encoding is a platform-dependent encoding for the host machine. For non-UNIX Roman systems, it is **MacRomanEncoding** in Mac OS and **WinAnsiEncoding** in Windows. In UNIX (except HP-UX) Roman systems, it is ISO8859-1 (ISO Latin-1); for HP-UX, it is **HP-ROMAN8**. See Appendix D in the [PDF Reference](#) for descriptions of **MacRomanEncoding**, **WinAnsiEncoding**, and **PDFDocEncoding**.

For non-Roman systems, the host encoding may be a variety of encodings, which are defined by a **CMap** (character map). See Section 5.6.4 in the [PDF Reference](#) for a list of predefined **CMaps**.

For non-Roman systems, use **PDXlateToPDFDocEncEx**. You can use **PDGetHostEncoding** to determine whether a system's host encoding is Roman or not.

In general, **PDXlateToPDFDocEncEx** can be called instead of **PDXlateToPDFDocEnc** since **PDXlateToPDFDocEncEx** works for **PDFDocEncoding** or Unicode.

Parameters

bUseUnicode	If true , translate the string to Unicode; otherwise use PDFDocEncoding .
inHostStr	Pointer to the string to translate (may point to the same memory as outPDFStr , allowing strings to translate in place).
inHostStrSize	Number of bytes in the string to translate.
outPDFStr	<i>(Filled by the method)</i> Pointer to the translated string (may point to the same memory as inHostStr).
outPDFStrSize	The length of the outPDFStr buffer, in bytes.

Return Value

Number of bytes in the translated string **outPDFStr**.

**Header File**

PDCalls.h

Related Methods

[PDFontXlateToHost](#)
[PDFontXlateToUCS](#)
[PDGetHostEncoding](#)
[PDXlateToHost](#)
[PDXlateToHostEx](#)
[PDXlateToPDFDocEnc](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020003** or higher.



PDAction

PDActionCanCopy

```
ASBool PDActionCanCopy (PDAction action);
```

Description

Tests whether the data from an action object can be copied to a clipboard for pasting. If the action is part of an action chain, the method tests all actions in the chain, and returns **true** only if all actions in the chain can be copied.

Parameters

action	The action to test.

Return Value

true if the action object or all actions in the action chain can be copied, **false** otherwise

Header File

PDCalls.h

Related Methods

[PDActionCanPaste](#)
[PDActionCopy](#)
[PDActionPaste](#)
[PDActionDestroyClipboardData](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



PDActionCanPaste

```
ASBool PDActionCanPaste (PDDoc doc,  
PDActionClipboardData data);
```

Description

Tests whether data from an action object that has been copied to a clipboard can be pasted into a destination document. Tests, for example, whether pasting is allowed by document permissions.

Parameters

doc	The destination document.
data	The action data to test.

Return Value

true if the action data can be pasted to the document, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDActionCanCopy](#)
[PDActionCopy](#)
[PDActionPaste](#)
[PDActionDestroyClipboardData](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDActionCopy

```
PDActionClipboardData PDActionCopy (PDAction action);
```

Description

Copies action object data to a clipboard structure, from which it can be pasted. When the [PDActionClipboardData](#) is no longer required, it must be explicitly freed using [PDActionDestroyClipboardData](#).

Parameters

action	The action to copy.

Return Value

The action clipboard data object.

Header File

PDCalls.h

Related Methods

[PDActionCanCopy](#)
[PDActionCanPaste](#)
[PDActionPaste](#)
[PDActionDestroyClipboardData](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDActionDestroy

```
void PDActionDestroy (PDAction action);
```

Description

Destroys an action object.

Parameters

action	The action to destroy.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDActionNew](#)
[PDActionNewFromDest](#)
[PDActionNewFromFileSpec](#)
[PDActionFromCosObj](#)

Example

```
if (PDActionIsValid(oldAction))  
    PDActionDestroy(oldAction);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDActionDestroyClipboardData

```
void PDActionDestroyClipboardData  
(PDActionClipboardData data);
```

Description

Destroys data that has been copied from an action object into a clipboard. Use this method when the clipboard data is no longer needed.

Parameters

data	The clipboard action data to destroy.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDActionCanCopy](#)
[PDActionCanPaste](#)
[PDActionCopy](#)
[PDActionPaste](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDActionEqual

```
ASBool PDActionEqual (PDAction action, PDAction action2);
```

Description

Compares two actions for equality. Two actions are equal if and only if their Cos objects are equal (see [CosObjEqual](#)).

Parameters

action	The first actions to be compared.
action2	The second action to be compared.

Return Value

Returns **true** if the actions are equal, **false** otherwise.

Header File

PDCalls.h

Related Methods

[CosObjEqual](#)

Example

```
if(PDActionIsValid(oldAction) && PDActionIsValid(newAction) &&
    PDActionEqual(oldAction, newAction))
    PDActionDestroy(oldAction);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDActionFromCosObj

```
PDAction PDActionFromCosObj (CosObj obj);
```

Description

Converts a dictionary Cos object to an action and verifies that the action is valid. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

<code>obj</code>	The dictionary Cos object whose action is obtained.

Return Value

The **PDAction** corresponding to `obj`.

Known Exceptions

Raises **pdErrBadAction** if the action is invalid as determined by **PDActionIsValid**.

Header File

PDCalls.h

Related Methods

```
PDActionGetCosObj
PDActionNew
PDActionNewFromDest
PDActionNewFromFileSpec
```

Example

```
CosObj theObj;
action = PDActionFromCosObj(cosObj);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDActionGetCosObj

```
CosObj PDActionGetCosObj (PDAction action);
```

Description

Gets the Cos object corresponding to an action. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

action	The action whose Cos object is obtained.

Return Value

Dictionary Cos object for the action. The contents of the dictionary can be enumerated using [CosObjEnum](#).

Header File

PDCalls.h

Related Methods

[PDActionFromCosObj](#)

Example

```
CosObj theObj;
if (PDActionGetSubtype(act) ==
    ASAtomFromString("Thread"))
    theObj = PDActionGetCosObj(act);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDActionGetDest

```
PDViewDestination PDActionGetDest (PDAction action);
```

Description

Gets an action's destination view. This only works for actions that contain a view destination—that is, actions whose subtype is **GoTo**.

For named destinations, this method may return a Cos string object or a Cos name object. See Section 8.2.1 in the *PDF Reference* for more information on named destinations.

NOTE: Since this method may not return a [PDViewDestination](#), use the [PDViewDestResolve](#) method on the returned value to obtain a [PDViewDestination](#).

Parameters

action	The action whose destination is obtained.

Return Value

The action's destination, which may be either a [PDViewDestination](#)—or for named destinations—a Cos string object or a Cos name object. Use the [PDViewDestResolve](#) method on this returned value to obtain a [PDViewDestination](#).

Header File

PDCalls.h

Related Methods

[PDActionGetFileSpec](#)
[PDViewDestResolve](#)

Example

```
PDViewDestination pdViewDest;
PDViewDestination pdViewDestFinal;
PDDoc pdDoc;

pdViewDest = PDActionGetDest(action);
pdViewDestFinal = PDViewDestResolve(pdViewDest, pdDoc);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDActionGetFileSpec

```
PDFFileSpec PDActionGetFileSpec (PDAction action);
```

Description

Gets a file specification from an action.

Not all types of actions have file specifications; this method only works for actions that contain a file specification. See Section 8.5 in the [PDF Reference](#) for more information on the contents of various types of actions.

Parameters

action	The action whose file specification is obtained.

Return Value

The action's file specification.

Header File

PDCalls.h

Related Methods

[PDActionGetDest](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDActionGetSubtype

```
ASAtom PDActionGetSubtype (PDAction action);
```

Description

Gets an action's subtype.

Parameters

action	The action whose subtype is obtained.

Return Value

The **ASAtom** corresponding to the action's subtype. The **ASAtom** can be converted to a string using [ASAtomGetString](#).

Header File

PDCalls.h

Example

```
CosObj theObj;
if (PDActionGetSubtype(act) == ASAtomFromString("Thread"))
    theObj = PDActionGetCosObj(act);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDActionIsValid

```
ASBool PDActionIsValid (PDAction action);
```

Description

Tests whether an action is valid. This method can be used in the following cases:

- To determine whether a [PDAction](#) returned from a method is really an action. For example, calling [PDLINKAnnotGetAction](#) returns an invalid action if no action is associated with the link annotation.
- To be sure an action has not been deleted.

Parameters

action	The action whose validity is determined.

Return Value

Returns **true** if the action is valid, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDActionEqual](#)

Example

```
if (PDActionIsValid(oldAction))
    PDActionDestroy(oldAction);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDActionNew

```
PDAction PDActionNew (PDDoc doc, ASAtom type);
```

Description

Creates a new action object.

Parameters

doc	The document in which the action is created.
type	The ASAtom corresponding to the action's subtype. The ASAtom can be obtained from a string using ASAtomFromString .

Return Value

The newly created **PDAction**.

Header File

PDCalls.h

Related Methods

```
PDActionNewFromDest
PDActionNewFromFileSpec
PDActionFromCosObj
PDActionDestroy
```

Example

```
#define Thread_K ASAtomFromString("Thread")
PDAction myAct = PDActionNew(doc, Thread_K);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDActionNewFromDest

```
PDAction PDActionNewFromDest (PDDoc doc,
PDViewDestination dest, PDDoc destDoc);
```

Description

Creates a new action that takes the user to the specified destination view. This method can only be used for destinations in the same document as the source document. Cross-document links must be built up from the Cos level, populating the Action dictionary for the **GotoR** action as described in Section 8.5.3 in the [PDF Reference](#).

Parameters

doc	The document in which the action is created and used.
dest	The destination view.
destDoc	Destination document. destDoc must be the same as doc .

Return Value

The newly created action.

Header File

PDCalls.h

Related Methods

[PDActionNew](#)
[PDActionNewFromFileSpec](#)
[PDActionFromCosObj](#)
[PDActionDestroy](#)

Example

```
dstPage = PDDocAcquirePage(pdDoc, (offset +
    Pages[i].pg)-1);
dest = PDViewDestCreate(pdDoc, dstPage,
    fit, &initRect, zoom, Pages[i].pg-1);
if(PDViewDestIsValid(dest)){
    pdAction = PDActionNewFromDest(pdDoc,
        dest, pdDoc);
    PDBookmarkSetAction(newBm, pdAction);
}
```

Product

reader, base, pro, pdfI

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDActionNewFromFileSpec

```
PDAction PDActionNewFromFileSpec (PDDoc containingDoc,  
ASAtom type, PDFFileSpec fileSpec);
```

Description

Creates an action of the specified type from a file specification.

Parameters

containingDoc	The document in which the action is created and used.
type	The type of action to create.
fileSpec	The file specification that is made part of an action.

Return Value

The newly created **PDAction**.

Header File

PDCalls.h

Related Methods

```
PDActionNew  
PDActionNewFromDest  
PDActionFromCosObj  
PDActionDestroy
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDActionPaste

```
PDAction PDActionPaste (PDDoc dest,
PDActionClipboardData data);
```

Description

Creates a new **PDAction** in the destination document, using clipboard data generated by [PDActionCopy](#). If the original **PDAction** was an action chain, the entire action chain is recreated. The returned **PDAction** is the first item in the chain.

When the data is no longer needed, use [PDActionDestroyClipboardData](#) to free the structure.

Parameters

doc	The destination document for the paste operation.
data	The clipboard structure holding the copied action data.

Return Value

A newly created action object (or the first such object in the action chain) associated with the specified document, containing the same data as the copied action.

Header File

PDCalls.h

Related Methods

[PDActionCanCopy](#)
[PDActionCanPaste](#)
[PDActionCopy](#)
[PDActionDestroyClipboardData](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDRegisterActionHandler

```
void PDRegisterActionHandler (PDActionHandler handler);
```

Description

Registers a handler for [PDAction](#) operations.

Parameters

handler	Pointer to a structure containing the action handler's callbacks. This structure must not be freed after this call, but must be retained.

Return Value

None

Header File

PDCalls.h

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.



PDAnnot

PDAnnotCanCopy

```
ASBool PDAnnotCanCopy (PDPage sourcePage, PDAnnot annot);
```

Description

Tests whether the data from an annotation on a given page can be copied to a clipboard for pasting. This depends on whether there is a [PDAnnotHandler](#) with copy and paste support for the annotation, and whether copying is allowed by document permissions.

Parameters

sourcePage	The page containing the annotation to test. Can be NULL (as when copying annotations while spawning a hidden template).
annot	The annotation to test.

Return Value

true if the annotation object can be copied, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDAnnotCanPaste](#)
[PDAnnotCopy](#)
[PDAnnotDestroyClipboardData](#)
[PDAnnotPaste](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDAnnotCanPaste

```
ASBool PDAnnotCanPaste (PDPPage destPage,
const ASFixedPoint *center, PDAnnotClipboardData data);
```

Description

Tests whether data from an annotation that has been copied to a clipboard can be pasted to a location on a page. Pasting can be disallowed by document permissions, or because the annotation cannot be accurately reproduced in the destination document.

Parameters

destPage	The page to which the annotation would be pasted.
center	The location for the center of the annotation on the destination page, or a NULL pointer to center the annotation on the destination page.
data	The copied annotation data to test.

Return Value

true if the annotation data can be pasted, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDAnnotCanCopy](#)
[PDAnnotCopy](#)
[PDAnnotDestroyClipboardData](#)
[PDAnnotPaste](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDAnnotCopy

```
PDAnnotClipboardData PDAnnotCopy (PDPage sourcePage,  
PDAnnot annot);
```

Description

Copies action object data to a clipboard structure, from which it can be pasted.

Parameters

sourcePage	The page containing the annotation to copy. Can be NULL (as when copying annotations while spawning a hidden template).
annot	The annotation to copy.

Return Value

The annotation clipboard data object.

Header File

PDCalls.h

Related Methods

[PDAnnotCanCopy](#)
[PDAnnotCanPaste](#)
[PDAnnotDestroyClipboardData](#)
[PDAnnotPaste](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDAnnotDestroyClipboardData

```
void PDAnnotDestroy (PDAnnotClipboardData data);
```

Description

Destroys data that has been copied from an annotation object into a clipboard. Use this method after successfully pasting the data to a new document.

Parameters

data	The clipboard annotation data to destroy.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDAnnotCanCopy](#)
[PDAnnotCanPaste](#)
[PDAnnotCopy](#)
[PDAnnotPaste](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDAnnotEqual

```
ASBool PDAnnotEqual (PDAnnot anAnnot, PDAnnot annot2);
```

Description

Tests whether two annotations are identical.

Parameters

anAnnot	The first annotation to compare.
annot2	The second annotation to compare.

Return Value

Returns **true** if the annotations are equal, **false** otherwise. Two annotations are equal if and only if their Cos objects are equal (see [CosObjEqual](#)).

Known Exceptions

[pdErrBadAnnotation](#)

Header File

PDCalls.h

Related Methods

[CosObjEqual](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDAnnotFromCosObj

```
PDAnnot PDAnnotFromCosObj (CosObj obj);
```

Description

Converts a dictionary Cos object to an annotation. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

obj	The dictionary Cos object whose annotation is obtained.

Return Value

The **PDAnnot** corresponding to the Cos object.

Known Exceptions

Raises **pdErrrBadAnnotation** if the annotation is invalid, as determined by **PDAnnotIsValid**.

Header File

PDCalls.h

Related Methods

[PDAnnotGetCosObj](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDAnnotGetColor

ASBool PDAnnotGetColor (**PDAnnot** anAnnot, **PDColorValue** color);

Description

Gets a note or link annotation’s color. If the annotation does not specify an explicit color, a default color is returned. Text annotations return “default yellow;” all others return black. Only RGB color specifications are currently supported.

Parameters

anAnnot	The note or link annotation whose color is obtained.
color	<i>(Filled by the method)</i> The annotation’s color, which is used as follows: ● Closed text note—the icon background color ● Open, un-selected text note—bounding rectangle color ● Open, selected text note—color of annotation’s title bar ● Link annotation—link border color

Return Value

Returns **true** if the annotation specifies an explicit color, **false** if a default color was used.

Header File

PDCalls.h

Related Methods

[PDAnnotSetColor](#)
[PDAnnotGetDate](#)
[PDAnnotGetFlags](#)
[PDAnnotGetRect](#)
[PDAnnotGetTitle](#)

Example

```
for (i=0; i<PDPageGetNumAnnots (pdPage); i++) {
    annot = PDPageGetAnnot (pdPage, i);
    if (PDAnnotIsValid (annot) &&
        PDAnnotGetSubtype (annot) == ASAtomFromString ("Text"))
    {
        if (!PDAnnotGetColor (annot, oldColor))
            PDAnnotSetColor (annot, newColor);
    }
}
```

Product

reader, base, pro, pdf



Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDAnnotGetCosObj

```
CosObj PDAnnotGetCosObj (PDAnnot annot);
```

Description

Gets the Cos object corresponding to an annotation. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

annot	The annotation whose Cos object is obtained.

Return Value

Dictionary Cos object for the annotation. The contents of the dictionary can be enumerated using [CosObjEnum](#). Returns a **NULL** Cos object if the annotation is not valid, as determined by [PDAnnotIsValid](#).

Header File

PDCalls.h

Related Methods

[PDAnnotFromCosObj](#)

Example

```
for (i=0;i<PDPageGetNumAnnots(pdPage);i++)
{
    annot = PDPageGetAnnot(pdPage, i);
    if(PDAnnotIsValid(annot) &&
        PDAnnotGetSubtype(annot) ==
        ASAtomFromString("Link"))
    {
        CosObj myCos = PDAnnotGetCosObj(
            annot);
        /* now process myCos */
        ...
    }
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDAnnotGetDate

```
ASBool PDAnnotGetDate (PDAnnot anAnnot, ATimeRecP date);
```

Description

Gets an annotation's date.

Parameters

anAnnot	The annotation whose date is obtained.
date	<i>(Filled by the method)</i> The annotation's time and date.

Return Value

Returns **true** if the annotation contains a date key and the value of that key can be successfully parsed as a date string, **false** otherwise.

Known Exceptions

Raises [pdErrBadAnnotation](#) if the annotation is not valid or if the value of the annotation's **M** (ModDate) key is not a string.

Raises if [genErrBadParm](#) **date** is **NULL**.

Header File

PDCalls.h

Related Methods

[PDAnnotSetDate](#)
[PDAnnotGetColor](#)
[PDAnnotGetFlags](#)
[PDAnnotGetRect](#)
[PDAnnotGetTitle](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDAnnotGetFlags

[ASUns32](#) PDAnnotGetFlags ([PDAnnot](#) anAnnot) ;

Description

Gets an annotation's flags.

Parameters

anAnnot	The annotation whose flags are obtained.
flags	<i>(Filled by the method)</i> An OR of the PDAnnot Flags values.

Return Value

The flags, or 0 if the annotation does not have a **flags** key.

Header File

PDCalls.h

Related Methods

[PDAnnotSetFlags](#)
[PDAnnotGetColor](#)
[PDAnnotGetDate](#)
[PDAnnotGetRect](#)
[PDAnnotGetTitle](#)

Example

```
if (!PDAnnotGetFlags (annot) &
    pdAnnotInvisible)
    PDAnnotSetFlags (annot, pdAnnotInvisible) ;
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDAnnotGetOCMD

```
PDOCMD PDAnnotGetOCMD (PDAnnot annot);
```

Description

Gets an optional-content membership dictionary (OCMD) object associated with the annotation.

Parameters

annot	The annotation from which the dictionary is obtained.

Return Value

The dictionary object, or **NULL** if the annotation does not contain a dictionary.

Header File

PDCalls.h

Related Methods

[PDAnnotSetOCMD](#)
[PDAnnotRemoveOCMD](#)
[PDEElementGetOCMD](#)
[PDOCMDFindOrCreate](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDAnnotGetRect

```
void PDAnnotGetRect (PDAnnot anAnnot, ASFixedRect* boxP);
```

Description

Gets the size and location of an annotation on its page.

Parameters

anAnnot	The annotation whose location and size are set.
boxP	<i>(Filled by the method)</i> Pointer to a rectangle that specifies the annotation's bounding rectangle, specified in user space coordinates.

Known Exceptions

[pdErrBadAnnotation](#)

Header File

PDCalls.h

Related Methods

[PDAnnotSetRect](#)
[PDAnnotGetColor](#)
[PDAnnotGetDate](#)
[PDAnnotGetFlags](#)
[PDAnnotGetTitle](#)

Example

```
ASFixedRect frect;
for(i=0;i<PDPageGetNumAnnots(pdPage);i++){
    annot = PDPageGetAnnot(pdPage, i);
    if(PDAnnotIsValid(annot) &&
        PDAnnotGetSubtype(annot) == ASAtomFromString("Text"))
    {
        PDAnnotGetRect(annot, &frect);
        /* Modify the annot's rect */
        ...
        /* Set the annot's rect */
        PDAnnotSetRect(annot, &frect);
    }
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDAnnotGetSubtype

```
ASAtom PDAnnotGetSubtype (PDAnnot anAnnot);
```

Description

Gets an annotation's subtype.

Parameters

anAnnot	The annotation whose subtype is obtained.

Return Value

The **ASAtom** for the annotation's subtype. This can be converted to a string using **ASAtomGetString**. The storage pointed to by the return value is owned by the Acrobat viewer and should be assumed to be valid only until the next call into *any* client API method; it should be immediately copied by the client if the client wants to reference it later.

Returns **ASAtomNull** if the annotation does not have a **Subtype** key or its value is not a name object.

Known Exceptions

pdErrBadAnnotation

Header File

PDCalls.h

Example

```
for (i=0;i<PDPageGetNumAnnots(pdPage);i++)
{
    annot = PDPageGetAnnot(pdPage, i);
    if (PDAnnotIsValid(annot) &&
        PDAnnotGetSubtype(annot) ==
        ASAtomFromString("Link"))
    {
        ...
    }
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDAnnotGetTitle

```
ASInt32 PDAnnotGetTitle (PDAnnot anAnnot, char* buffer,
ASInt32 bufSize);
```

Description

Gets an annotation's label text.

NOTE: For Roman viewers, this text is always stored in the **PDFDocEncoding**. For non-Roman character set viewers, this text is stored as **PDFDocEncoding** or Unicode, depending on the file's creator. Files created in a non-Roman environment contain Unicode versions of these strings; in a Roman environment, files contain **PDFDocEncoding** versions of these strings.

Parameters

anAnnot	The annotation whose label is obtained.
buffer	<i>(Filled by the method)</i> String into which the annotation's label string is copied. If string is non- NULL , up to bufsSize bytes are copied into buffer .
bufSize	Buffer size, in bytes. Up to bufSize bytes of the annotation label string are copied into the string and an ASCII null character is appended. The caller is expected to have allocated bufSize + 1 bytes to allow for the null. If buffer is NULL , copies nothing.

Return Value

If the string is non-**NULL**, the number of bytes copied, not counting the trailing null. If string is **NULL**, the number of bytes that would be copied if string were not **NULL**.

Known Exceptions

[pdErrBadAnnotation](#)

Header File

PDCalls.h

Related Methods

[PDAnnotSetTitle](#)
[PDAnnotGetColor](#)
[PDAnnotGetDate](#)
[PDAnnotGetRect](#)
[PDAnnotGetFlags](#)



Example

```
for(i=0;i<PDPageGetNumAnnots(pdPage);i++){
    annot = PDPageGetAnnot(pdPage, i);
    if(PDAnnotIsValid(annot) &&
        PDAnnotGetSubtype(annot) ==
        ASAtomFromString("Text"))
    {
        if(!PDAnnotGetTitle(annot, buf,
            sizeof(buf)))
        {
            strcat(buf, "-RD");
            PDAnnotSetTitle(annot, buf,
                sizeof(buf));
        }
    }
}
```

Product

reader, base, pro, pdfll

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDAnnotIsCurrentlyVisible

```
ASBool PDAnnotIsCurrentlyVisible (PDAnnot annot,
PDObjectContext ctx);
```

Description

Tests whether an annotation with an **OC** entry is visible in a given optional-content context, considering the current ON-OFF states of the optional-content groups in the optional-content dictionary (OCMD) and the dictionary's visibility policy.

Parameters

annot	The annotation to test.
ctx	The optional-content context in which the visibility is tested.

Return Value

Returns **true** if the annotation is visible in the given context or if the annotation has no **OC** entry, **false** if it is hidden.

Header File

PDCalls.h

Related Methods

[PDAnnotGetOCMD](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDAnnotIsValid

```
ASBool PDAnnotIsValid (PDAnnot anAnnot);
```

Description

Tests whether an annotation is valid. This is intended only to ensure that the annotation has not been deleted, not to ensure that all necessary information is present and valid.

Parameters

anAnnot	The annotation whose validity is tested.

Return Value

Returns **true** if **anannot** is a valid annotation object, **false** otherwise. An annotation is valid if it is a Cos dictionary object and has a **Rect** key.

Header File

PDCalls.h

Related Methods

[PDAnnotEqual](#)

Example

```
for (i=0; i<PDPageGetNumAnnots (pdPage) ; i++)
{
    annot = PDPageGetAnnot (pdPage, i);
    if (PDAnnotIsValid (annot) &&
        PDAnnotGetSubtype (annot) ==
        ASAtomFromString ("Link"))
    {
        ...
    }
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDAnnotNotifyDidChange

```
void PDAnnotNotifyDidChange (PDAnnot annot, ASAtom key,
ASInt32 err);
```

Description

Broadcasts a [PDAnnotDidChange](#) notification. Clients must call this method after making any change to a custom annotation.

Parameters

anAnnot	The annotation that has changed.
key	The ASAtom corresponding to the name of the key in the annotation's Cos dictionary that is changing.
err	An error code to pass to any method registered to receive the PDAnnotDidChange notification. Pass zero if the annotation was changed successfully. Pass a nonzero value if an error occurred while changing the annotation.

Return Value

None

Notifications

[PDAnnotDidChange](#)

Header File

PDCalls.h

Related Methods

[PDAnnotNotifyWillChange](#)
[AVAppRegisterNotification](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDAnnotNotifyWillChange

```
void PDAnnotNotifyWillChange (PDAnnot annot, ASAtom key);
```

Description

Broadcasts a [PDAnnotWillChange](#) notification. Clients must call this method before making any change to a custom annotation.

Parameters

anAnnot	The annotation that has changed.
key	The ASAtom corresponding to the name of the key in the annotation's Cos dictionary that is changing.

Return Value

None

Notifications

[PDAnnotWillChange](#)

Header File

PDCalls.h

Related Methods

[PDAnnotNotifyDidChange](#)
[AVAppRegisterNotification](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDAnnotPaste

```
PDAnnot PDAnnotPaste (PDPage destPage,
const ASFixedPoint *center, PDAnnotClipboardData data);
```

Description

Pastes copied annotation data from a clipboard structure to a new annotation object in a specified document. After successfully pasting the data, use [PDAnnotDestroyClipboardData](#) to free the associated memory.

Parameters

destPage	The page to which the annotation is pasted.
center	The location for the center of the annotation on the destination page, or a NULL pointer to center the annotation on the destination page.
data	The copied annotation data to paste.

Return Value

A newly created annotation object associated with the specified document, containing the same data as the copied annotation.

Header File

PDCalls.h

Related Methods

[PDAnnotCanCopy](#)
[PDAnnotCanPaste](#)
[PDAnnotCopy](#)
[PDAnnotDestroyClipboardData](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDAnnotRemoveOCMD

```
void PDAnnotRemoveOCMD (PDAnnot annot);
```

Description

Dissociates any optional-content membership dictionary (OCMD) object from the annotation.

Parameters

annot	The annotation for which to remove the dictionary.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDAnnotGetOCMD](#)

[PDAnnotSetOCMD](#)

[PDOCMDFindOrCreate](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDAnnotSetColor

```
void PDAnnotSetColor (PDAnnot anAnnot,
const PDColorValue color);
```

Description

Sets a note or link annotation's color. Only RGB color specifications are currently supported.

Parameters

anAnnot	The note or link annotation whose color is set.
color	The annotation's color, which is used as follows: • Closed text note—the icon background color • Open, unselected text note—bounding rectangle color • Open, selected text note—color of annotation's title bar • Link annotation—link border color

Return Value

None

Notifications

[PDAnnotWillChange](#)
[PDAnnotDidChange](#)

Header File

PDCalls.h

Related Methods

[PDAnnotGetColor](#)
[PDAnnotSetDate](#)
[PDAnnotSetFlags](#)
[PDAnnotSetRect](#)
[PDAnnotSetTitle](#)



Example

```
for (i=0; i<PDPageGetNumAnnots(pdPage); i++)
{
    annot = PDPageGetAnnot(pdPage, i);
    if (PDAnnotIsValid(annot) &&
        PDAnnotGetSubtype(annot) ==
            ASAtomFromString("Text"))
    {
        if (!PDAnnotGetColor(annot, oldColor))
            PDAnnotSetColor(annot, newColor);
    }
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDAnnotSetDate

```
void PDAnnotSetDate (PDAnnot anAnnot, const ATimeRecP date);
```

Description

Sets an annotation's date.

Parameters

anAnnot	The annotation whose date is set.
date	The annotation's time and date.

Return Value

None

Notifications

[PDAnnotWillChange](#)
[PDAnnotDidChange](#)

Header File

PDCalls.h

Related Methods

[PDAnnotGetDate](#)
[PDAnnotSetColor](#)
[PDAnnotSetFlags](#)
[PDAnnotSetRect](#)
[PDAnnotSetTitle](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDAnnotSetFlags

```
void PDAnnotSetFlags (PDAnnot anAnnot, ASUns32 flags);
```

Description

Sets an annotation's flags.

Parameters

anAnnot	The annotation whose flags are set.
flags	An OR of the PDAnnot Flags values.

Return Value

None

Notifications

[PDAnnotWillChange](#)
[PDAnnotDidChange](#)

Header File

PDCalls.h

Related Methods

[PDAnnotGetFlags](#)
[PDAnnotSetColor](#)
[PDAnnotSetDate](#)
[PDAnnotSetRect](#)
[PDAnnotSetTitle](#)

Example

```
if (!PDAnnotGetFlags (annot) &
    pdAnnotInvisible)
    PDAnnotSetFlags (annot, pdAnnotInvisible);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDAnnotSetOCMD

```
void PDAnnotSetOCMD (PDAnnot annot, PDOCMD pdocmd) ;
```

Description

Associates an optional-content membership dictionary (OCMD) object with the annotation, making it optionally visible according to the OCMD's visibility policy. If the annotation already has a dictionary, the method replaces it.

Parameters

annot	The annotation for which to set the dictionary.
pdocmd	The new dictionary.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDAnnotGetOCMD](#)

[PDOCMDFindOrCreate](#)

[PDAnnotRemoveOCMD](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDAnnotSetRect

```
void PDAnnotSetRect (PDAnnot anAnnot,
const ASFixedRect* newBox);
```

Description

Sets the size and location of an annotation on its page.

Parameters

anAnnot	The annotation whose location and size are set.
newBox	Pointer to a rectangle that specifies the annotation's bounding rectangle, specified in user space coordinates.

Notifications

[PDAnnotWillChange](#)
[PDAnnotDidChange](#)

Header File

PDCalls.h

Related Methods

[PDAnnotGetRect](#)
[PDAnnotSetColor](#)
[PDAnnotSetDate](#)
[PDAnnotSetFlags](#)
[PDAnnotSetTitle](#)

Example

```
ASFixedRect frect;
for(i=0;i<PDPageGetNumAnnots(pdPage);i++){
    annot = PDPageGetAnnot(pdPage, i);
    if(PDAnnotIsValid(annot) &&
        PDAnnotGetSubtype(annot) == ASAtomFromString("Text"))
    {
        PDAnnotGetRect(annot, &frect);
        /* Modify the annot's rect */
        ...
        /* Set the annot's rect */
        PDAnnotSetRect(annot, &frect);
    }
}
```

Product

reader, base, pro, pdf

**Availability**

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDAnnotSetTitle

```
void PDAnnotSetTitle (PDAnnot anAnnot, const char* str,
ASInt32 nBytes);
```

Description

Sets an annotation’s label text.

NOTE: For Roman viewers, this text is always stored in the **PDFDocEncoding**. For non-Roman character set viewers, this text is stored as **PDFDocEncoding** or Unicode, depending on the file’s creator. Files created in a non-Roman environment contain Unicode versions of these strings; in a Roman environment, files contain **PDFDocEncoding** versions of these strings.

Parameters

anAnnot	The annotation whose label is set.
str	String containing the label to set.
nBytes	Length of label. The first len bytes of str are used as the label.

Return Value

None

Notifications

[PDAnnotWillChange](#)
[PDAnnotDidChange](#)

Header File

PDCalls.h

Related Methods

[PDAnnotGetTitle](#)
[PDAnnotSetColor](#)
[PDAnnotSetDate](#)
[PDAnnotSetFlags](#)
[PDAnnotSetRect](#)



Example

```
for(i=0;i<PDPageGetNumAnnots(pdPage);i++){
    annot = PDPageGetAnnot(pdPage, i);
    if(PDAnnotIsValid(annot) &&
        PDAnnotGetSubtype(annot) ==
        ASAtomFromString("Text"))
    {
        if(!PDAnnotGetTitle(annot, buf,
            sizeof(buf)))
        {
            strcat(buf, "-RD");
            PDAnnotSetTitle(annot, buf,
                sizeof(buf));
        }
    }
}
```

Product

reader, base, pro, pdfll

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDAnnotHandler

PDGetAnnotHandlerByName

```
PDAnnotHandler PDGetAnnotHandlerByName (ASAtom name);
```

Description

Gets the annotation handler that handles the specified annotation type.

Parameters

name	Name of the requested annotation handler. The character string for the name can be converted to an ASAtom using ASAtomFromString .

Return Value

The annotation handler that services annotations of type name. Returns the default annotation handler if no handler services the specified annotation type.

Header File

PDCalls.h

Related Methods

[PDRegisterAnnotHandler](#)
[AVAppRegisterAnnotHandler](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDRegisterAnnotHandler

```
void PDRegisterAnnotHandler (PDAnnotHandler handler);
```

Description

Registers a handler for an annotation subtype, replacing any previous handler that had been registered for that subtype. The annotation handler is not registered if its [PDAnnotHandlerGetTypeProc](#) returns **NULL**.

To effectively use a **PDAnnotHandler**, the [AVAnnotHandler](#) associated with this annotation must have its [AVAnnotHandlerGetInfoProc](#) and [AVAnnotHandlerDeleteInfoProc](#) callbacks defined.

Parameters

handler	Pointer to a structure containing the annotation handler's callbacks. This structure must not be freed after this call, but must be retained.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDGetAnnotHandlerByName](#)
[AVAppGetAnnotHandlerByName](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDBead

PDBeadAcquirePage

```
PDPage PDBeadAcquirePage (PDBead bead, PDDoc doc);
```

Description

Acquires the page on which a bead is located.

Parameters

bead	The bead whose page is acquired.
doc	The document in which bead is located.

Return Value

The page on which the bead resides. The acquired page must be freed using [PDPageRelease](#) when it is no longer needed.

Header File

PDcalls.h

Related Methods

[PDPageRelease](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDBeadDestroy

```
void PDBeadDestroy (PDBead bead);
```

Description

Destroys a bead.

Parameters

bead	The bead to destroy.

Return Value

None

Known Exceptions

[pdErrBadBead](#)

Header File

PDCalls.h

Related Methods

[PDBeadNew](#)

[PDBeadFromCosObj](#)

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.



PDBeadEqual

```
ASBool PDBeadEqual (PDBead bead, PDBead bead2);
```

Description

Tests two beads for equality. This method is useful to detect the end of a thread since the last bead in a thread points to the first.

Parameters

bead	The first bead to compare.
bead2	The second bead to compare.

Return Value

Returns **true** if the two beads are identical, **false** otherwise. Two beads are equal if and only if their Cos objects are equal (see [CosObjEqual](#)).

Header File

```
PDCalls.h
```

Related Methods

```
CosObjEqual
```

Example

```
fbead = bead =
    PDThreadGetFirstBead(thread);
do {
    conPage = PDBeadAcquirePage(bead, pdDoc);
    pageNum = PDPageGetNumber(conPage);

    /* process pageNum */
    ...

    PDPageRelease(conPage);
    bead = PDBeadGetNext(bead);
} while (PDBeadIsValid(bead) &&
        !PDBeadEqual(fbead, bead));
```

Product

```
reader, base, pro, pdf
```

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDBeadFromCosObj

```
PDBead PDBeadFromCosObj (CosObj obj) ;
```

Description

Gets the **PDBead** corresponding to a Cos object, after checking the bead's validity. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

obj	The dictionary Cos object whose bead is obtained.

Return Value

The **PDBead** object for the bead.

Known Exceptions

Raises **pdErrBadBead** if the bead is not valid, as determined by **PDBeadIsValid**.

Header File

PDCalls.h

Related Methods

[PDBeadGetCosObj](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDBeadGetCosObj

```
CosObj PDBeadGetCosObj ( PDBead bead) ;
```

Description

Gets the Cos object corresponding to a bead. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

bead	The bead whose Cos object is obtained.

Return Value

Dictionary Cos object for the bead. The contents of the dictionary can be enumerated using [CosObjEnum](#). Returns a **NULL** Cos object if [PDBeadIsValid](#)(bead) returns **false**.

Header File

PDCalls.h

Related Methods

[PDBeadFromCosObj](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDBeadGetIndex

```
ASInt32 PDBeadGetIndex (PDBead bead) ;
```

Description

Gets the index of a bead in its thread.

Parameters

bead	The bead whose index is obtained.

Return Value

The index of the bead in its thread. The first bead in a thread has an index of zero.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDBeadGetNext

```
PDBead PDBeadGetNext (PDBead bead);
```

Description

Gets the next bead in a thread.

Parameters

bead	The bead for which the next bead is obtained.

Return Value

The next bead, or a **NULL** Cos object (cast to a **PDBead** using [PDBeadFromCosObj](#)). On the last bead, **PDBeadGetNext** returns the first bead.

Header File

PDCalls.h

Related Methods

[PDBeadGetPrev](#)
[PDThreadGetFirstBead](#)
[PDBeadEqual](#)

Example

```
fbead = bead =
    PDThreadGetFirstBead(thread);
do {
    conPage = PDBeadAcquirePage(bead, pdDoc);
    pageNum = PDPageGetNumber(conPage);

    /* process pageNum */
    ...

    PDPageRelease(conPage);
    bead = PDBeadGetNext(bead);
} while (PDBeadIsValid(bead) &&
        !PDBeadEqual(fbead, bead));
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDBeadGetPrev

```
PDBead PDBeadGetPrev (PDBead bead);
```

Description

Gets the previous bead in a thread.

Parameters

bead	The bead for which the previous bead is obtained.

Return Value

The previous bead, or a **NULL** Cos object (cast to a **PDBead** using [PDBeadFromCosObj](#)) if this is the first bead in the thread.

Header File

PDCalls.h

Related Methods

[PDBeadGetNext](#)
[PDThreadGetFirstBead](#)
[PDBeadEqual](#)

Example

```
fbead = bead =
    PDThreadGetFirstBead(thread);
do {
    conPage = PDBeadAcquirePage(bead, pdDoc);
    pageNum = PDPageGetNumber(conPage);

    /* process pageNum */
    ...

    PDPageRelease(conPage);
    bead = PDBeadGetNext(bead);
} while (PDBeadIsValid(bead) &&
        !PDBeadEqual(fbead, bead));
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDBeadGetRect

```
void PDBeadGetRect (PDBead bead, ASFixedRectP rectP);
```

Description

Gets a bead's bounding rectangle.

Parameters

bead	The bead whose bounding rectangle is obtained.
rectP	<i>(Filled by the method)</i> Pointer to a ASFixedRect specifying the bead's bounding rectangle, specified in user space coordinates.

Return Value

None

Header File

PDCalls.h

Related Method

[PDBeadSetRect](#)
[PDBeadGetIndex](#)
[PDBeadGetNext](#)
[PDBeadGetPrev](#)
[PDBeadGetThread](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.



PDBeadGetThread

```
PDThread PDBeadGetThread (PDBead bead) ;
```

Description

Gets the thread containing the specified bead.

Parameters

bead	The bead whose thread is obtained.

Return Value

The bead's thread, or a **NULL** Cos object if the bead does not belong to a thread.

Header File

PDCalls.h

Related Methods

[PDBeadGetRect](#)
[PDBeadGetIndex](#)
[PDBeadGetNext](#)
[PDBeadGetPrev](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDBeadInsert

```
void PDBeadInsert (PDBead bead, PDBead newNext);
```

Description

Inserts a bead after the specified bead.

Parameters

bead	The bead after which newNext will be inserted.
newNext	The bead to insert.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDBeadNew](#)

[PDThreadSetFirstBead](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDBeadIsValid

```
ASBool PDBeadIsValid (PDBead bead);
```

Description

Tests a bead's validity. This is intended only to ensure that the bead has not been deleted, not to ensure that all necessary information is present and valid.

Parameters

bead	The bead whose validity is tested.

Return Value

Returns **true** if the bead is valid, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDBeadEqual](#)

Example

```
fbead = bead =
    PDThreadGetFirstBead(thread);
do {
    conPage = PDBeadAcquirePage(bead, pdDoc);
    pageNum = PDPageGetNumber(conPage);

    /* process pageNum */
    ...

    PDPageRelease(conPage);
    bead = PDBeadGetNext(bead);
} while (PDBeadIsValid(bead) &&
        !PDBeadEqual(fbead, bead));
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDBeadNew

```
PDBead PDBeadNew (PDPage page, const ASFixedRectP destRect);
```

Description

Creates a new bead on the specified page. The newly created bead is not linked to a thread or another bead. Use [PDThreadSetFirstBead](#) to make the bead the first bead in a thread. Use [PDBeadInsert](#) to link it to another bead.

Parameters

page	The page on which the bead is created.
destRect	Pointer to a ASFixedRect specifying the bead's bounding rectangle, specified in user space coordinates.

Return Value

The newly created bead.

Header File

PDCalls.h

Related Methods

[PDThreadSetFirstBead](#)
[PDBeadInsert](#)
[PDBeadFromCosObj](#)
[PDBeadDestroy](#)

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDBeadSetPage

```
void PDBeadSetPage (PDBead bead, PDPage newPage);
```

Description

Sets the page for a bead.

Parameters

bead	The bead whose page is set.
newPage	The page on which bead is located.

Return Value

None

Known Exceptions

[pdErrBadBead](#)

Header File

PDcalls.h

Related Methods

[PDBeadSetRect](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDBeadSetRect

```
void PDBeadSetRect (PDBead bead,  
const ASFixedRectP newDestRect) ;
```

Description

Sets a bead’s bounding rectangle.

Parameters

bead	The bead whose bounding rectangle is set.
newDestRect	Pointer to a ASFixedRect specifying the bead’s bounding rectangle, specified in user space coordinates.

Return Value

None

Header File

PDCalls.h

Related Methods

- [PDBeadGetRect](#)
- [PDBeadSetPage](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDBookmark

PDBookmarkAddChild

```
void PDBookmarkAddChild (PDBookmark parent,
                          PDBookmark aBookmark);
```

Description

Adds **aBookmark** as the last child of **parent**, adjusting the tree containing **parent** appropriately. If **parent** previously had no children, it is open after the child is added.

Parameters

parent	The parent of the bookmark being added.
aBookmark	The bookmark that will become the last child of aBookmark . aBookmark must have been previously unlinked.

Return Value

None

Notifications

[PDBookmarkDidChangePosition](#)

Header File

PDCalls.h

Related Methods

[PDBookmarkAddNewSibling](#)
[PDBookmarkAddNewChild](#)
[PDBookmarkAddSubtree](#)
[PDBookmarkAddPrev](#)
[PDBookmarkAddNext](#)
[PDBookmarkUnlink](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDBookmarkAddNewChild

```
PDBookmark PDBookmarkAddNewChild (PDBookmark aBookmark,
char* initialText);
```

Description

Adds a new bookmark to the tree containing **aBookmark**, as the new last child of **aBookmark**. If **aBookmark** previously had no children, it will be open after the child is added.

Parameters

aBookmark	The bookmark to which a new last child is added.
initialText	The new bookmark's title.

Return Value

The newly created bookmark.

Notifications

[PDBookmarkDidChangePosition](#)
[PDBookmarkWasCreated](#)

Header File

PDCalls.h

Related Methods

[PDBookmarkAddNewSibling](#)
[PDBookmarkAddSubtree](#)
[PDBookmarkAddPrev](#)
[PDBookmarkAddNext](#)
[PDBookmarkAddChild](#)
[PDBookmarkUnlink](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDBookmarkAddNewSibling

```
PDBookmark PDBookmarkAddNewSibling (PDBookmark aBookmark,
char* initialText);
```

Description

Adds a new bookmark to the tree containing a **aBookmark**, as the new right sibling.

Parameters

aBookmark	The bookmark that will be the left sibling of the new bookmark.
initialText	The new bookmark's title.

Return Value

The newly created bookmark.

Notifications

[PDBookmarkDidChangePosition](#)
[PDBookmarkWasCreated](#)

Header File

PDCalls.h

Related Methods

[PDBookmarkAddChild](#)
[PDBookmarkAddNewChild](#)
[PDBookmarkAddNext](#)
[PDBookmarkAddPrev](#)
[PDBookmarkAddSubtree](#)
[PDBookmarkUnlink](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDBookmarkAddNext

```
void PDBookmarkAddNext (PDBookmark aBookmark,
PDBookmark newNext);
```

Description

Adds **newNext** as the new right sibling to **aBookmark**.

Parameters

aBookmark	The bookmark that will receive a new right sibling .
newNext	The bookmark to become the new right sibling of aBookmark . newNext must have been previously unlinked.

Return Value

None

Notifications

[PDBookmarkDidChangePosition](#)

Header File

PDCalls.h

Related Methods

[PDBookmarkAddNewSibling](#)
[PDBookmarkAddChild](#)
[PDBookmarkAddSubtree](#)
[PDBookmarkAddPrev](#)
[PDBookmarkAddNewChild](#)
[PDBookmarkUnlink](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDBookmarkAddPrev

```
void PDBookmarkAddPrev (PDBookmark aBookmark,
PDBookmark newPrev);
```

Description

Adds **newPrev** as the new left sibling to **aBookmark**, adjusting the tree containing **aBookmark** appropriately.

Parameters

aBookmark	The bookmark that will receive a new left sibling newPrev .
newPrev	The bookmark to become the new left sibling of aBookmark . newPrev must have been previously unlinked.

Return Value

None

Notifications

[PDBookmarkDidChangePosition](#)

Header File

PDCalls.h

Related Methods

[PDBookmarkAddNewSibling](#)
[PDBookmarkAddNewChild](#)
[PDBookmarkAddSubtree](#)
[PDBookmarkAddNext](#)
[PDBookmarkAddChild](#)
[PDBookmarkUnlink](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDBookmarkAddSubtree

```
void PDBookmarkAddSubtree (PDBookmark aBookmark,
PDBookmark source, char* sourceTitle);
```

Description

Adds a copy of the bookmark sub-tree **source** to **aBookmark**, as a new last child of **aBookmark**. This new item will have the text value **sourceTitle**, will be open, and will have no destination attribute. **source** must have been previously unlinked. If **aBookmark** previously had no children, it will be open after the subtree is added.

Parameters

aBookmark	The bookmark to which the subtree source will be added as a new last child.
source	The bookmark subtree to add.
sourceTitle	The new bookmark's title.

Return Value

None

Notifications

[PDBookmarkWillChange](#)
[PDBookmarkDidChange](#)
[PDBookmarkDidChangePosition](#)

Header File

PDCalls.h

Related Methods

[PDBookmarkAddNewSibling](#)
[PDBookmarkAddNewChild](#)
[PDBookmarkAddPrev](#)
[PDBookmarkAddNext](#)
[PDBookmarkAddChild](#)
[PDBookmarkUnlink](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDBookmarkDestroy

```
void PDBookmarkDestroy (PDBookmark aBookmark);
```

Description

Removes a bookmark subtree from the bookmark tree containing it.

Parameters

aBookmark	The root bookmark of the subtree to remove.

Return Value

None

Notifications

[PDBookmarkWillDestroy](#)
[PDBookmarkDidDestroy](#)

Header File

PDCalls.h

Related Methods

[PDBookmarkAddNewChild](#)
[PDBookmarkAddNewSibling](#)
[PDBookmarkFromCosObj](#)
[PDBookmarkUnlink](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.



PDBookmarkEqual

```
ASBool PDBookmarkEqual (PDBookmark aBookmark,  
PDBookmark bookmark2);
```

Description

Tests whether two bookmarks are equal. Two bookmarks are equal if and only if their Cos objects are equal (see [CosObjEqual](#)) .

Parameters

aBookmark	The first bookmark to compare.
bookmark2	The second bookmark to compare.

Return Value

Returns **true** if the bookmarks are equal, **false** otherwise.

Header File

PDCalls.h

Related Methods

[CosObjEqual](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDBookmarkFromCosObj

```
PDBookmark PDBookmarkFromCosObj (CosObj obj);
```

Description

Converts a Cos dictionary object to a bookmark and checks the validity of the bookmark. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

obj	The dictionary Cos object whose bookmark is obtained.

Return Value

Bookmark corresponding to the Cos object.

Known Exceptions

Raises [pdErrrBadBookmark](#) if the bookmark is not valid as determined by [PDBookmarkIsValid](#).

Header File

PDCalls.h

Related Methods

[PDBookmarkGetCosObj](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDBookmarkGetAction

```
PDAction PDBookmarkGetAction (PDBookmark aBookmark);
```

Description

Gets a bookmark's action. After you obtain the action, you can execute it with [AVDocPerformAction](#).

Parameters

aBookmark	The bookmark whose action is obtained.
action	<i>(Filled by the method)</i> The bookmark's action.

Return Value

The bookmark's action.

Header File

PDCalls.h

Related Methods

[AVDocPerformAction](#)
[PDBookmarkSetAction](#)
[PDLinkAnnotGetAction](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDBookmarkGetByTitle

```
PDBookmark PDBookmarkGetByTitle (PDBookmark aBookmark,
char* aName, ASInt32 nameLen, ASInt32 maxdepth);
```

Description

Gets the first bookmark whose title is **aName**.

NOTE: For Roman viewers, this text is always stored in the **PDFDocEncoding**. For non-Roman character set viewers, this text is stored as **PDFDocEncoding** or Unicode, depending on the file's creator. Files created in a non-Roman environment contain Unicode versions of these strings; in a Roman environment, files contain **PDFDocEncoding** versions of these strings.

Parameters

aBookmark	The root of the bookmark subtree to search.
aName	The text value to search for. Character codes in aName are interpreted using the PDFDocEncoding .
nameLen	The length of aName .
maxDepth	The number of subtree levels to search, not counting the root level. • 0 — Only look at aBookmark, not at any of its children. • 1 — Check aBookmark and its children, but not any grandchildren or great grandchildren, and so on. • -1 — Check the entire subtree.

Return Value

The bookmark with the specified title, or a **NULL** Cos object if there is no such bookmark.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDBookmarkGetColor

```
ASBool PDBookmarkGetColor (PDBookmark bm,
PDColorValue pdcvOut);
```

Description

Retrieves the color of the specified bookmark.

Parameters

bm	The bookmark in question.
pdcvOut	(Filled by the method) Color of the bookmark in PDColorValue format.

Return Value

Returns **true** if color was specified, **false** otherwise (default color returned in **pdcvOut**).

Known Exceptions

If bookmark is invalid or existing color is malformed in the PDF file.

Header File

PDCalls.h

Related Methods

[PDBookmarkSetColor](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDBookmarkGetCosObj

[CosObj](#) PDBookmarkGetCosObj ([PDBookmark](#) aBookmark);

Description

Gets the Cos object for a bookmark. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

aBookmark	The bookmark whose Cos object is obtained.

Return Value

Dictionary Cos object for the bookmark. The contents of the dictionary can be enumerated using [CosObjEnum](#).

Header File

PDCalls.h

Related Methods

[PDBookmarkFromCosObj](#)
[CosObjEnum](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDBookmarkGetCount

```
ASInt32 PDBookmarkGetCount (PDBookmark aBookmark);
```

Description

Gets the number of open bookmarks in a subtree.

Parameters

aBookmark	The root bookmark of a subtree to count.

Return Value

Number of open bookmarks in the subtree (not including **aBookmark**).

Header File

PDCalls.h

Related Methods

[PDBookmarkGetIndent](#)
[PDBookmarkHasChildren](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDBookmarkGetFirstChild

```
PDBookmark PDBookmarkGetFirstChild (PDBookmark aBookmark);
```

Description

Gets a bookmark's first child.

Parameters

aBookmark	The bookmark whose first child is obtained.

Return Value

First child of **aBookmark**, or a **NULL** Cos object if **aBookmark** has no children.

Header File

PDCalls.h

Related Methods

[PDBookmarkGetParent](#)
[PDBookmarkGetLastChild](#)
[PDBookmarkHasChildren](#)
[PDBookmarkGetNext](#)
[PDBookmarkGetPrev](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDBookmarkGetFlags

```
ASInt32 PDBookmarkGetFlags (PDBookmark bm) ;
```

Description

Retrieves the flags of the specified bookmark.

Parameters

bm	The bookmark whose flags are obtained.

Return Value

Bookmark's flags. Bit 1 (least-significant bit) indicates italic font; bit 2 indicates bold font.

Known Exceptions

If bookmark is invalid or existing style is malformed in the PDF file.

Header File

PDCalls.h

Related Methods

[PDBookmarkSetFlags](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDBookmarkGetIndent

```
ASInt32 PDBookmarkGetIndent (PDBookmark aBookmark);
```

Description

Returns the indentation level of a bookmark in its containing tree.

Parameters

aBookmark	The bookmark whose indentation level is obtained.

Return Value

The indentation level of **aBookmark** in its containing tree. The root level has an indentation level of zero.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDBookmarkGetLastChild

```
PDBookmark PDBookmarkGetLastChild (PDBookmark aBookmark);
```

Description

Gets a bookmark's last child.

Parameters

aBookmark	The bookmark whose last child is obtained.

Return Value

Last child of **aBookmark**, or a **NULL** Cos object if **aBookmark** has no children.

Header File

PDCalls.h

Related Methods

[PDBookmarkGetParent](#)
[PDBookmarkGetFirstChild](#)
[PDBookmarkGetNext](#)
[PDBookmarkGetPrev](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDBookmarkGetNext

```
PDBookmark PDBookmarkGetNext (PDBookmark aBookmark);
```

Description

Gets bookmark's next (right) sibling.

Parameters

aBookmark	The bookmark whose right sibling is obtained.

Return Value

aBookmark's next (right) sibling. Returns a **NULL** Cos object if **aBookmark** has no next sibling (that is, it is its parent's last child).

Header File

PDCalls.h

Related Methods

[PDBookmarkGetParent](#)
[PDBookmarkGetFirstChild](#)
[PDBookmarkGetLastChild](#)
[PDBookmarkGetPrev](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDBookmarkGetParent

```
PDBookmark PDBookmarkGetParent (PDBookmark aBookmark);
```

Description

Gets a bookmark's parent bookmark.

Parameters

aBookmark	The bookmark whose parent is obtained.

Return Value

Parent bookmark of **aBookmark**, or a **NULL** Cos object if **aBookmark** is the root of its tree.

Header File

PDCalls.h

Related Methods

[PDBookmarkGetFirstChild](#)
[PDBookmarkGetLastChild](#)
[PDBookmarkGetNext](#)
[PDBookmarkGetPrev](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDBookmarkGetPrev

```
PDBookmark PDBookmarkGetPrev (PDBookmark aBookmark);
```

Description

Returns bookmark's previous (left) sibling.

Parameters

aBookmark	The bookmark whose left sibling is obtained.

Return Value

Previous (left) sibling of **aBookmark**, or a **NULL** Cos object if bookmark has no previous sibling (it is its parent's first child).

Header File

PDCalls.h

Related Methods

[PDBookmarkGetParent](#)
[PDBookmarkGetFirstChild](#)
[PDBookmarkGetLastChild](#)
[PDBookmarkGetNext](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDBookmarkGetTitle

```
ASInt32 PDBookmarkGetTitle (PDBookmark aBookmark,
char* buffer, ASInt32 bufSize);
```

Description

Gets a bookmark's title.

NOTE: For Roman viewers, this title text is always stored in **PDFDocEncoding**. For non-Roman character set viewers, this text is stored as **PDFDocEncoding** or Unicode, depending on the file's creator. Files created in a non-Roman environment contain Unicode versions of these strings; in a Roman environment, files contain **PDFDocEncoding** versions of these strings.

Parameters

aBookmark	The bookmark whose title is obtained.
buffer	<i>(Filled by the method)</i> Buffer into which the title will be written. If buffer is non- NULL , its length is assumed to be bufSize + 1, because a null byte is appended to the title. The returned text remains valid only until next PModel method call. The text may be converted to a platform's native encoding using PDXlateToHost or PDXlateToHostEx .
bufSize	The size of the buffer.

Return Value

The number of bytes copied into **buffer**, not counting the trailing null byte. If **buffer** is **NULL**, the number of bytes in the bookmark is returned.

Header File

PDCalls.h

Related Methods

- [PDBookmarkSetTitle](#)
- [PDXlateToHost](#)
- [PDXlateToHostEx](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDBookmarkHasChildren

```
ASBool PDBookmarkHasChildren (PDBookmark aBookmark);
```

Description

Tests whether a bookmark has children.

Parameters

aBookmark	The bookmark to test.

Return Value

Returns **true** if **aBookmark** has any children, **false** otherwise.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDBookmarksOpen

```
ASBool PDBookmarksOpen (PDBookmark aBookmark) ;
```

Description

Tests whether a bookmark is open. An open bookmark shows all its children.

Parameters

aBookmark	The bookmark to test.

Return Value

Returns **true** if the bookmark is open, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDBookmarkSetOpen](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDBookmarksValid

```
ASBool PDBookmarksValid (PDBookmark aBookmark);
```

Description

Tests whether a bookmark is valid. This is intended only to ensure that the bookmark has not been deleted, not to ensure that all necessary information is present and valid.

Parameters

aBookmark	The bookmark whose validity is tested.

Return Value

Returns **true** if the bookmark is valid, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDBookmarkEqual](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDBookmarkRemoveAction

```
void PDBookmarkRemoveAction (PDBookmark aBookmark);
```

Description

Removes a bookmark's action.

Parameters

aBookmark	The bookmark whose action is removed.

Return Value

None

Known Exceptions

[pdErrBadAction](#)

Notifications

[PDBookmarkDidChange](#)
[PDBookmarkWillChange](#)

Header File

PDCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



PDBookmarkSetAction

```
void PDBookmarkSetAction (PDBookmark aBookmark,  
PDAction action);
```

Description

Sets a bookmark's action.

Parameters

aBookmark	The bookmark whose action is set.
action	The bookmark's action.

Return Value

None

Notifications

[PDBookmarkWillChange](#)
[PDBookmarkDidChange](#)

Header File

PDCalls.h

Related Methods

[PDBookmarkGetAction](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.



PDBookmarkSetColor

```
void PDBookmarkSetColor (PDBookmark bm, PDColorValue pdcvIn);
```

Description

Sets the color of the specified bookmark.

Parameters

aBookmark	The bookmark whose color is set.
pdcvIn	The new color. Must be in DeviceRGB.

Return Value

None

Known Exceptions

If color is **NULL** or not in DeviceRGB, bookmark is invalid or existing bookmark color is malformed.

Notifications

[PDBookmarkWillChange](#) with **C** as the key.

[PDBookmarkDidChange](#) with **C** as the key.

Header File

PDCalls.h

Related Methods

[PDBookmarkGetColor](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDBookmarkSetFlags

```
void PDBookmarkSetFlags (PDBookmark bm, ASInt32 nFlags);
```

Description

Sets the flags of the specified bookmark.

Parameters

bm	The bookmark whose flags are set.
nFlags	The new bookmark flags. Bit 1 (the least-significant bit) indicates italic font; bit 2 indicates bold font.

Return Value

None

Known Exceptions

If bookmark is invalid or existing flags are malformed in the PDF file.

Notifications

[PDBookmarkWillChange](#) with **F** as the key.

[PDBookmarkDidChange](#) with **F** as the key

Header File

PDCalls.h

Related Methods

[PDBookmarkGetFlags](#)

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

PDBookmarkSetOpen

```
void PDBookmarkSetOpen (PDBookmark aBookmark, ASBool isOpen);
```

Description

Opens or closes a bookmark. An open bookmark shows its children, while a closed bookmark does not.

Parameters

aBookmark	The bookmark to open or close.
isOpen	true if the bookmark is opened, false if the bookmark is closed.

Return Value

None

Notifications

[PDBookmarkWillChange](#)
[PDBookmarkDidChange](#)

Header File

PDCalls.h

Related Methods

[PDBookmarkIsOpen](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDBookmarkSetTitle

```
void PDBookmarkSetTitle (PDBookmark aBookmark,
    const char* str, ASInt32 nBytes);
```

Description

Sets a bookmark's title.

NOTE: For Roman viewers, this title text is always stored in the **PDFDocEncoding**. For non-Roman character set viewers, this text is stored as **PDFDocEncoding** or Unicode, depending on the file's creator. Files created in a non-Roman environment contain Unicode versions of these strings; in a Roman environment, files contain **PDFDocEncoding** versions of these strings.

Parameters

aBookmark	The bookmark whose title is set.
str	Read-only string containing the bookmark's new title. The text must be encoded using PDFDocEncoding . Strings encoded using a platform's native encoding can be converted to PDFDocEncoding using PDXlateToPDFDocEnc or PDXlateToPDFDocEncEx .
nBytes	Number of bytes of str to copy.

Return Value

None

Known Exceptions

Raises **pdErrBookmarksError** if there is an error setting the title.

Notifications

[PDBookmarkWillChange](#)
[PDBookmarkDidChange](#)

Header File

PDCalls.h

Related Methods

[PDBookmarkGetTitle](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDBookmarkUnlink

```
void PDBookmarkUnlink (PDBookmark aBookmark);
```

Description

Unlinks a bookmark from the bookmark tree that contains it, and adjusts the tree appropriately.

Parameters

aBookmark	The bookmark to unlink.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDBookmarkDestroy](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDCharProc

PDCharProcEnum

```
void PDCharProcEnum (PDCharProc charProc,
                     PDGraphicEnumMonitor mon, void* clientData);
```

Description

Enumerates the graphic description of a single character procedure for a Type 3 font. To enumerate all the character procedures in a Type 3 font (but not their graphic descriptions), use [PDFontEnumCharProcs](#).

Parameters

charProc	The character procedure whose graphic description is enumerated.
mon	Pointer to a structure containing user-supplied callbacks that are called for each drawing operator on a page. Enumeration halts if any procedure returns false .
clientData	Pointer to user-supplied data to pass to each monitor routine that is called.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDCharProcEnumWithParams](#)
[PDFontEnumCharProcs](#)
[PDDocEnumFonts](#)
[PDDocEnumLoadedFonts](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDCharProcEnumWithParams

```
void PDCharProcEnumWithParams (PDCharProc charProc,  
    PDGraphicEnumParams params)
```

Description

Enumerates the graphic description of a single character procedure for a Type 3 font, for those contents that are visible in a given optional-content context. The parameters include both the monitor and data you would pass to [PDCharProcEnum](#), and an optional-content context that determines which contents are visible.

Parameters

charProc	The character procedure whose graphic descriptions are enumerated.
params	The parameters, including the optional-content context to use for content visibility.

Return Value

None

Known Exceptions

[pdPErrUnableToCreateRasterPort](#)

Header File

PDcalls.h

Related Methods

[PDCharProcEnum](#)
[PDFormEnumPaintProcWithParams](#)
[PDPageEnumContents](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.



PDCharProcGetCosObj

```
CosObj PDCharProcGetCosObj (PDCharProc obj);
```

Description

Get the stream Cos object associated with the **PDCharProc**. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

obj	The character procedure whose Cos object is obtained.

Return Value

The character procedure's stream Cos object.

Header File

PDCalls.h

Related Methods

[PDFontEnumCharProcs](#)
[PDCharProcEnum](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to 0x00020000 or higher.

PDDoc

PDFCryptAuthorizeFilterAccess

```
ASBool PDFCryptAuthorizeFilterAccess (PDDoc doc,
ASAtom handlerName, ASAtom filterName, ASBool bEncrypt);
```

Description

Gets authorization to encrypt or decrypt an embedded file, where that file's cryptographic filter is not the one used to open the document in which the file is embedded.

Parameters

doc	The document containing the embedded file whose filter access is requested.
handlerName	The security handler containing the Authorize callback procedure to run.
filterName	The ASAtom corresponding to the name of the security filter used by the embedded file.
bEncrypt	When true , the access is required for an encryption operation. When false , it is a decryption operation.

Return Value

true if the authorization succeeds, **false** otherwise.

Known Exceptions

Raises **pdErrNoCryptHandler** if there is no security handler registered for the document.

Header File

PDCalls.h

Related Methods

```
PDDocSetNewCryptFilterData
PDDocSetNewCryptFilterMethod
PDDocSetNewDefaultFilters
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



PDDocAcquire

```
void PDDocAcquire (PDDoc doc);
```

Description

Increments a document's reference count. The document will not be closed until the reference count is zero, or the application terminates.

Parameters

<code>doc</code>	The document whose reference count is incremented.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocRelease](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDDocAcquirePage

[PDPPage](#) PDDocAcquirePage ([PDDoc](#) doc, [ASInt32](#) pageNum) ;

Description

Gets a **PDPPage** from a document. Increments the page's reference count. After you are done using the page, release it using [PDPPageRelease](#).

Parameters

doc	The document containing the page to acquire.
pageNum	The page number of the page to acquire. The first page is 0.

Return Value

The acquired page.

Known Exceptions

[genErrBadParm](#)

Header File

PDCalls.h

Related Methods

[PDPPageRelease](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDDocAddThread

```
void PDDocAddThread (PDDoc doc, ASInt32 addAfterIndex,
PDThread thread);
```

Description

Adds an article thread to a document after the specified thread index.

Parameters

doc	The document in which the thread is added. Must match the document used in the call to PDThreadNew that created the thread.
addAfterIndex	The index of the thread after which thread is added.
thread	The thread to add.

Return Value

None

Notifications

PDDocDidAddThread

Header File

PDCalls.h

Related Methods

PDThreadNew
PDThreadFromCosObj
PDDocRemoveThread

Product

base, pro, pdf

Availability

Available if PI_PDMODEL_VERSION (in PIRequir.h) is set to 0x00020000 or higher.

PDDocAuthorize

```
PDPperms PDDocAuthorize (PDDoc pdDoc, PDPperms permsWanted,
void* authData);
```

Description

Adds permissions to the specified document, if permitted. Calls the [PDCryptAuthorizeProc](#) callback of the document's security handler to determine which of the specified permissions will actually be granted. After calling this method, the document's permissions will be the OR of the previous permissions and the permissions granted by the [PDCryptAuthorizeProc](#) callback.

Use [PDDocPermRequest](#) to determine if a document's permissions allow a particular operation for a particular object.

Parameters

pdDoc	The document for which new permissions are requested.
permsWanted	The new permissions being requested. Must be an OR of the PDPperms values.
authData	Pointer to data to pass to the PDCryptAuthorizeProc callback of the document's security handler. For the Acrobat viewer's built-in security handler, authData is a char* containing the password.

Return Value

The OR of the previous value of the document's permissions field, and the permissions granted by the [PDCryptAuthorizeProc](#) callback of the document's security handler. The result will be an OR of the [PDPperms](#) values.

Known Exceptions

Raises [pdErrNeedCryptHandler](#) if no security handler is associated with **pdDoc**.

Raises whatever exceptions are raised by the security handler's [PDCryptAuthorizeProc](#) callback.

Header File

PDcalls.h

Related Methods

[PDDocGetNewCryptHandler](#)
[PDDocGetPermissions](#) (obsolete)
[PDDocPermRequest](#)
[PDDocSetNewCryptHandler](#)
[PDDocSetNewCryptHandlerEx](#)
[PDRegisterCryptHandler](#)



PDRegisterCryptHandlerEx

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDDocCalculateImplicitMetadata

```
void PDDocCalculateImplicitMetadata (PDDoc pdDoc);
```

Description

Notifies all registered implicit metadata calculators to run.

Issues the notification [PDDocCalculateMetadata](#), passing **pdDoc** to all registered handlers.

Parameters

pdDoc	The document for which implicit metadata is calculated.

Return Value

None

Notifications

[PDDocCalculateMetadata](#)

Header File

PDMetadataCalls.h

Related Methods

[PDDocSave](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMETADATA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDDocClearFlags

```
void PDDocClearFlags (PDDoc doc, ASInt32 flags);
```

Description

Clears flags associated with a document. This method is most frequently used to mark a modified document as clean (by clearing the **PDDocNeedsSave** flag) to avoid bringing up the Save dialog when the file is closed.

Parameters

doc	The document whose flags are cleared.
flags	The flags to clear. Must be an OR of the PDDocFlags values.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocSave](#)
[PDDocClose](#)
[PDDocGetFlags](#)
[PDDocSetFlags](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020001** or higher.



PDDocClose

```
void PDDocClose (PDDoc doc);
```

Description

Closes a document and releases its resources. If **doc** is **NULL**, does nothing. Changes are not saved. You must use [PDDocSave](#) to save any modifications before calling **PDDocClose**.

If the document has been modified but you wish to mark it as clean, use [PDDocClearFlags](#).

Parameters

doc	The document to close.

Return Value

None

Known Exceptions

Raises [pdErrUnableToCloseDueToRefs](#) if there are any outstanding references to objects in the document, and the document will still be valid (its resources will not be released).

Raises [genErrBadUnlock](#) if the document's open count is less than one.

Header File

PDCalls.h

Related Methods

[PDDocSave](#)
[PDDocOpen](#)
[PDDocCreate](#)
[PDDocClearFlags](#)
[PDDocSetFlags](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDDocCopyToFile

```
void PDDocCopyToFile (PDDoc doc, PDDocCopyParams params);
```

Description

In the Adobe Reader or for documents that are not dirty, this method copies the bytes from the document's **ASFile** to the specified location. This also occurs if the **saveChanges** field in **params** is **false**. If **saveChanges** is **true**, the document is dirty, and the product is Acrobat, a full save is performed to the specified file. The resulting file is linearized. If the file already exists, it is overwritten.

Parameters

doc	The document to copy.
params	A structure that defines how the PDF file is copied.

Return Value

None

Known Exceptions

Raises [genErrBadParm](#) if an invalid argument is passed in **params**.

Raises [pdErrAlreadyOpen](#) if the target and source files are the same.

Raises [fileErrDiskFull](#) if there is no space for the copy.

Raises [pdErrCancelSave](#) if the save was canceled (**cancelProc** in **params** returned **true**).

Raises [pdErrUnableToRead](#) if a read error occurred on the source.

Raises [pdErrUnableToWrite](#) if a write error occurred on the destination.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040005** or higher.



PDDocCreate

```
PDDoc PDDocCreate (void);
```

Description

Creates a new document. The only Cos object in the document will be a Catalog. See Section 3.6 in the [PDF Reference](#). After the document is created, at least one page must be added using [PDDocCreatePage](#) or [PDDocInsertPages](#) before the Acrobat viewer can display or save the document.

When you are done with the document, you must call [PDDocClose](#) to release the resources used by the **PDDoc**; do *not* call [PDDocRelease](#).

Parameters

None

Return Value

The newly created document.

Header File

PDCalls.h

Related Methods

[PDDocClose](#)
[PDDocOpen](#)
[PDDocSave](#)
[PDDocCreatePage](#)
[PDDocInsertPages](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDDocCreateNameTree

```
PDNameTree PDDocCreateNameTree (PDDoc thePDDoc,
ASAtom theTree);
```

Description

Retrieves the name tree inside the Names dictionary with the specified key name, or creates it if it does not exist.

Parameters

thePDDoc	The document in which the name tree is created.
theTree	The name of the name tree to create. A string can be converted to an ASAtom using ASAtomFromString.

Return Value

The newly created PDNameTree for the PDDoc. Returns a NULL PDNameTree if pdDoc has no root dictionary. The return value should be tested with PDNameTreeIsValid.

Header File

```
PDCalls.h
```

Related Methods

```
PDNameTreeIsValid
PDDocGetNameTree
PDDocRemoveNameTree
```

Example

```
PDNameTree pdNameTree;
PDDoc thePDDoc;

pdNameTree = PDDocCreateNameTree (thePDDoc, ASAtomFromString
("docNameTree"));
```

Product

```
base, pro, pdf
```

Availability

Available if PI_PDMODEL_VERSION (in PIRequir.h) is set to 0x00040000 or higher.

PDDocCreatePage

```
PDPAGE PDDocCreatePage (PDDOC doc, ASInt32 afterPageNum,
ASFixedRect mediaBox);
```

Description

Creates and acquires a new page. The page is inserted into the document at the specified location. Call **PDPAGERelease** when you are done using the page.

Parameters

doc	The document in which the page is created.
afterPageNum	The page number after which the new page is inserted. The first page is 0. Use PDBeforeFirstPage (see PDExpT.h) to insert the new page at the beginning of a document.
mediaBox	Rectangle specifying the page's media box, specified in user space coordinates.

Return Value

The newly created page.

Notifications

PDDocWillInsertPages
PDDocDidInsertPages
PDDocDidChangePages
PDDocDidChangeThumbs

Header File

PDCalls.h

Related Methods

PDPAGERelease
PDDocDeletePages
PDDocInsertPages
PDDocReplacePages

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDDocCreateStructTreeRoot

```
void PDDocCreateStructTreeRoot (PDDoc pdDoc,  
                                PDSTreeRoot* treeRoot);
```

Description

Creates a new **StructTreeRoot** element.

If **PDDocCreateStructTreeRoot** is called on a **PDDoc** that already has a structure tree root, it returns without modifying the document.

Parameters

pdDoc	The PDDoc for which StructTreeRoot element is created.
treeRoot	<i>(Filled by the method)</i> The newly created StructTreeRoot element.

Return Value

None

Known Exceptions

Raises an exception if **pdDoc** already has a **StructTreeRoot**.

Header File

PDSWriteCalls.h

Related Methods

[PDDocGetStructTreeRoot](#)
[PDSTreeRootGetRoleMap](#)
[PDSTreeRootGetClassMap](#)
[PDDocRemoveStructTreeRoot](#)
[PDSTreeRootCreateRoleMap](#)
[PDSTreeRootRemoveRoleMap](#)
[PDSTreeRootCreateClassMap](#)
[PDSTreeRootRemoveClassMap](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDDocCreateTextSelect

```
PDTextSelect PDDocCreateTextSelect (PDDoc doc,
ASInt32 pageNum, ASFixedRect* boundingRect);
```

Description

Creates a text selection that includes all words totally or partially enclosed by a rectangle. The text selection can then be set as the current selection using [AVDocSetSelection](#).

NOTE: When this method is used to create a text selection on a rotated page, you must pass in a rotated **boundingRect**.

Parameters

doc	The document in which a text selection is created.
pageNum	The page number on which the text selection is created.
boundingRect	Pointer to a rectangle specifying the text selection's bounding rectangle, specified in user space coordinates.

Return Value

The newly created text selection.

Header File

PDCalls.h

Related Methods

[PDTextSelectDestroy](#)
[AVDocSetSelection](#)
[PDTextSelectEnumQuads](#)
[PDTextSelectEnumText](#)
[PDWordCreateTextSelect](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to 0x00020000 or higher.

PDDocCreateThumbs

```
void PDDocCreateThumbs (PDDoc doc, ASInt32 firstPage,
ASInt32 lastPage, PDThumbCreationServer server,
void* serverClientData, ASAtom colorSpace,
ASInt32 bitsPerComponent, ASInt32 hiVal, char* lookupTable,
ProgressMonitor progMon, void* progMonClientData,
CancelProc cancelProc, void* cancelProcClientData);
```

Description

Creates thumbnail images for the specified range of pages. Thumbnail images are only created for pages that have none.

Use as large of a page range as possible because the color space object is shared by all the thumbnails created by a single invocation of this method (that is, if you call this method separately for each page, there will be duplicate color space objects).

See Section 4.5 in the [PDF Reference](#) for more details on color spaces.

See the Windows chapter of Technical Note #5167, [Acrobat Development Overview](#), for additional important information about creating thumbnails on the Windows platform.

Parameters

doc	The document for which thumbnail images are created.
firstPage	The page number of the first page for which thumbnails are created. The first page is 0.
lastPage	The page number of the last page for which thumbnails are created. The constant PDLastPage (see <code>PDExpT.h</code>) can also be used.
server	A server (set of callback procedures) that provides the sampled image used as the thumbnail image. Pass NULL to use the default server.
serverClientData	User-supplied data to pass to the thumbnail creation server.

colorSpace	The color space in which the thumbnail data is represented. It must be DeviceRGB. Thumbnails may be created in either a direct or an indexed color space; however, it is strongly recommended that you use indexed color spaces over direct color spaces. Using direct color spaces with this version of Acrobat may cause bad looking thumbnails. To specify a direct color space, pass 0 for hiVal and NULL for lookupTable. To specify an indexed color space, pass the appropriate values in hiVal and lookupTable. Direct color spaces in Windows are supported in Acrobat 3.0. Prior to Acrobat 3.0 in Windows, you had to use indexed color spaces.
bitsPerComponent	The number of bits per color component in the thumbnail image's data. 8 is the only valid value.
hiVal	Used only for indexed color space; pass 0 for direct color spaces, as described in colorSpace. hiVal specifies the highest valid index in lookupTable. Because indices start at 0, the number of entries in lookupTable is hiVal + 1. hiVal must be 0 for device color spaces.
lookupTable	Used only for indexed color space; pass NULL for direct color spaces, as described in colorSpace. lookupTable is a table that maps data values to colors. Used only for indexed color spaces. Must be NULL for device color spaces. For an indexed color space, the size of the lookup table must be $(\text{hiVal} + 1) \times \text{sizeof}(\text{ASUns8}) \times 3$, where the 3 arises because an RGB color space has three color components.
progMon	A monitor to call to display thumbnail creation progress. Use AVAppGetDocProgressMonitor to obtain the standard progress monitor to pass for this parameter. NULL may be passed, in which case no progress monitor is used.
progMonClientData	User-supplied data to pass to progMon each time it is called. Should be NULL if progMon is NULL.
cancelProc	A procedure to call frequently to allow the user to cancel thumbnail creation. Use AVAppGetCancelProc to obtain the default cancel proc for this parameter. May be NULL, in which case no cancel proc is used.
cancelProcClientData	User-supplied data to pass to cancelProc each time it is called. Should be NULL if cancelProc is NULL.

**Return Value**

None

Notifications

[PDDocDidChangeThumbs](#)

Header File

PDCalls.h

Related Methods

[PDDocDeleteThumbs](#)

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDDocCreateWordFinder

```
PDWordFinder PDDocCreateWordFinder (PDDoc doc,
ASUns16* outEncInfo, char** outEncVec, char** ligatureTbl,
ASInt16 algVersion, ASUns16 rdFlags, void* clientData);
```

Description

Creates a word finder that is used to extract text in the host encoding from a PDF file. The word finder may either be used by [PDWordFinderEnumWords](#) (which enumerates words one-by-one) or by [PDWordFinderAcquireWordList](#) (which fills a table with all the words on a page).

NOTE: The word finder also extracts text from Form XObjects that are executed in the page contents. For information about Form XObjects, see Section 4.9 in the [PDF Reference](#).

A default ligature table is used, containing the following ligatures: fi, ff, fl, ffi, ffl, ch, cl, ct, ll, ss, fs, st, oe, OE. The glyph name is substituted for the ligature.

This method also works for non-Roman (CJK or Chinese-Japanese-Korean) viewers. In this case, words are extracted to the host encoding. Users desiring Unicode output must use [PDDocCreateWordFinderUCS](#), which does the extraction for Roman or non-Roman text.

The type of [PDWordFinder](#) determines the encoding of the string returned by [PDWordGetString](#). For instance, if [PDDocCreateWordFinderUCS](#) is used to create the word finder, [PDWordGetString](#) returns only Unicode.

For CJK viewers, words are stored internally using CID encoding. For more information on CIDFonts and related topics, see Section 5.6 in the [PDF Reference](#). For detailed information on CIDFonts, see Technical Note #5092, *CID-Keyed Font Technology Overview*, and Technical Note #5014, *Adobe CMap and CIDFont Files Specification*.

Parameters

doc	The document on which the word finder is used.
outEncInfo	Array of 256 flags, specifying the type of character at each position in the encoding. Each flag is an OR of the Character Type Codes . If outEncInfo is NULL , the platform's default encoding info is used. Use outEncInfo and outEncVec together; for every outEncInfo use a corresponding outEncVec to specify the character at that position in the encoding. Regardless of the characters specified in outEncInfo as word separators, a default set of word separators is used (see Glyph Names of Word Separators). There is no way to change the list of characters that are considered to be word separators.

outEncVec	Array of 256 null-terminated strings that are the glyph names in encoding order. See the discussion of character names in Section 5.3 of the <i>PostScript Language Reference Manual, Third Edition</i>. If outEncVec is NULL, the platform's default encoding vector is used. For non-UNIX Roman systems, it is MacRomanEncoding in Mac OS and WinAnsiEncoding in Windows. In UNIX (except HP-UX) Roman systems, it is ISO8859-1 (ISO Latin-1); for HP-UX, it is HP-ROMAN8. See Appendix D in the PDF Reference for descriptions of MacRomanEncoding, WinAnsiEncoding, and PDFDocEncoding. Use this parameter with outEncInfo. See outEncInfo for more information.
ligatureTbl	A null-terminated array of null-terminated strings. Each string is the glyph name of a ligature in the font. When a word contains a ligature, the glyph name of the ligature is substituted for the ligature (for example, ff is substituted for the ff ligature). This table must be terminated with NULL. If ligatureTbl is NULL, a default ligature table is used, containing the following ligatures: fi, ff, fl, ffi, ffl, ch, cl, ct, ll, ss, fs, st, oe, OE.
algVersion	The version of the word-finding algorithm to use (see <code>PDExpT.h</code>), as follows (pass 0 if your client does not care): WF_LATEST_VERSION—To obtain latest available version. WF_VERSION_2—Version used for Acrobat 3.x, 4.x. WF_VERSION_3—Available in Acrobat 5.0 without Accessibility enabled. Includes some improved word piecing algorithms. WF_VERSION_4—For Acrobat 5.0 with Accessibility enabled. Includes advanced word ordering algorithms in addition to improved word piecing algorithms.
rdFlags	Word-finding options that determine the tables filled when using PDWordFinderAcquireWordList. Must be an OR of one or more of the WordFinder Sort Order Flags. In Acrobat 5.0 this parameter is ignored and you should pass in NULL.
clientData	Pointer to user-supplied data to pass to the newly created word finder.

Return Value

The newly created word finder.

Header File

`PDCalls.h`

Related Methods

[PDDocCreateWordFinderUCS](#)
[PDWordFinderEnumWords](#)
[PDWordFinderAcquireWordList](#)
[PDWordFinderDestroy](#)
[PDWordFilterWord](#)

Example

```

ACCB1 ASBool ACCB2 WordCounterProc (
    PDWordFinder WordFinder, PDWord word,
    ASInt32 PageNum, void* fileNum)
{
    ASUns32 cur;

    cur = (ASUns32) PDWordGetCharOffset(word);
    WordCnt += (cur - gLast);
    gLast = cur;
    return true;
}

WordFinder = PDDocCreateWordFinder(
    CurrentPDDoc, NULL, NULL, NULL, 0,
    WXE_PDF_ORDER, NULL);
PDWordFinderEnumWords(WordFinder, PageNum,
    ASCallbackCreateProto(PDWordProc,
    &WordCounterProc), NULL);
PDWordFinderDestroy(WordFinder);

```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDDocCreateWordFinderEx

```
PDWordFinder PDDocCreateWordFinderEx (PDDoc doc,
ASInt16 algVersion, ASBool outUnicode,
PDWordFinderConfig wbConfig);
```

Description

This is a version 6.0 replacement for [PDDocCreateWordFinder](#) and [PDDocCreateWordFinderUCS](#) that adds configurable word-breaking behavior. This method creates a word finder that is used to extract text from a PDF file, according to the given configuration. The word finder can be used to enumerate words one-by-one or to fill a table with all the words on a page. You can choose to find only words that are visible in a given context.

You can use version 6.0 methods such as [PDWordGetCharOffsetEx](#) to extract character information from words if you create the word finder with **WF_VERSION_3** or later.

NOTE: The word finder also extracts text from Form XObjects that are executed in the page contents. For information about Form XObjects, see Section 4.9 in the [PDF Reference](#).

Parameters

doc	The document on which the word finder is used.
algVersion	The version of the word-finding algorithm to use (see <code>PDExpT.h</code>), as follows (pass 0 if your client does not care): WF_LATEST_VERSION —To obtain latest available version. WF_VERSION_2 —Version used for Acrobat 3.x, 4.x. WF_VERSION_3 —Available in Acrobat 5.0 and 6.0 without Tagged PDF support. WF_VERSION_4 —For Acrobat 5.0 and 6.0 with Tagged PDF support.
outUnicode	Whether to return Unicode. When true , the word finder encodes the extracted text in Unicode format. Otherwise, the word finder extracts the text in the host encoding.
wbConfig	Pointer to a configuration record for the new word finder that customizes the way the extraction is performed. The configuration is only used if the algorithm version is WF_VERSION_3 or higher. When NULL , the default configuration is used.

Return Value

The newly created word finder object.

**Header File**

PDCalls.h

Related Methods

[PDDocCreateWordFinder](#)
[PDDocCreateWordFinderUCS](#)
[PDWordFinderEnumWords](#)
[PDWordFinderAcquireWordList](#)
[PDWordFinderDestroy](#)
[PDWordFilterWord](#)
[PDWordGetCharOffsetEx](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDDocCreateWordFinderUCS

```
PDWordFinder PDDocCreateWordFinderUCS (PDDoc doc,  
ASInt16 algVersion, ASUns16 rdFlags, void* clientData);
```

Description

Creates a word finder that is used to extract text in Unicode format from a PDF file. The word finder may either be used by **PDWordFinderEnumWords** (which enumerates words one-by-one) or by **PDWordFinderAcquireWordList** (which fills a table with all the words on a page).

NOTE: The word finder also extracts text from Form XObjects that are executed in the page contents. For information about Form XObjects, see Section 4.9 in the [PDF Reference](#).

NOTE: **PDDocCreateWordFinderUCS** is useful for converting non-Roman text (CJK or Chinese-Japanese-Korean) to Unicode. This method also converts Roman text to Unicode in any document.

PDDocCreateWordFinder also works for non-Roman character set viewers. For **PDDocCreateWordFinder**, words are extracted to the host encoding. Users desiring Unicode output should use **PDDocCreateWordFinderUCS**.

The type of **PDWordFinder** determines the encoding of the string returned by **PDWordGetString**. If **PDDocCreateWordFinderUCS** is used to create the word finder, **PDWordGetString** returns only Unicode.

NOTE: There is no way to detect Unicode strings returned by **PDWordGetString**, since there is no UCS header (FEFF) added to each string returned.

In CJK viewers, words are stored internally using CID encoding. For more information on CIDFonts and related topics, see Section 5.6 in the [PDF Reference](#). For detailed information on CIDFonts, see Technical Note #5092, *CID-Keyed Font Technology Overview*, and Technical Note #5014, *Adobe CMap and CIDFont Files Specification*.

Parameters

doc	The document on which the word finder is used.
algVersion	The version of the word-finding algorithm to use. If WF_LATEST_VERSION (see <code>PDExpT.h</code>), the most recent version is used. Set to 0 if your client does not care.
rdFlags	Word-finding options that determine the tables filled when using PDWordFinderAcquireWordList . Must be an OR of one or more of the WordFinder Sort Order Flags .
clientData	Pointer to user-supplied data to pass to the newly created word finder.

Return Value

The newly created word finder.

Header File

PDCalls.h

Related Methods

[PDDocCreateWordFinder](#)
[PDWordFinderEnumWords](#)
[PDWordFinderAcquireWordList](#)
[PDWordFinderDestroy](#)
[PDWordFilterWord](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020003` or higher.

PDDocDeletePages

```
void PDDocDeletePages (PDDoc doc, ASInt32 firstPage,
ASInt32 lastPage, ProgressMonitor progMon,
void* progMonClientData);
```

Description

Deletes the specified pages, inclusively.

Parameters

doc	The document from which pages are deleted.
firstPage	The page number of the first page to delete. The first page is 0.
lastPage	The page number of the last page to delete.
progMon	A progress monitor. Use AVAppGetDocProgressMonitor to obtain the default progress monitor. NULL may be passed, in which case no progress monitor is used.
progMonClientData	Pointer to user-supplied data passed to mon each time it is called. Should be NULL if progMon is NULL .

Return Value

None

Notifications

[PDDocWillChangePages](#)
[PDDocWillDeletePages](#)
[PDDocDidDeletePages](#)
[PDDocDidChangePages](#)

Header File

PDCalls.h

Related Methods

[PDDocInsertPages](#)
[PDDocReplacePages](#)
[PDDocMovePage](#)

Product

base, pro, pdf



Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDDocDeleteThumbs

```
void PDDocDeleteThumbs (PDDoc doc, ASInt32 firstPage,
ASInt32 lastPage, ProgressMonitor progMon,
void* progMonClientData);
```

Description

Deletes thumbnail images for a range of pages in a document.

Parameters

doc	The document from which thumbnail images are deleted.
firstPage	The page number of the first page in doc whose thumbnail image is deleted. The first page is 0.
lastPage	The page number of the last page in doc whose thumbnail image is deleted.
progMon	A monitor to call to display thumbnail deletion progress. Use AVAppGetDocProgressMonitor to obtain the standard progress monitor to pass for this parameter. NULL may be passed, in which case no progress monitor is used.
progMonClientData	Pointer to user-supplied data to pass to progMon. Should be NULL if progMon is NULL.

Return Value

None

Notifications

[PDDocDidChangeThumbs](#)

Header File

PDCalls.h

Related Methods

[PDDocCreateThumbs](#)

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDDocEnumFonts

```
void PDDocEnumFonts (PDDoc doc, ASInt32 firstPage,
ASInt32 lastPage, PDFFontEnumProc proc, void* clientData,
ProgressMonitor progMon, void* progMonClientData);
```

Description

Enumerates all the fonts in the specified page range. This may take a considerable amount of time for a large page range.

Parameters

doc	The document whose fonts are enumerated.
firstPage	The page number of the first page for which fonts are enumerated. The first page is 0.
lastPage	The page number of the last page for which fonts are enumerated.
proc	User-supplied callback to call for each font. Enumeration terminates if proc returns false .
clientData	Pointer to user-supplied data to pass to proc each time it is called.
progMon	A progress monitor. Use AVAppGetDocProgressMonitor to obtain the standard progress monitor. NULL may be passed, in which case no progress monitor is used.
progMonClientData	Pointer to user-supplied data to pass to progMon each time it is called. Should be NULL if progMon is NULL .

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocEnumLoadedFonts](#)
[PDFFontEnumCharProcs](#)

Product

reader, base, pro, pdf

**Availability**

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDDocEnumLoadedFonts

```
void PDDocEnumLoadedFonts (PDDoc doc, PDFontEnumProc proc,
void* clientData);
```

Description

Enumerates all the fonts that have been encountered so far. A font is loaded when a page that uses it is processed. This typically happens when a page is drawn or its thumbnail image is created.

Parameters

doc	The document whose loaded fonts are enumerated.
proc	User-supplied callback to call for each loaded font. Enumeration terminates if proc returns false .
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Header File

PDCalls.h

Related Methods

- [PDDocEnumFonts](#)
- [PDFontEnumCharProcs](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDDocEnumOCConfigs

```
void PDDocEnumOCConfigs (PDDoc pdDoc,  
PDOCConfigEnumProc enumProc, void *clientData);
```

Description

Enumerates the optional-content configurations for the document, calling the supplied procedure for each one. These include the configuration for the **D** configuration dictionary and those for all entries in the **Configs** array dictionary.

Parameters

pdDoc	The document whose configurations are enumerated.
enumProc	User-supplied callback to call for each configuration. Enumeration terminates if proc returns false .
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocGetOCConfig](#)
[PDDocEnumOCGs](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDDocEnumOCGs

```
void PDDocEnumOCGs (PDDoc pdDoc, PDOCGEnumProc enumProc,
void* clientData);
```

Description

Enumerates the optional-content groups for the document, calling the supplied procedure for each one. Enumeration continues until all groups have been enumerated, or until **enumProc** returns **false**. Each group is reported once, even if it is referenced multiple times in a page, or on multiple pages.

Parameters

pdDoc	The document whose groups are enumerated.
enumProc	User-supplied callback to call for each group. Enumeration terminates if proc returns false .
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocGetOCGs](#)
[PDDocEnumOCConfigs](#)
[PDPageEnumOCGs](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDDocEnumResources

```
void PDDocEnumResources (PDDoc pdDoc, ASInt32 startPage,
ASInt32 endPage, ASAtom resourceType, CosObjEnumProc enumProc,
void* clientData);
```

Description

Enumerates the specified type of page resources, for a specified range of pages.

This method enumerates resources in each page’s **Resources** dictionary (ColorSpace resources, Fonts, ExtGState objects, or others). In addition, it looks inside in-line images and page contents to enumerate ColorSpace resources that are not in the **Resources** dictionary, such as **DeviceGray**, **DeviceRGB**, and **DeviceCMYK**.

Parameters

pdDoc	The document whose resources are enumerated.
startPage	The first page whose resources are enumerated. (The first page in a document is 0.)
endPage	The last page whose resources are enumerated.
resourceType	Resource type to enumerate. Must be one of the valid PDF resource types, such as Font , ColorSpace , XObject , Pattern , and so on, described in Section 3.7 in the PDF Reference . Pass ASAtomNull to enumerate all resource types.
enumProc	User-supplied callback to call once for each resource of the specified type. The resource is presented as a CosObj , and it is the first parameter of enumProc (the second parameter is unused).
clientData	User-supplied data to pass to enumProc each time it is called.

Return Value

None

Known Exceptions

- [genErrBadParm](#)
- [pdErrOpNotPermitted](#)

Header File

PDCalls.h

Related Methods

- [PDDocEnumFonts](#)
- [PDEEnumElements](#)



PDELogDump
PDEObjectDump

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.

PDDocExportNotes

```
CosDoc PDDocExportNotes (PDDoc sourceDoc, ASFileSys fileSys,
ASPathName path, void* progMon, void* monClientData,
PDDocWillExportAnnotCallback exportFilter,
ASInt32* numNotesP);
```

Description

Creates a document containing empty pages plus text annotations (notes) from **sourceDoc**. Does not create a new document if **sourceDoc** contains no notes.

Parameters

sourceDoc	The document from which notes are exported.
fileSys	Currently unused.
path	Currently unused.
progMon	Currently unused.
monClientData	Currently unused.
exportFilter	User-supplied routine that selects which notes to export.
numNotesP	If non- NULL , the number of notes exported.

Return Value

The **CosDoc** of the document created to hold the exported notes.

Notifications

PDDocWillExportAnnots
PDDocDidExportAnnots

Header File

PDCalls.h

Related Methods

PDDocImportCosDocNotes
PDDocImportNotes
PDDocExportSomeNotes

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDDocExportSomeNotes

```
CosDoc PDDocExportSomeNotes (PDDoc doc, ASFileSys unused1,
ASPathName unused2, void* unused3, void* unused4,
PDDocWillExportAnnotCallback exportFilter, PDAnnotArray
pdanArray, ASInt32* numNotesP);
```

Description

Like [PDDocExportNotes](#) but the caller provides the list of annots to export. This is useful in scenarios when it may be inappropriate to use [PDDocExportNotes](#) and look for annots on every page. This is an especially important consideration when in a browser.

NOTE: Make sure to explicitly include popups.

Parameters

doc	The document from which notes are exported.
unused1	Currently unused.
unused2	Currently unused.
unused3	Currently unused.
unused4	Currently unused.
exportFilter	User-supplied routine that selects which notes to export.
pdanArray	An array of PDAnnotArrayRecs ; the number of items in the array is the number of pages in doc .
numNotesP	(Filled by the method) If non- NULL , the number of notes exported.

Return Value

The **CosDoc** of the document created to hold the exported notes.

Notifications

[PDDocWillExportAnnots](#)
[PDDocDidExportAnnots](#)

Header File

PDCalls.h

Related Methods

[PDDocGetXAPMetadata](#)
[PDDocImportNotes](#)
[PDDocExportNotes](#)



Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDDocFindPageNumForLabel

```
ASInt32 PDDocFindPageNumForLabel (PDDoc pdDoc,
const char* labelStr, ASInt32 labelLen);
```

Description

Finds the first page in the document with a specified label.

NOTE: Superseded by [PDDocFindPageNumForLabelEx](#) in Acrobat 6.0.

Parameters

pdDoc	The document to search for the page named in labelStr .
labelStr	Label of the page to find.
labelLen	Length of labelStr .

Return Value

The page number of the first page with the specified label, or -1 if no such page exists.

Header File

PDCalls.h

Related Methods

[PDDocGetLabelForPageNum](#)
[PDDocGetPageLabel](#)
[PDDocRemovePageLabel](#)
[PDDocSetPageLabel](#)
[PDPageLabelNew](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDDocFindPageNumForLabelEx

```
ASInt32 PDDocFindPageNumForLabelEx (PDDoc pdDoc,
ASConstText labelText);
```

Description

Finds the first page in the document with a specified label.

NOTE: Supersedes [PDDocFindPageNumForLabel](#) in Acrobat 6.0.

Parameters

pdDoc	The document to search for the page named in labelStr .
labelText	Label of the page to find.

Return Value

The page number of the first page with the specified label, or -1 if no such page exists.

Header File

PDCalls.h

Related Methods

[PDDocGetLabelForPageNumEx](#)
[PDDocGetPageLabel](#)
[PDDocRemovePageLabel](#)
[PDDocSetPageLabel](#)
[PDPageLabelNew](#)

Product

reader, base, pro

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDDocFlattenOC

ASBool PDDocFlattenOC (**PDDoc** pdDoc, **PDOCContext** context)

Description

Replaces the contents of every page in the document with a version that has no optional content, containing only what was visible on the page when the call was made, and removes all other optional-content information.

Parameters

pdDoc	The document to be modified.
context	The optional-content context in which content is checked for visibility.

Return Value

true if the operation is successful, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDPageFlattenOC](#)

[PDEContentFlattenOC](#)

PDDocFromCosDoc

```
PDDoc PDDocFromCosDoc (CosDoc cosDoc) ;
```

Description

Gets the **PDDoc** associated with a **CosDoc**.

Parameters

cosDoc	The Cos-level document object for which a PDDoc is obtained. This object represents the PDF.

Return Value

The **PDDoc** associated with **cosDoc**.

Known Exceptions

Raises **genErrBadParm** if the **CosDoc** is not valid.

Raises **pdErrNoPDDocForCosDoc** if there is no **PDDoc** associated with this **cosDoc**.

Header File

PDCalls.h

Related Methods

[AVDocGetPDDoc](#)
[PDPageGetDoc](#)
[PDDocGetCosDoc](#)

Example

```
CosDoc cosDoc;
PDDoc pdDoc;

pdDoc = PDDocFromCosDoc (cosDoc);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDDocGetBookmarkRoot

```
PDBookmark PDDocGetBookmarkRoot (PDDoc pdDoc) ;
```

Description

Gets the root of the document's bookmark tree. The return value is valid even if document's bookmark tree is empty (that is, even if there is no **Outlines** key in the underlying PDF file).

Parameters

pdDoc	Document whose root bookmark is obtained.

Return Value

The document's root bookmark.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDDocGetCosDoc

```
CosDoc PDDocGetCosDoc (PDDoc doc);
```

Description

Gets a document's Cos-level document object.

Parameters

doc	The document whose CosDoc is obtained.

Return Value

The document's **CosDoc**.

Header File

PDCalls.h

Related Methods

[CosObjGetDoc](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDDocGetCryptHandlerClientData

```
void* PDDocGetCryptHandlerClientData (PDDoc doc);
```

Description

Gets the client data for the encryption handler associated with the **PDDoc**. This is the client data provided as a parameter in [PDRegisterCryptHandlerEx](#).

Parameters

doc	The document whose encryption handler client data is obtained.

Return Value

Client data for the encryption handler associated with the **PDDoc**. Returns **NULL** if there is no encryption handler or no client data was provided.

Header File

PDCalls.h

Related Methods

[PDRegisterCryptHandlerEx](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.



PDDocGetFile

```
ASFile PDDocGetFile (PDDoc doc);
```

Description

Gets the file object for a document.

Parameters

doc	The document whose ASFile is obtained.

Return Value

The document's **ASFile**.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDDocGetFlags

```
ASInt32 PDDocGetFlags (PDDoc doc);
```

Description

Gets information about the document's file and its state.

Parameters

<code>doc</code>	The document whose flags are obtained.

Return Value

Flags field, containing an OR of the [PDDocFlags](#) values.

Header File

`PDCalls.h`

Related Methods

[PDDocSetFlags](#)
[PDDocClearFlags](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDDocGetFullScreen

```
ASBool PDDocGetFullScreen (PDDoc pdDoc);
```

Description

Tests whether the document will open in full-screen mode. This provides an alternative to calling [PDDocGetPageMode](#) to test for **PDFFullScreen**.

Parameters

pdDoc	The document to test.

Return Value

Returns **true** if the **PDDoc** is in full-screen mode, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDDocSetFullScreen](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

PDDocGetID

```
ASInt32 PDDocGetID (PDDoc doc, ASInt32 nElemNum,
ASUns8* buffer, ASInt32 bufferSize);
```

Description

Gets an element of a document's file identifier. See Section 10.3 in the [PDF Reference](#) for a description of file IDs.

Parameters

doc	The document whose file ID is obtained.
nElemNum	The element number to get from the document's file ID. Must be one of the following: 0 — The permanent ID 1 — The changing ID
buffer	<i>(Filled by the method)</i> If buffer is non- NULL , then up to the bufferSize bytes of the ID will be written to the buffer.
bufferSize	Length of buffer , in bytes.

Return Value

The number of bytes in the ID element.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDDocGetInfo

```
ASInt32 PDDocGetInfo (PDDoc doc, const char* infoKey,
char* buffer, ASInt32 bufSize);
```

Description

Gets the value of a key in a document's Info dictionary, or the value of this same key in the XMP metadata, whichever is latest. However, it is preferable to use [PDDocGetXAPMetadataProperty](#), because it also allows accessing XMP properties that are not duplicated in the Info dictionary.

See Section 10.2.1 in the [PDF Reference](#) for information about Info dictionaries. All values in the Info dictionary should be strings; other data types such as numbers and booleans should not be used as values in the Info dictionary.

Users may define their own Info dictionary entries. In this case, it is strongly recommended that the key have the developer's prefix assigned by the Adobe Solutions Network.

NOTE: For Roman viewers, this text is always stored in the **PDFDocEncoding**. For non-Roman character set viewers, this text is stored as **PDFDocEncoding** or Unicode, depending on the file's creator.

Parameters

doc	The document whose Info dictionary key is obtained.
infoKey	The name of the Info dictionary key whose value is obtained.
buffer	<i>(Filled by the method)</i> Buffer containing the value associated with infoKey . If buffer is NULL , the method will just return the number of bytes required.
bufSize	The maximum number of bytes that can be written into buffer .

Return Value

If **buffer** is **NULL**, the number of bytes in the specified key's value. If **buffer** is not **NULL**, returns the number of bytes copied into **buffer**, excluding the terminating **NULL**. You must pass at least the **length** + 1 as the buffer size since the routine adds a '\0' terminator to the data, even though the data is not a C string (it can contain embedded '\0's).

Header File

PDCalls.h

Related Methods

[PDDocGetXAPMetadataProperty](#)

[PDDocSetInfo](#)

**Product**

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDDocGetLabelForPageNum

```
ASInt32 PDDocGetLabelForPageNum (PDDoc pdDoc, ASInt32 pageNum,
char* buffer, ASInt32 bufferLen);
```

Description

Retrieves the label string associated with the given page number. The page number is returned in host encoding and is truncated to the length of the buffer.

NOTE: Superseded by [PDDocGetLabelForPageNumEx](#) in Acrobat 6.0.

Parameters

pdDoc	Document containing the page for which a label is requested.
pageNum	The number of the page whose label is requested.
buffer	If a label exists for pageNum , it will be placed in this buffer.
bufferLen	Length of the label (null-terminated).

Return Value

The length of the resulting label. If no such page number exists, the resulting string will be the ASCII representation of **pageNum** + 1.

Header File

PDCalls.h

Related Methods

[PDDocGetLabelForPageNumEx](#)
[PDDocFindPageNumForLabel](#)
[PDDocGetPageLabel](#)
[PDDocRemovePageLabel](#)
[PDDocSetPageLabel](#)
[PDPageLabelNew](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDDocGetLabelForPageNumEx

```
void PDDocGetLabelForPageNumEx (PDDoc pdDoc, ASInt32 pageNum,
ASConstText text);
```

Description

Retrieves the label string associated with the given page number. The page number is returned in host encoding as a **ASText** object.

NOTE: Supersedes [PDDocGetLabelForPageNum](#) in Acrobat 6.0.

Parameters

pdDoc	Document containing the page for which a label is requested.
pageNum	The number of the page whose label is requested.
text	If a label exists for pageNum , it is returned in this object.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocFindPageNumForLabelEx](#)
[PDDocGetPageLabel](#)
[PDDocRemovePageLabel](#)
[PDDocSetPageLabel](#)
[PDPageLabelNew](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDDocGetNameTree

```
PDNameTree PDDocGetNameTree (PDDoc thePDDoc, ASAtom theTree);
```

Description

Retrieves a name tree, with the key name specified in **theTree**, from the Names dictionary of **thePDDoc**.

Parameters

thePDDoc	The document containing the name tree desired.
theTree	The name of the tree requested. This can be created by passing a string to the ASAtomFromString method.

Return Value

The **PDNameTree** requested.

Header File

PDCalls.h

Related Methods

[PDNameTreeIsValid](#)
[PDDocCreateNameTree](#)
[PDDocRemoveNameTree](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDDocGetNewCryptHandler

```
ASAtom PDDocGetNewCryptHandler (PDDoc doc);
```

Description

Gets the specified document's new security handler (that is, the security handler that will be used after the document is saved).

If the document does not have a new security handler, returns the document's current security handler.

Parameters

<code>doc</code>	The document whose new security handler is obtained.

Return Value

The **ASAtom** corresponding to the **pdfName** of the document's new security handler. Returns **ASAtomNull** if the document does not have a new security handler.

Header File

PDCalls.h

Related Methods

[PDDocPermRequest](#)
[PDDocSetNewCryptHandler](#)
[PDDocSetNewCryptHandlerEx](#)
[PDRegisterCryptHandler](#)
[PDRegisterCryptHandlerEx](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDDocGetNewSecurityData

```
void* PDDocGetNewSecurityData (PDDoc doc);
```

Description

Gets the security data structure for the specified document's new security handler. Use [PDDocGetSecurityData](#) to get the security data structure for the document's current security handler.

Parameters

doc	The document whose new security data structure is obtained.

Return Value

The security data structure for the document's new security handler.

Header File

PDCalls.h

Related Methods

[PDDocGetSecurityData](#)
[PDDocNewSecurityData](#)
[PDDocSetNewCryptHandler](#)
[PDDocSetNewSecurityData](#)
[PDDocPermRequest](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDDocGetNewSecurityInfo

```
void PDDocGetNewSecurityInfo (PDDoc pdDoc, ASUns32* secInfo);
```

Description

Gets the security information from the specified document's new security handler. Calls the [PDCryptGetSecurityInfoProc](#) callback of the document's new security handler. No permissions are required to call this method.

Parameters

pdDoc	The document whose new security information is obtained.
secInfo	<i>(Filled by the method)</i> The document's new security information. The value mst be an OR of the Security Info Flags . Set to pdInfoCanPrint pdInfoCanEdit pdInfoCanCopy pdInfoCanEditNotes (see PDPerms) if the document's new security handler does not have a PDCryptGetSecurityInfoProc callback.

Return Value

None

Known Exceptions

Raises only those exceptions raised by the new security handler's [PDCryptGetSecurityInfoProc](#) callback.

Header File

PDCalls.h

Related Methods

[PDRegisterCryptHandler](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDDocGetNumOCGs

[ASUns32](#) PDDocGetNumOCGs ([PDDoc](#) pdDoc) ;

Description

Returns the number of optional-content groups associated with a document, which is the number of unique entries in the document's **OCProperties OCGs** array.

Parameters

pdDoc	The document whose groups are counted.

Return Value

The number of OCGs for the document.

Header File

PDCalls.h

Related Methods

[PDDocHasOC](#)
[PDDocGetOCGs](#)
[PDDocReplaceOCG](#)
[PDDocEnumOCGs](#)
[PDDocGetOCConfig](#)
[PDDocGetOCContext](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDDocGetNumPages

```
ASInt32 PDDocGetNumPages (PDDoc doc) ;
```

Description

Gets the number of pages in a document.

Parameters

<code>doc</code>	The document for which the number of pages is obtained.

Return Value

The number of pages in the document. Remember to subtract one from this value if you are going to pass it to a PD-level method that takes a zero-based page number.

Header File

`PDCalls.h`

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDDocGetNumThreads

ASInt32 PDDocGetNumThreads (**PDDoc** doc);

Description

Gets the number of article threads in a document.

Parameters

doc	The document whose article thread count is obtained.

Return Value

The number of article threads in the document.

Header File

PDCalls.h

Example

```
size = PDDocGetNumThreads(pdDoc);
if (size)
{
    for(i = 0;i<size;i++)
    {
        /* get the right thread */
        thread = PDDocGetThread(pdDoc, i);
        threadIndex = i;
        len = PDThreadGetInfo(thread, "Title",
            msg, sizeof(msg));
        if(!strcmp(msg, "Contents"))
            break;
    }
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to 0x00020000 or higher.



PDDocGetOCConfig

```
PDOCConfig PDDocGetOCConfig (PDDoc pdDoc);
```

Description

Gets the built-in default optional-content configuration for the document from the **OCProperties D** entry.

Parameters

doc	The document whose configuration is obtained.

Return Value

The document's current optional-content configuration.

Header File

PDCalls.h

Related Methods

[PDDocGetOCContext](#)
[PDDocGetOCGs](#)
[PDDocEnumOCGs](#)
[PDDocEnumOCConfigs](#)
[PDOCConfigGetPDDoc](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDDocGetOCContext

```
PDOCContext PDDocGetOCContext (PDDoc pdDoc);
```

Description

Gets the built-in default optional-content context for the document. This context is used by all content drawing and enumeration calls that do not take an optional-content context parameter, or for which no context is specified.

Parameters

doc	The document whose context is obtained.

Return Value

The document's current optional-content context.

Header File

PDCalls.h

Related Methods

[PDDocGetOCConfig](#)
[PDDocGetOCGs](#)
[PDOCContextGetPDDoc](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



PDDocGetOCGs

```
PDOCG* PDDocGetOCGs (PDDoc pdDoc);
```

Description

Gets the optional-content groups for the document. The order of the groups is not guaranteed to be the creation order, and is not the same as the display order (see [PDOCConfigGetOCGOrder](#)).

Parameters

doc	The document whose OCGs are obtained.

Return Value

A **NULL**-terminated array of **PDOCG** objects. The client is responsible for freeing the array using [ASfree](#).

Header File

PDCalls.h

Related Methods

[PDDocHasOC](#)
[PDDocReplaceOCG](#)
[PDDocEnumOCGs](#)
[PDDocGetNumOCGs](#)
[PDDocGetOCConfig](#)
[PDDocGetOCContext](#)
[PDOCGGetPDDoc](#)
[PDOCMDGetPDDoc](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDDocGetOpenAction

```
PDAction PDDocGetOpenAction (PDDoc doc);
```

Description

Gets the value of the **OpenAction** key in the Catalog dictionary, which is the action performed when the document is opened. After you obtain the action, you can execute it with **AVDocPerformAction**.

Parameters

doc	The document whose open action is obtained.

Return Value

The document's open action. Invalid if there is no **OpenAction** key in the Catalog dictionary (this can be tested with **PDActionIsValid**).

Header File

PDCalls.h

Related Methods

[AVDocPerformAction](#)
[PDDocSetOpenAction](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDDocGetPageLabel

```
PDPageLabel PDDocGetPageLabel (PDDoc pdDoc, ASInt32 pageNum,
ASInt32* firstPage, ASInt32* lastPage);
```

Description

Returns the label that is in effect for the given page.

Parameters

pdDoc	The document for which a page label is desired.
pageNum	The page number of the page for which a label is requested.
firstPage	<i>(Filled by the method)</i> If non- NULL , the number of the first page that the page label is attached to.
lastPage	<i>(Filled by the method)</i> If non- NULL , the number of the last page that the page label is attached to. Setting lastPage to non- NULL forces the implementation to perform another traverse of the page label tree, with some slight performance impact.

Return Value

The label that is in effect for the given page. If there is no label object in effect, this method returns an invalid page label object and **firstPage** and **lastPage** will be set to -1.

Header File

PDCalls.h

Related Methods

[PDDocFindPageNumForLabel](#)
[PDDocGetLabelForPageNum](#)
[PDDocRemovePageLabel](#)
[PDDocSetPageLabel](#)
[PDPageLabelNew](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDDocGetPageMode

```
PDPageMode PDDocGetPageMode (PDDoc doc);
```

Description

Gets the value of the **PageMode** key in the Catalog dictionary.

NOTE: **PDDocGetFullScreen** should be used when the page mode is set to full screen.

Parameters

doc	The document whose page mode is obtained.

Return Value

Page mode value from PDF **Catalog** dictionary.

Header File

PDCalls.h

Related Methods

[PDDocSetPageMode](#)

[AVDocSetViewMode](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDDocGetPageObjByNum

```
CosObj PDDocGetPageObjByNum(PDDoc pdd, ASInt32 nPage) ;
```

Description

Returns the page Cos object corresponding to the given page number.

Parameters

pdd	The PDDoc containing the given page.
nPage	The page number.

Return Value

A Cos object representing the page, or an object of type **CosNull** if the page does not exist..

Known Exceptions

```
genErrBadParm  
pdErrBadPageObj
```

Header File

PDCalls.h

Product

reader, base, pro, pdfi

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDDocGetPermissions

```
PDPerns PDDocGetPermissions (PDDoc doc);
```

Description

Gets the permissions for the specified document. You can set permissions with [PDDocAuthorize](#).

NOTE: Deprecated in Acrobat 5.0. Use [PDDocPermRequest](#) instead.

Parameters

doc	The document whose permissions are obtained.

Return Value

A bit field indicating the document's permissions. It is an OR of the [PDPerns](#) values.

Header File

PDCalls.h

Related Methods

[PDDocAuthorize](#)
[PDDocGetSecurityData](#)
[PDDocPermRequest](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDDocGetSecurityData

```
void* PDDocGetSecurityData (PDDoc doc);
```

Description

NOTE: Superseded in Acrobat 5.0 by [PDDocPermRequest](#).

Gets the security data structure for the specified document's current security handler. Use [PDDocGetNewSecurityData](#) to get the data structure for the document's new security handler.

Parameters

<code>doc</code>	The document whose security data structure is obtained.

Return Value

A pointer to the document's current security data structure.

Header File

`PDCalls.h`

Related Methods

[PDDocGetNewSecurityData](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDDocGetStructTreeRoot

```
ASBool PDDocGetStructTreeRoot (PDDoc pdDoc,
PDSTreeRoot* treeRoot);
```

Description

Gets the structure tree root for a document.

Parameters

pdDoc	The PDDoc whose root is obtained.
treeRoot	<i>(Filled by the method)</i> The structure tree root.

Return Value

Returns **true** if structure tree root found, **false** otherwise.

Header File

PDSReadCalls.h

Related Methods

[PDDocCreateStructTreeRoot](#)
[PDDocRemoveStructTreeRoot](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDDocGetThread

[PDThread](#) PDDocGetThread ([PDDoc](#) doc, [ASInt32](#) index);

Description

Gets an article thread having the specified index.

Parameters

doc	The document containing the article thread.
index	The index of the article thread to obtain.

Return Value

The specified article thread.

Header File

PDCalls.h

Related Methods

[PDDocAddThread](#)
[PDDocRemoveThread](#)
[PDThreadIsValid](#)

Example

```
size = PDDocGetNumThreads(pdDoc);
if(size)
{
    for(i = 0;i<size;i++)
    {
        /* get the right thread */
        thread = PDDocGetThread(pdDoc, i);
        threadIndex = i;
        len = PDThreadGetInfo(thread, "Title",
                               msg, sizeof(msg));
        if(!strcmp(msg, "Contents"))
            break;
    }
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDDocGetThreadIndex

```
ASInt32 PDDocGetThreadIndex (PDDoc doc, PDThread thread);
```

Description

Gets the index of the specified article thread.

Parameters

doc	The document containing the thread.
thread	The thread whose index is obtained.

Return Value

The index of **thread** in **doc**. Returns -1 if **thread** is not in **doc**.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDDocGetTrapped

```
ASAtom PDDocGetTrapped (PDDoc doc);
```

Description

Gets the value of the **Trapped** key in the Info dictionary.

Parameters

doc	The document whose Trapped key value is obtained.

Return Value

The value of the **Trapped** key in the Info dictionary if the entry exists and is a name, or **ASAtomNull** if the entry does not exist or is not a name.

Header File

PDCalls.h

Related Methods

[PDDocSetTrapped](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDDocGetVersion

```
void PDDocGetVersion (PDDoc doc, ASInt16* majorP,  
ASInt16* minorP);
```

Description

Gets the major and minor PDF document versions. This is the PDF version of the document, which is specified in the header of a PDF file in the string “%PDF-**xx.yy**” where **xx** is the major version and **yy** is the minor version. For example, version 1.2 has the string “%PDF-1.2”. See Section H.1 in the [PDF Reference](#).

Parameters

doc	The document whose version is obtained.
majorP	<i>(Filled by the method)</i> The major version number.
minorP	<i>(Filled by the method)</i> The minor version number.

Return Value

None

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDDocGetWordFinder

```
PDWordFinder PDDocGetWordFinder (PDDoc docP,
ASInt16 WXVersion);
```

Description

Gets the word finder associated with a document. It is not necessary to destroy the word finder returned by this method.

Parameters

pdDoc	The document whose word finder is obtained.
WXVersion	The version of the word finder to get.

Return Value

The document's word finder. Returns **NULL** if the document does not have a word finder or its version does not match the version requested.

Known Exceptions

If an invalid version number is passed, [genErrBadParm](#) is thrown.

Header File

PDCalls.h

Related Methods

[PDDocCreateWordFinder](#)
[PDDocCreateWordFinderUCS](#)

Example

```
PDWordFinder WordFinder =
    PDDocGetWordFinder(pdDoc,
        WF_LATEST_VERSION );
if (WordFinder)
{
    PDWordFinderEnumWords (WordFinder,
        PageNum, ASCallbackCreateProto(
            PDWordProc, &WordCounterProc), NULL);
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDDocGetXAPMetadata

ASText PDDocGetXAPMetadata (**PDDoc** pdDoc) ;

Description

Gets the XMP metadata associated with a document.

NOTE: The term “XAP” refers to an early internal code name for Adobe’s Extensible Metadata Platform (XMP). For more information on this protocol, see the Adobe XMP specification.

Returns an **ASText** whose text is the XML text of the XMP metadata associated with the document **pdDoc**. The **ASText** becomes the property of the client, who is free to alter or destroy it.

The XMP metadata returned always represents all the properties in **pdDoc**'s **Info** dictionary, and can also contain properties not present in the **Info** dictionary. This call is preferred to **PDDocGetInfo**, which only returns properties that are in the **Info** dictionary (although the older function is supported for compatibility).

Parameters

pdDoc	The document containing the metadata.

Return Value

An **ASText** object containing the XMP metadata associated with the document **pdDoc**.

Known Exceptions

pdMetadataErrCouldntCreateMetaXAP

Header File

PDMetadataCalls.h

Related Methods

PDDocGetXAPMetadataProperty
PDDocSetXAPMetadata
PDDocSetXAPMetadataProperty

Product

base, pro, pdf

Availability

Available if **PI_PDMETADATA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDDocGetXAPMetadataProperty

ASText PDDocGetXAPMetadataProperty (**PDDoc** pdDoc,
ASText namespaceName, **ASText** path);

Description

Gets the value of an XMP metadata property associated with a document.

NOTE: The term “XAP” refers to an early internal code name for Adobe’s Extensible Metadata Platform (XMP). For more information on this protocol, see the Adobe XMP specification.

Returns an **ASText** whose text is the XML text of the value of the specified property in the XMP metadata associated with the document **pdDoc**. The **ASText** becomes the property of the client, who is free to alter or destroy it.

The XMP metadata can represent all properties in **pdDoc**'s Info dictionary, as well as other properties. This call is preferred to **PDDocGetInfo**, which only allows access to properties that are in the Info dictionary (although the latter is supported for compatibility).

Parameters

pdDoc	The document containing the metadata.
namespaceName	The XML namespace URI for the schema in which the property is to be found.
path	The name of the desired simple property. NOTE: XMP properties can have an XML substructure; this method can only retrieve values from simple textual properties.

Return Value

An **ASText** object containing the XML text of the value of the specified property in the XMP metadata associated with the document **pdDoc**, or an empty **ASText** if no such property is found.

Known Exceptions

pdMetadataErrCouldntCreateMetaXAP

Header File

PDMetadataCalls.h

Related Methods

PDDocSetXAPMetadataProperty
PDDocGetXAPMetadata
PDDocSetXAPMetadata



Product

base, pro, pdf

Availability

Available if **PI_PDMETADATA_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDDocHasOC

```
ASBool PDDocHasOC (PDDoc pdDoc) ;
```

Description

Determines whether the optional content feature is associated with the document. The document is considered to have optional content if there is an **OCProperties** dictionary in the document's catalog, and that dictionary has one or more entries in the **OCGs** array.

Parameters

doc	The document whose OC status is obtained.

Return Value

true if the document has optional content, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDDocGetOCConfig](#)
[PDDocGetOCContext](#)
[PDDocGetOCGs](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDDocImportCosDocNotes

```
ASInt32 PDDocImportCosDocNotes (PDDoc doc, CosDoc src,
const char* noteTitle, ASInt32 noteTitleLen,
PDColorValue color, void* progMon, void* monClientData,
PDDocWillImportAnnotCallback importFilter);
```

Description

Adds text annotations from **sourceDoc** to **doc**.

Parameters

doc	The document that will receive the imported annotations.
src	The document from which the annotations will be imported.
noteTitle	Not currently used.
noteTitleLen	Not currently used.
color	If non- NULL , the color attribute of imported annotations. color indicates the color space (PDDeviceGray , PDDeviceRGB , PDDeviceCMYK), and color values for the annotation.
progMon	If supplied, a procedure to call regularly to update a progress bar for the user.
monClientData	If supplied, a pointer to private data buffer used by progMon .
importFilter	A user-supplied procedure that will be called to provide a filtering process, allowing only desired annotations to import.

Return Value

The number of notes imported.

Known Exceptions

Raises an exception if the given object has the wrong Cos type. Also raises exceptions if storage is exhausted or file access fails.

Notifications

[PDDocDidImportAnnots](#)
[PDDocWillImportAnnots](#)

Header File

PDCalls.h

Related Methods

[PDDocImportNotes](#)



PDDocExportNotes

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.



PDDocImportNotes

```
ASInt32 PDDocImportNotes (PDDoc doc, PDDoc sourceDoc,
void* progMon, void* monClientData,
PDDocWillImportAnnotCallback importFilter);
```

Description

Adds text annotations (notes) from **sourceDoc** to **doc**.

Parameters

doc	The document to which the notes are exported.
sourceDoc	The document from which the notes are exported.
progMon	User-supplied progress monitor.
monClientData	Data supplied by the monitoring routine.
importFilter	User-supplied filter which determines what type of notes will be exported.

Return Value

The number of notes imported.

Known Exceptions

Raises an exception if the given object has the wrong Cos type. Also raises exceptions if storage is exhausted or file access fails.

Notifications

[PDDocDidImportAnnots](#)
[PDDocWillImportAnnots](#)

Header File

PDCalls.h

Related Methods

[PDDocExportNotes](#)
[PDDocImportCosDocNotes](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDDocInsertPages

```
void PDDocInsertPages (PDDoc doc, ASInt32 mergeAfterThisPage,
PDDoc doc2, ASInt32 startPage, ASInt32 numPages,
ASUns16 insertFlags, ProgressMonitor progMon,
void* progMonClientData, CancelProc cancelProc,
void* cancelProcClientData);
```

Description

Inserts **numPages** pages from **doc2** into **doc**. All annotations, and anything else associated with the page (such as a thumbnail image) are copied from the **doc2** pages to the new pages in **doc**. This method does not insert if **doc** equals **doc2**.

The **insertFlags** control whether bookmarks and threads are inserted along with the specified pages. If the **PDInsertAll** flag is set, this operation not only inserts the pages themselves, but also merges other document data from **doc2** into **doc**. In particular:

- The bookmark tree of **doc2** is merged into the bookmark tree of **doc** by copying it as a new first-level subtree of **doc**'s bookmark tree root, of which it becomes the last child. If **doc** has no bookmark tree, it acquires one identical to the bookmark tree from **doc2**.
- Named destinations from **doc2** (of PDF 1.1 and later) are copied into **doc**. If there are named destinations in **doc2** with the same name as some named destination in **doc**, the ones in **doc** retain their names and the copied named destinations are given new names based on the old ones with distinguishing digits added. Actions and bookmarks referring to the old names are made to refer to the new names after being copied into **doc**.
- Document logical structure from **doc2** is copied into **doc**. The top-level children of the structure tree root of **doc2** are copied as new top-level children of the structure tree root of **doc**; a structure tree root is created in **doc** if there was none before. The role maps of the two structure trees are merged, with name conflicts resolved in favor of the role mappings present in **doc**. Attribute objects are not copied, nor is class map information from **doc2** merged into that for **doc**.

If the **PDInsertAll** flag is *not* set, pages copied from **doc2** into **doc** have their structure back pointer information stripped off.

Parameters

doc	The document into which pages are inserted. This document must have at least one page.
mergeAfterThisPage	The page number in doc after which pages from doc2 are inserted. The first page is 0. If PDBeforeFirstPage (see <code>PDExpT.h</code>) is used, the pages are inserted before the first page in doc . Use PDLastPage to insert pages after the last page in doc .

doc2	The document containing the pages that are inserted into doc .
startPage	The page number of the first page in doc2 to insert into doc . The first page is 0.
numPages	The number of pages in doc2 to insert into doc . Use PDAllPages to insert all pages from doc2 into doc .
insertFlags	Flags that determine what additional information is copied from doc2 into doc . An OR of the following constants (see <code>PDExpT.h</code>): ● PDInsertBookmarks — Inserts bookmarks as well as pages. ● PDInsertThreads — Inserts threads as well as pages. ● PDInsertAll — Inserts document data from pages.
progMon	A progress monitor. Use AVAppGetDocProgressMonitor to obtain the default progress monitor. NULL may be passed, in which case no progress monitor is used.
progMonClientData	Pointer to user-supplied data to pass to progMon each time it is called. Should be NULL if progMon is NULL .
cancelProc	A cancel procedure. Use AVAppGetCancelProc to obtain the current cancel procedure. May be NULL , in which case no cancel proc is used.
cancelProcClientData	Pointer to user-supplied data to pass to cancelProc each time it is called. Should be NULL if cancelProc is NULL .

Return Value

None

Known Exceptions

Among others:

Raises [pdErrOpNotPermitted](#) unless **doc** is editable and **doc2** is not encrypted or the owner opened it.

Raises [pdErrCantUseNewVersion](#) if **doc2** is a newer major version than the Acrobat viewer understands.

Raises [pdErrTooManyPagesForInsert](#) if the insertion would result in a document with too many pages.

Raises [genErrBadParm](#) if **mergeAfterThisPage** is an invalid page number or **doc** has no pages.

Raises [genErrNoMemory](#) if there is insufficient memory to perform the insertion.

Raises [pdErrWhileRecoverInsertPages](#) if an error occurs while trying to recover from an error during inserting.

Notifications

[PDDocWillInsertPages](#)
[PDDocDidInsertPages](#)
[PDDocDidChangePages](#)
[PDDocPrintingTiledPage](#)
[PDDocDidChangeThumbs](#)
[PDDocDidAddThread](#)

Header File

PDCalls.h

Related Methods

[AVAppGetCancelProc](#)
[AVAppGetDocProgressMonitor](#)
[PDDocCreatePage](#)
[PDDocDeletePages](#)
[PDDocMovePage](#)
[PDDocReplacePages](#)

Example

```
/* Extract pages from a document and put them in a new document. */
ASFixedRect mediaBoxRect;
PDDoc oldDoc, newDoc = PDDocCreate();

PDDocCreatePage(newDoc, 0, mediaBoxRect);
PDDocInsertPages(newDoc, mergeAfterThisPage, oldDoc, start, numPages,
flags, NULL, NULL, NULL, NULL);
PDDocDeletePages(newDoc, 0, 1, NULL, NULL);
```

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDDocMovePage

```
void PDDocMovePage (PDDoc doc, ASInt32 moveToAfterThisPage,
ASInt32 pageToMove) ;
```

Description

Moves one page in a document.

Parameters

doc	The document in which a page is moved.
moveToAfterThisPage	The new location of the page to move. The first page is 0. May either be a page number, or the constant PDBeforeFirstPage (see PDExpT.h).
pageToMove	The page number of the page to move.

Return Value

None

Known Exceptions

Among others:

Raises **genErrBadParm** if **moveAfterThisPage** or **pageToMove** is invalid.

Notifications

PDDocWillMovePages
PDDocDidMovePages
PDDocDidChangePages
PDDocWillChangePages

Header File

PDCalls.h

Related Methods

PDDocInsertPages
PDDocReplacePages
PDDocDeletePages

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in PIRequir.h) is set to **0x00020000** or higher.

PDDocNewSecurityData

```
void* PDDocNewSecurityData (PDDoc doc);
```

Description

Creates a security data structure appropriate for the specified document's new security handler. The new security handler must have been previously set using [PDDocSetNewCryptHandler](#). The structure is created by calling the new security handler's [PDCryptNewSecurityDataProc](#).

After calling [PDDocNewSecurityData](#), fill the structure as appropriate, then call [PDDocSetNewSecurityData](#) with the structure. [PDDocSetNewSecurityData](#) frees the data using [ASfree](#).

Parameters

<code>doc</code>	The document for which a security data structure is created.

Return Value

The newly created security data structure.

Known Exceptions

Raises [pdErrOpNotPermitted](#) if `pdPermSecure` (see [PDPerms](#)) has not been granted for `doc`.

Raises [pdErrNeedCryptHandler](#) if the document does not have a new security handler.

Header File

PDCalls.h

Related Methods

[PDDocSetNewCryptHandler](#)
[PDDocSetNewSecurityData](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDDocOpen

```
PDDoc PDDocOpen (ASPathName fileName, ASFileSys fileSys,
PDAuthProc authProc, ASBool doRepair);
```

Description

Opens the specified document. If the document is already open, returns a reference to the already opened **PDDoc**. You must call **PDDocClose** once for every successful open. If the call fails and the exception is **pdErrNeedRebuild**, then call again with **doRepair** set to **true**. This allows the application to decide whether to perform the time-consuming repair operation.

Parameters

fileName	Pathname to the file, specified in whatever format is correct for fileSys .
fileSys	Pointer to an ASFileSysRec containing the file system in which the file resides. If NULL , uses the default file system.
authProc	Authorization callback, called only if the file has been secured (that is, if the file has either the user or the master password set). This callback should obtain whatever information is needed to determine whether the user is authorized to open the file, then call PDDocAuthorize (which returns the permissions that the authentication data enables). The Acrobat viewer's built-in authorization procedure requires the user to enter a password, and allows the user to try three times before giving up. If the authProc requires data, use PDDocOpenEx instead of PDDocOpen .
doRepair	If true , attempt to repair the file if it is damaged. If false , do not attempt to repair the file if it is damaged.

Return Value

The newly-opened document.

Known Exceptions

Among others:

Raises **pdErrNeedPassword** if the file is encrypted and **authProc** is **NULL** or returns **false**.

Raises **pdErrNotEnoughMemoryToOpenDoc** or **genErrNoMemory** if there is insufficient memory to open the document.

Raises **pdErrNeedRebuild** if the document needs to be rebuilt, and **doRepair** is **false**.

Raises [pdErrBadOutlineObj](#) if the Outlines object appears to be invalid (if the value of the **Outlines** key in the Catalog is not a **NULL** or dictionary object).

Raises [pdErrBadRootObj](#) if the Catalog object (as returned by [CosDocGetRoot](#)) is not a dictionary.

Raises [pdErrBadBaseObj](#) if the Pages tree appears to be invalid (if the value of the **Pages** key in the Catalog is not a **NULL** or dictionary object).

Raises [pdErrTooManyPagesForOpen](#) if the document contains too many pages.

Raises [cosSynErrNoHeader](#) if the document's header appears to be bad.

Raises [cosSynErrNoStartXRef](#) if no end-of-file line can be located.

Raises [cosErrRebuildFailed](#) if **doRepair** is **true**, and rebuild failed.

Header File

PDCalls.h

Related Methods

[PDDocClose](#)

[PDDocCreate](#)

[PDDocAuthorize](#)

[PDDocOpenEx](#)

[PDDocOpenFromASFile](#)

[PDDocOpenFromASFileEx](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDDocOpenEx

```
PDDoc PDDocOpenEx (ASPathName fileName, ASFileSys fileSys,
PDAuthProcEx authProcEx, void* authProcClientData,
ASBool doRepair);
```

Description

Opens the specified document. If the document is already open, returns a reference to the already opened **PDDoc**. You must call **PDDocClose** once for every successful open. If the call fails and the exception is **pdErrNeedRebuild**, then call again with **doRepair true**. This allows the application to decide whether to perform the time-consuming repair operation.

Parameters

fileName	Pathname to the file, specified in whatever format is correct for fileSys .
fileSys	Pointer to an ASFileSysRec containing the file system in which the file resides.
authProcEx	Authorization callback, called only if the file has been secured (that is, if the file has either the user or the master password set). This callback should obtain whatever information is needed to determine whether the user is authorized to open the file, then call PDDocAuthorize (which returns the permissions that the authentication data enables). The Acrobat viewer's built-in authorization procedure requires the user to enter a password, and allows the user to try three times before giving up.
authProcClientData	Pointer to user-supplied data to pass to authProcEx each time it is called.
doRepair	If true , attempt to repair the file if it is damaged. If false , do not attempt to repair the file if it is damaged.

Return Value

The newly-opened document.

Known Exceptions

Among others:

Raises **pdErrNeedPassword** if the file is encrypted and **authProcEx** is **NULL** or returns **false**.

Raises [pdErrNotEnoughMemoryToOpenDoc](#) or [genErrNoMemory](#) if there is insufficient memory to open the document.

Raises [pdErrNeedRebuild](#) if the document needs to be rebuilt, and `doRepair` is `false`.

Raises [pdErrBadOutlineObj](#) if the Outlines object appears to be invalid (if the value of the `Outlines` key in the Catalog is not a `NULL` or dictionary object).

Raises [pdErrBadRootObj](#) if the Catalog object (as returned by [CosDocGetRoot](#)) is not a dictionary.

Raises [pdErrBadBaseObj](#) if the Pages tree appears to be invalid (if the value of the `Pages` key in the Catalog is not a `NULL` or dictionary object).

Raises [pdErrTooManyPagesForOpen](#) if the document contains too many pages.

Raises [cosSynErrNoHeader](#) if the document's header appears to be bad.

Raises [cosSynErrNoStartXRef](#) if no end-of-file line can be located.

Raises [cosErrRebuildFailed](#) if `doRepair` is `true`, and rebuild failed.

Header File

`PDCalls.h`

Related Methods

[PDDocClose](#)

[PDDocCreate](#)

[PDDocAuthorize](#)

[PDDocOpen](#)

[PDDocOpenFromASFile](#)

[PDDocOpenFromASFileEx](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020002` or higher.

PDDocOpenFromASFile

```
PDDoc PDDocOpenFromASFile (ASFile aFile, PDAuthProc authProc,
ASBool doRepair);
```

Description

Opens the document specified by the **ASFile**. **aFile** must be a valid **ASFile**. It is the caller's responsibility to dispose of the **ASFile** after calling **PDDocClose**.

This method is useful when the document referenced by the **ASFile** is not on the local machine, and is being retrieved incrementally using the multi-read protocol of an **ASFileSys**. If the bytes required to open a **PDDoc** are not yet available, this method will raise the exception **fileErrBytesNotReady**. The client should call **PDDocOpenFromASFile** until this exception is no longer raised.

Parameters

aFile	The ASFile to open. The ASFile should be released after the PDDoc is closed.
authProc	Authorization callback, called only if the file is encrypted. This callback should obtain whatever information is needed to determine whether the user is authorized to open the file, then call PDDocAuthorize (which returns the permissions that the authentication data enables). The Acrobat viewer's built-in authorization procedure requires the user to enter a password, and allows the user to try three times before giving up.
doRepair	If true , attempt to repair the file if it is damaged. If false , do not attempt to repair the file if it is damaged.

Return Value

A valid **PDDoc** if successfully opened.

Known Exceptions

Raises **pdErrNeedPassword** if the file is encrypted and **authProc** is **NULL** or returns **false**.

Raises **fileErrBytesNotReady** if the bytes required to open a **PDDoc** are not yet available.

Header File

PDCalls.h

Related Methods

[PDDocClose](#)
[PDDocOpen](#)
[PDDocOpenEx](#)



PDDocOpenFromASFileEx

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

PDDocOpenFromASFileEx

```
PDDoc PDDocOpenFromASFileEx (ASFile aFile,
PDAuthProcEx authProcEx, void* authProcClientData,
ASBool doRepair);
```

Description

Opens the document specified by the **ASFile**. **aFile** must be a valid **ASFile**. It is the caller's responsibility to dispose of the **ASFile** after calling **PDDocClose**.

This method is useful when the document referenced by the **ASFile** is not on the local machine, and is being retrieved incrementally using the multiread protocol of an **ASFileSys**. If the bytes required to open a **PDDoc** are not yet available, this method will raise the exception **fileErrBytesNotReady**. The client should call **PDDocOpenFromASFile** until this exception is no longer raised.

Parameters

aFile	The ASFile to open. The ASFile should be released after the PDDoc is closed.
authProcEx	Authorization callback, called only if the file is encrypted. This callback should obtain whatever information is needed to determine whether the user is authorized to open the file, then call PDDocAuthorize (which returns the permissions that the authentication data enables). The Acrobat viewer's built-in authorization procedure requires the user to enter a password, and allows the user to try three times before giving up.
authProcClientData	Pointer to user-supplied data to pass to authProcEx each time it is called.
doRepair	If true , attempt to repair the file if it is damaged. If false , do not attempt to repair the file if it is damaged.

Return Value

A valid **PDDoc** if successfully opened.

Known Exceptions

Raises **pdErrNeedPassword** if the file is encrypted and **authProc** is **NULL** or returns **false**.

Raises **fileErrBytesNotReady** if the bytes required to open a **PDDoc** are not yet available.

Header File

PDCalls.h



Related Methods

[PDDocClose](#)
[PDDocOpen](#)
[PDDocOpenEx](#)
[PDDocOpenFromASFile](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020002` or higher.

PDDocOpenWithParams

```
PDDoc PDDocOpenWithParams (PDDocOpenParams openParams) ;
```

Description

Opens the document specified by the **ASFile** or **ASFileSys/ASPathName**. If both are set, the **ASFile** is used and the **fileSys/pathName** is ignored.

Parameters

openParams	A structure that defines which PDF file is opened. Parameters like: filename, file system, an authorization procedure, and a set of flags that define what permissions the user has on a file.

Return Value

The **PDDoc** for the PDF document described by the structure passed in **openParams**.

Header File

PDCalls.h

Related Methods

[PDDocOpen](#)
[PDDocOpenEx](#)
[PDDocOpenFromASFile](#)
[PDDocOpenFromASFileEx](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDDocPermRequest

```
PDPPermReqStatus PDDocPermRequest (PDDoc pdDoc,  
PDPPermReqObj reqObj, PDPPermReqOpr reqOpr, void *authData);
```

Description

Checks the permissions that the user has placed on the specified document using the latest permissions format, and determines whether the requested operation is allowed for the specified object in the document.

NOTE: This method supercedes [PDDocGetPermissions](#).

This method first checks the requested object and operation in a cached list. If a value is not found, it calls the document's security handler via [PDCryptAuthorizeExProc](#) to request permissions for the operation. If the document's security handler does not support this Acrobat 5.0 call, the method calls [PDCryptAuthorizeProc](#) instead. The method then interprets the returned [PDPerms](#) to determine whether the requested operation is allowed for the specified object in the document.

Parameters

pdDoc	The PDDoc whose permissions are being requested.
reqObj	The target object of the permissions request.
reqOpr	The target operation of the permissions request.
authData	A pointer to an authorization data structure.

Return Value

The request status constant, 0 if the requested operation is allowed, a non-zero status code otherwise.

Known Exceptions

Numerous

Header File

PDCalls.h

Related Methods

[PDDocAuthorize](#)
[PDDocGetPermissions](#) (obsolete)
[PDDocGetNewCryptHandler](#)
[PDDocSetNewCryptHandler](#)
[PDDocSetNewCryptHandlerEx](#)
[PDRegisterCryptHandler](#)
[PDRegisterCryptHandlerEx](#)

**Product**

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDDocReadAhead

```
void PDDocReadAhead (PDDoc doc, ASUns32 flags,  
void* clientData);
```

Description

Used for page-at-a-time downloading and byte-serving Acrobat data. If a document is being viewed over a slow file system, **PDDocReadAhead** issues a byte range request for all the data associated with the flags in **flags**.

Parameters

doc	The document being read.
flags	Flags describing type of data to read ahead. Must be OR of flags in PDDocReadAhead Flags .
clientData	Currently unused.

Return Value

None

Header File

PDCalls.h

Example

```
PDDocReadAhead(pdDoc,  
kPDDocReadAheadAcroForms, (void *) NULL);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

PDDocReadAheadEmbeddedFile

```
void PDDocReadAheadEmbeddedFile (PDDoc doc,
CosObj embeddedFileObj);
```

Description

Used for page-at-a-time downloading and byte-serving Acrobat data. If a document is being viewed over a slow file system, the method issues a byte range request for all the data associated with an embedded file.

Parameters

doc	The document being read.
embeddedFileObj	The Cos object of the embedded file stream (the stream referenced by entries in the EF dictionary).

Return Value

None

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDDocReadAheadPages

```
void PDDocReadAheadPages (PDDoc doc, ASInt32 startPage,
ASInt32 nPages) ;
```

Description

Reads ahead **nPages** starting at **startPage** (if the file is linearized).

Parameters

doc	The document for which pages are read ahead.
startPage	The page for which read ahead is initiated.
nPages	The number of pages to read ahead.

Return Value

None

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDDocRelease

```
void PDDocRelease (PDDoc doc);
```

Description

Decrements a document's reference count. The document will not be closed until the reference count is zero, or the application terminates.

Parameters

<code>doc</code>	The document whose reference count is decremented.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

PDCalls.h

Related Methods

[PDDocAcquire](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.



PDDocRemoveNameTree

```
void PDDocRemoveNameTree (PDDoc thePDDoc, ASAtom theTree);
```

Description

Removes the name tree inside the Names dictionary with the specified key name. Does nothing if no object with that name exists.

Parameters

thePDDoc	The document from which a name tree is removed.
theTree	The name tree to remove.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocCreateNameTree](#)

[PDDocGetNameTree](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDDocRemoveOpenAction

```
void PDDocRemoveOpenAction (PDDoc doc);
```

Description

Removes the value of the **OpenAction** key in the Catalog dictionary. The value is the action performed when the document is opened.

Parameters

<code>doc</code>	The document whose open action is removed.

Return Value

None

Known Exceptions

[genErrBadParm](#)
[pdErrOpNotPermitted](#)

Header File

PDCalls.h

Related Methods

[PDDocGetOpenAction](#)
[PDDocSetOpenAction](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDDocRemovePageLabel

```
void PDDocRemovePageLabel (PDDoc pdDoc, ASInt32 pageNum) ;
```

Description

Removes the page label that is attached to the specified page, effectively merging the specified range with the previous page label sequence.

Parameters

pdDoc	The document from which a page label is removed.
pageNum	The page from which the page label is removed.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocSetPageLabel](#)
[PDDocGetPageLabel](#)
[PDPageLabelNew](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDDocRemoveStructTreeRoot

```
void PDDocRemoveStructTreeRoot (PDDoc pdDoc);
```

Description

Removes, but does *not* destroy, the specified **StructTreeRoot** element from the specified **PDDoc**.

Parameters

pdDoc	The PDDoc for which the StructTreeRoot element is removed.

Return Value

None

Header File

PDSWriteCalls.h

Related Methods

[PDDocCreateStructTreeRoot](#)
[PDDocGetStructTreeRoot](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDDocRemoveThread

```
void PDDocRemoveThread (PDDoc doc, ASInt32 index);
```

Description

Removes an article thread from a document. If you also wish to destroy the thread, use [PDThreadDestroy](#) after calling [PDDocRemoveThread](#).

Parameters

doc	The document from which the thread is removed.
index	The index of the thread to remove.

Return Value

None

Notifications

[PDDocWillRemoveThread](#)
[PDDocDidRemoveThread](#)

Header File

PDCalls.h

Related Methods

[PDThreadDestroy](#)
[PDDocAddThread](#)
[PDBeadGetThread](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDDocReplaceOCG

```
void PDDocReplaceOCG (PDOCG replaceOCG, PDOCG keepOCG);
```

Description

In the document associated with a specified optional-content group, replaces that group with another group.

Parameters

replaceOCG	The OCG to replace.
keepOCG	The replacement OCG.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocHasOC](#)
[PDDocGetOCGs](#)
[PDDocEnumOCGs](#)
[PDDocGetNumOCGs](#)
[PDDocGetOCConfig](#)
[PDDocGetOCContext](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDDocReplacePages

```
void PDDocReplacePages (PDDoc doc, ASInt32 startPage,
PDDoc doc2, ASInt32 startPageDoc2, ASInt32 numPages,
ASBool mergeTextAnnots, ProgressMonitor progMon,
void* progMonClientData, CancelProc cancelProc,
void* cancelProcClientData);
```

Description

Replaces the specified range of pages in one document with pages from another. The contents, resources, size and rotation of the pages are replaced. Bookmarks are *not* copied, because they are attached to the document, not to individual pages.

NOTE: Annotations in the replaced pages are not replaced and remain with the page. Use [PDDocDeletePages](#) to remove annotations.

Parameters

doc	The document in which pages are replaced.
startPage	The first page number in doc to replace. The first page is 0.
doc2	The document from which pages are copied into doc .
startPageDoc2	The page number of the first page in doc2 to copy. The first page is 0.
numPages	The number of pages to replace.
mergeTextAnnots	If true , text annotations from doc2 are appended if they are different than all existing annotations on the page in doc . No other types of annotations are copied.
progMon	A progress monitor. Use AVAppGetDocProgressMonitor to obtain the default progress monitor. NULL may be passed, in which case no progress monitor is used.
progMonClientData	Pointer to user-supplied data to pass to progMon each time it is called. Should be NULL if progMon is NULL .
cancelProc	Currently unused. A cancel procedure. Use AVAppGetCancelProc to obtain the current cancel procedure. May be NULL , in which case no cancel proc is used.

cancelProcClientData	Pointer to user-supplied data to pass to cancelProc each time it is called. Should be NULL if cancelProc is NULL .

Return Value

None

Known Exceptions

Among others:

Raises [pdErrOpNotPermitted](#) unless **doc** is editable and **doc2** is not encrypted or the owner opened it.

Raises [pdErrCantUseNewVersion](#) if **doc2** is a newer major version than the Acrobat viewer understands.

Raises [genErrBadParm](#) if **numPages** < 1, 1 **startPage** < 0, **startPage** + **numPages** is greater than the number of pages in **doc**, **startPageDoc2** < 0, or **startPageDoc2** + **numPages** is greater than the number of pages in **doc2**.

Raises [genErrNoMemory](#) if there is insufficient memory to perform the insertion.

Notifications[PDDocWillReplacePages](#)[PDDocDidReplacePages](#)[PDDocDidChangePages](#)[PDDocWillChangePages](#)**Header File**

PDCalls.h

Related Methods[PDDocInsertPages](#)[PDDocMovePage](#)[PDDocDeletePages](#)[PDDocCreatePage](#)**Product**

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDDocRequestEntireFile

```
void PDDocRequestEntireFile (PDDoc doc,  
PDDocRequestEntireFileProc requestProc, void *clientData);
```

Description

Requests the document file and performs the specified procedure on it.

Parameters

doc	The document for which pages are read ahead.
startPage	The first page requested.
nPages	The number of pages requested.
requestProc	The procedure to call to process the request.
clientData	A pointer to user-defined data to pass to the requestProc .

Return Value

None

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDDocRequestPages

```
void PDDocRequestPages (PDDoc doc, ASInt32 startPage,
ASInt32 nPages, PDDocRequestPagesProc requestProc,
void *clientData);
```

Description

Requests **nPages** starting at **startPage**, and performs a specified procedure on them.

Parameters

doc	The document for which pages are read ahead.
startPage	The first page requested.
nPages	The number of pages requested.
requestProc	The procedure to call to process the request.
clientData	A pointer to user-defined data to pass to the requestProc .

Return Value

None

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDDocSave

```
void PDDocSave (PDDoc doc, PDSaveFlags saveFlags,
ASPathName newPath, ASFileSys fileSys,
ProgressMonitor progMon, void* progMonClientData);
```

Description

Saves a document to disk. If a full save is requested to the original path, the file is saved to a file system-determined temporary file, the old file is deleted, and the temporary file is renamed to **newPath**. You must call **PDDocClose** to release resources; do *not* call **PDDocRelease**.

If the document was created with **PDDocCreate**, at least one page must be added using **PDDocCreatePage** or **PDDocInsertPages** before Acrobat can save the document.

You can replace this method with your own version, using **HFTReplaceEntry**.

A full save with linearization optimizes the PDF file. During optimization, *all* objects in a PDF file are rearranged, many of them acquiring not only a new file position, but also a new Cos object number. At the end of the save operation, Acrobat flushes its information of the PD layer and below to synchronize its in-memory state with the new disk file just written.

It is crucial that all objects that have been acquired from a PDDoc be released before Acrobat attempts to flush its in-memory state. This includes any object that was acquired with a **PD*Acquire** method, such as **PDDocAcquirePage** or **PDBeadAcquirePage**. Failing to release these objects before the full save results in a save error, and the resulting PDF file is not valid. In addition, all PD level objects and Cos objects derived from the **PDDoc** are *no longer valid* after the full save. Attempting to use these objects after a full save produces undefined results.

Clients and applications should register for the **PDDocWillSaveEx** and **PDDocDidSave** notifications so that they can clean up appropriately. See these notifications for more information on releasing and reacquiring objects from the **PDDoc**.

Parameters

doc	The document to save.
saveFlags	A bit field composed of an OR of the PDSaveFlags values.
newPath	The path to which the file is saved. A path must be specified when either PDSaveFull or PDSaveCopy are used for saveFlags . If PDSaveIncremental is specified in saveFlags , then newPath should be NULL . If PDSaveFull is specified and newPath is the same as the file's original path, the new file is saved to a file system-determined temporary path, then the old file is deleted and the new file is renamed to newPath .

fileSys	File system. If NULL , uses the fileSys of the document's current backing file. Files can only be saved to the same file system. fileSys must be either NULL or the default file system obtained with ASGetDefaultFileSys , otherwise an error is raised.
progMon	Progress monitor. Use AVAppGetDocProgressMonitor to obtain the default. NULL may be passed, in which case no progress monitor is used.
progMonClientData	Pointer to user-supplied data to pass to progMon each time it is called. Should be NULL if progMon is NULL .

Return Value

None

Known Exceptions

Among others:

Raises **pdErrAfterSave** if the save was completed successfully, but there were problems cleaning up afterwards. The document is no longer consistent and cannot be changed. It must be closed and reopened.

Raises **pdErrOpNotPermitted** if saving is not permitted. Saving is permitted if either edit or **editNotes** (see **PDPerms**) is allowed, or you are doing a full save and **saveAs** is allowed.

Raises **pdErrAlreadyOpen** if **PDSaveFull** is used, and the file specified by **newPath** is already open.

Notifications

PDDocWillSave
PDDocDidSave

Header File

PDCalls.h

Related Methods

PDDocClose
PDDocSaveWithParams

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDDocSaveWithParams

```
void PDDocSaveWithParams (PDDoc doc,
PDDocSaveParams inParams);
```

Description

Saves a document to disk as specified in a parameter’s structure. This is essentially the same as [PDDocSave](#) with two additional parameters: a cancel proc and cancel proc client data (so you could cut and paste description information, and so on, from [PDDocSave](#)). You can replace this method with your own version, using [HFTReplaceEntry](#).

NOTE: Saving a **PDDoc** invalidates all objects derived from it. See [PDDocSave](#) for important information about releasing objects that you may have acquired or used from a **PDDoc** before it is saved.

Parameters

doc	The document to save.
inParams	PDDocSaveParams structure specifying how the document should be saved.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocSave](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

PDDocSetFlags

```
void PDDocSetFlags (PDDoc doc, ASInt32 flags);
```

Description

Sets information about the document's file and its state. This method can only be used to set, not clear, flags. As a result, it is not possible, for example, to use this method to clear the flag that indicates that a document has been modified and needs to be saved. Instead, use [PDDocClearFlags](#) to clear flags.

Parameters

doc	The document whose flags are set.
flags	A bit field composed of an OR of the PDDocFlags values.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocGetFlags](#)
[PDDocClearFlags](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDDocSetFullScreen

```
void PDDocSetFullScreen (PDDoc pdDoc, ASBool fs);
```

Description

Sets whether this document opens in full-screen mode. This provides an alternative to calling [PDDocSetPageMode](#) with **PDFullScreen**.

Parameters

pdDoc	The document to set.
fs	true if the document is set to open in full-screen mode, false otherwise.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocGetFullScreen](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

PDDocSetInfo

```
void PDDocSetInfo (PDDoc doc, const char* infoKey,
char* buffer, ASInt32 nBytes);
```

Description

Sets the value of a key in a document’s Info dictionary. However, it is preferable to use [PDDocSetXAPMetadataProperty](#), because it also allows accessing XMP properties that are not duplicated in the Info dictionary.

See Section 10.2.1 on **Info** dictionaries in the [PDF Reference](#) for information about Info dictionaries. All values in the Info dictionary should be strings; other data types such as numbers and booleans should not be used as values in the Info dictionary. If an info dictionary key is specified that is not currently in the info dictionary, it is added to the dictionary.

Users may define their own Info dictionary entries. In this case, it is strongly recommended that the key have the developer’s prefix assigned by the Adobe Developers Association.

NOTE: For Roman viewers, this text is always stored in the **PDFDocEncoding**. For non-Roman character set viewers, this text is stored as **PDFDocEncoding** or Unicode, depending on the file’s creator.

Parameters

doc	The document whose Info dictionary key is set.
infoKey	The name of the Info dictionary key whose value is set.
buffer	Buffer containing the value to associate with infoKey .
nBytes	The number of bytes in buffer .

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocGetInfo](#)
[PDDocSetXAPMetadataProperty](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDDocSetNewCryptFilterData

```
void PDDocSetNewCryptFilterData (PDDoc pdDoc,
ASAtom filterName, char *cryptData, ASInt32 cryptDataLen);
```

Description

Sets the encrypted data for the specified document’s encryption filter to decrypt. Call before accessing the stream to be decrypted.

NOTE: Not available for Adobe Reader.

Parameters

doc	The document whose new encrypted data is set.
filterName	The ASAtom corresponding to the name of the security filter used by the document.
cryptData	The new encrypted data for the document.
cryptDataLen	The length of cryptData in bytes.

Return Value

None

Known Exceptions

Raises pdErrNoCryptHandler if there is no security handler registered for the document.

Raises pdErrOpNotPermitted if the document’s permissions do not allow its data to be modified.

Header File

PDCalls.h

Related Methods

- PDcryptAuthorizeFilterAccess
- PDDocSetNewCryptFilterMethod
- PDDocSetNewDefaultFilters

Product

reader, base, pro, pdf

Availability

Available if PI_PDMODEL_VERSION (in PIRequir.h) is set to 0x00060000 or higher.

PDDocSetNewCryptFilterMethod

```
void PDDocSetNewCryptFilterMethod (PDDoc pdDoc,
ASAtom method);
```

Description

Sets or resets the specified document's security filter method, used for encryption and decryption of the document's data.

NOTE: Not available for Adobe Reader.

Parameters

doc	The document whose new security filter method is set.
method	The ASAtom corresponding to the name of the security filter to use.

Return Value

None

Known Exceptions

Raises [pdErrNoCryptHandler](#) if there is no security handler registered for the document.

Header File

PDCalls.h

Related Methods

[PDCryptAuthorizeFilterAccess](#)
[PDDocSetNewCryptFilterData](#)
[PDDocSetNewDefaultFilters](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

PDDocSetNewCryptHandler

```
void PDDocSetNewCryptHandler (PDDoc pdDoc,
ASAtom newCryptHandler);
```

Description

Sets specified document's new security handler (that is, the security handler that will be used after the document is saved).

This method returns with no action if the new security handler is the same as the old one. Otherwise, it calls the new security handler's [PDCryptNewSecurityDataProc](#) to set the document's **newSecurityData** field. If the new security handler does not have this callback, the document's **newSecurityData** field is set to 0.

Parameters

doc	The document whose new security handler is set.
newCryptHandler	The ASAtom for the name of the new security handler to use for the document. This name must be the same as the pdfName used when the security handler was registered using PDRegisterCryptHandler . Use ASAtomNull to remove security from the document.

Return Value

None

Known Exceptions

Raises [pdErrNoCryptHandler](#) if there is no security handler registered with the specified name and the name is not **ASAtomNull**.

Raises [pdErrOpNotPermitted](#) if the document's permissions do not allow its security to be modified.

Header File

PDCalls.h

Related Methods

[PDDocGetNewCryptHandler](#)
[PDDocPermRequest](#)
[PDDocSetNewCryptHandlerEx](#)
[PDRegisterCryptHandler](#)
[PDRegisterCryptHandlerEx](#)

Product

base, pro, pdf



Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDDocSetNewCryptHandlerEx

```
void PDDocSetNewCryptHandlerEx (PDDoc pdDoc,
ASAtom newCryptHandler, void *currentAuthData);
```

Description

Extends [PDDocSetNewCryptHandler](#) for Acrobat 6.0. Sets specified document's new security handler (that is, the security handler that will be used after the document is saved). This method should be called when the current document's security handler requires authorization data to validate permission to change security handlers.

This method returns with no action if the new security handler is the same as the old one. Otherwise, the new security handler's [PDCryptNewSecurityDataProc](#) is called to set the document's **newSecurityData** field. If the new security handler does not have this callback, the document's **newSecurityData** field is set to 0.

Parameters

doc	The document whose new security handler is set.
newCryptHandler	The ASAtom corresponding to the name of the new security handler to use for the document. This name must be the same as the pdfName used when the security handler was registered using PDRegisterCryptHandler . Use ASAtomNull to remove security from the document.
currentAuthData	A pointer to authorization data to be passed to the PDCryptAuthorizeProc callback for the document's current security handler. For the Acrobat viewer's built-in security handler, the password is passed in the authData parameter.

Return Value

None

Known Exceptions

Raises [pdErrNoCryptHandler](#) if there is no security handler registered with the specified name and the name is not **ASAtomNull**.

Raises [pdErrOpNotPermitted](#) if the document's permissions do not allow its security to be modified.

Header File

PDCalls.h

Related Methods

[PDDocGetNewCryptHandler](#)



PDDocPermRequest
PDDocSetNewCryptHandler
PDRegisterCryptHandler
PDRegisterCryptHandlerEx

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDDocSetNewDefaultFilters

```
void PDDocSetNewDefaultFilters (PDDoc pdDoc,
ASAtom defaultStmFilterName, ASAtom defaultStrFilterName);
```

Description

Sets or resets the document's default security filter methods for streams and strings, used to encrypt and decrypt the document's data. This method is only valid with version 4 algorithms (**V4** in the **Encrypt** dictionary).

NOTE: Not available for Adobe Reader.

Parameters

doc	The document whose new security filter is set.
defaultStmFilterName	The ASAtom corresponding to the name of the default security filter to use for streams. The filter must exist and be registered.
defaultStrFilterName	The ASAtom corresponding to the name of the default security filter to use for strings. The filter must exist and be registered.

Return Value

None

Known Exceptions

Raises **pdErrNoCryptHandler** if there is no security handler registered for the document.

Header File

PDCalls.h

Related Methods

[PDCryptAuthorizeFilterAccess](#)
[PDDocSetNewCryptFilterData](#)
[PDDocSetNewCryptFilterMethod](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDDocSetNewSecurityData

```
void PDDocSetNewSecurityData (PDDoc doc, void* secData);
```

Description

Sets the security data structure for the specified document's new security handler. Use [PDDocSetNewCryptHandler](#) to set a new security handler for a document.

Parameters

doc	The document whose new security data structure is set.
secData	Pointer to the new security data structure to set for doc . See PDDocNewSecurityData for information on creating and filling this structure.

Return Value

None

Known Exceptions

Raises [pdErrrNeedCryptHandler](#) if the document does not have a new security handler.

Raises [pdErrOpNotPermitted](#) if the document's permissions cannot be changed.

Header File

PDCalls.h

Related Methods

[PDDocGetNewSecurityData](#)

[PDDocGetSecurityData](#)

[PDDocNewSecurityData](#)

[PDDocSetNewCryptHandler](#)

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDDocSetOpenAction

```
void PDDocSetOpenAction (PDDoc doc, PDAction action);
```

Description

Sets the value of the **OpenAction** key in the Catalog dictionary, which is the action performed when the document is opened.

Parameters

doc	The document whose open action is set.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocGetOpenAction](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDDocSetPageLabel

```
void PDDocSetPageLabel (PDDoc pdDoc, ASInt32 pageNum,
PDPageLabel pgLabel);
```

Description

Attaches a label to a page. This establishes the numbering scheme for that page and all following it, until another page label is encountered. This label allows PDF producers to define a page numbering system other than the Acrobat default.

If **pageNum** is less than 0 or greater than the number of pages in **pdDoc**, the method does nothing.

Parameters

pdDoc	The document containing the page to label.
pageNum	The number of the page to label.
pgLabel	Label for the page specified by pageNum.

Return Value

None

Known Exceptions

Raises [genErrBadParm](#) if **pgLabel** is not a valid **PDPageLabel**.

Notifications

[PDDocPageLabelDidChange](#)

Header File

PDCalls.h

Related Methods

[PDDocFindPageNumForLabel](#)
[PDDocGetPageLabel](#)
[PDDocGetLabelForPageNum](#)
[PDDocRemovePageLabel](#)
[PDPageLabelNew](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDDocSetPageMode

```
void PDDocSetPageMode (PDDoc doc, PDPageMode mode);
```

Description

Sets the value of the **PageMode** key in the Catalog dictionary.

Parameters

doc	The document whose page mode is set.
mode	The page mode to set.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocGetPageMode](#)

[AVDocSetViewMode](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDDocSetTrapped

```
void PDDocSetTrapped (PDDoc doc, ASAtom newValue);
```

Description

Sets the value of the **Trapped** key in the Info dictionary to the specified **ASAtom**.
This method causes the corresponding XMP metadata item to be set to a string reflecting the characters in the **ASAtom**.

Parameters

doc	The document whose Trapped key value to set.
newValue	The new value of the Trapped key in the Info dictionary, or ASAtomNull to remove any existing entry. The method does not check that the value is one of the allowed values for the key.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocGetTrapped](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDDocSetXAPMetadata

```
void PDDocSetXAPMetadata (PDDoc pdDoc, ASText metadataASText);
```

Description

Sets the XMP metadata associated with a document.

NOTE: The term “XAP” refers to an early internal code name for Adobe’s Extensible Metadata Platform (XMP). For more information on this protocol, see the Adobe XMP specification.

Replaces the XMP metadata associated with the document **pdDoc** with the XMP metadata stored in **metadataASText**.

The contents of **metadataASText** must be well-formed XML and Resource Description Format (RDF) as defined by the W3C (see <http://www.w3.org/RDF>) that also forms valid XMP. If **metadataASText** is ill-formed, an error is raised.

The call does not destroy **metadataASText** or alter its text. This method copies the textual information it needs, so subsequent alteration or destruction of **metadataASText** does not affect the document XMP metadata.

Calling **PDDocSetXAPMetadata** changes the contents of **pdDoc**'s **Info** dictionary to reflect the values of corresponding metadata properties represented in **metadataASText**. The XMP metadata can also contain properties that are not reflected in the **Info** dictionary.

NOTE: This method raises an exception if the user does not have permission to change the document.

NOTE: If you use this method to set metadata that does not respect the requirement that aliased metadata items (such as **pdf:Title** and **xap:Title**) be equal, then the mechanism that maintains this equality when you set metadata via **PDDocSetInfo** is disabled.

Parameters

pdDoc	The document whose metadata is to be set.
metadataASText	An ASText object containing the metadata to be stored in the document.

Return Value

None

Known Exceptions

ErrSysPDModel
pdMetadataErrCouldntCreateMetaXAP
pdErrOpNotPermitted if **pdDoc** is not writable

Header File

PDMetadataCalls.h

Related Methods

[PDDocGetXAPMetadata](#)
[PDDocSetXAPMetadataProperty](#)
[PDDocSetInfo](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMETADATA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDDocSetXAPMetadataProperty

```
void PDDocSetXAPMetadataProperty (PDDoc pdDoc,
    AText namespaceName, AText namespacePrefix, AText path,
    AText newValue);
```

Description

Sets the value of an XMP metadata property associated with a document.

NOTE: The term “XAP” refers to an early internal code name for Adobe’s Extensible Metadata Platform (XMP). For more information on this protocol, see the Adobe XMP specification.

The XMP metadata represents all the properties in **pdDoc**'s **Info** dictionary, and can also contain properties that are not in the **Info** dictionary. This call is preferred to [PDDocSetInfo](#), which only allows access to properties that are in the **Info** dictionary (although the older function is supported for compatibility).

Parameters

pdDoc	The document containing the metadata.
namespaceName	The XML namespace URI for the schema in which the property is to be found.
namespacePrefix	A brief string to be used as an abbreviation when creating the XML representation of the property. This string must not be empty.
path	The name of the simple property to be modified. NOTE: XMP properties can have an XML substructure; this method can only set a value for simple textual properties.
newValue	The new XML text value for the property.

Return Value

None

Known Exceptions

[pdMetadataErrCouldntCreateMetaXAP](#)

Header File

PDMetadataCalls.h



Related Methods

[PDDocGetXAPMetadataProperty](#)
[PDDocGetXAPMetadata](#)
[PDDocSetXAPMetadata](#)

Product

base, pro, pdf

Availability

Available if `PI_PDMETADATA_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

PDFFileSpec

PDFFileSpecAcquireASPath

```
ASPathName PDFFileSpecAcquireASPath (PDFFileSpec fileSpec,  
ASPathName relativeToThisPath);
```

Description

Acquires an **ASPathName** for the specified file specification and relative path.

Parameters

fileSpec	The file specification for which an ASPathName is acquired.
relativeToThisPath	A pathname relative to which the fileSpec is interpreted. If NULL , fileSpec is assumed to be an absolute, not a relative, path.

Return Value

The **ASPathName** corresponding to **fileSpec**.

After you are done using the **ASPathName**, you must free it using [ASFileSysReleasePath](#).

Header File

PDCalls.h

Related Methods

[ASFileSysReleasePath](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDFFileSpecFromCosObj

```
PDFFileSpec PDFFileSpecFromCosObj (CosObj obj);
```

Description

Converts an appropriate string or dictionary Cos object to a file specification. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

<code>obj</code>	The Cos object to convert to a file specification.

Return Value

The file specification corresponding to `obj`.

Known Exceptions

Raises `pdErrrBadFileSpec` if the file specification is not valid, as determined by `PDFFileSpecIsValid`.

Header File

`PDCalls.h`

Related Methods

`PDFFileSpecGetCosObj`
`PDFFileSpecIsValid`

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.



PDFFileSpecGetCosObj

```
CosObj PDFFileSpecGetCosObj (PDFFileSpec fileSpec);
```

Description

Gets the Cos object associated with a file specification. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

fileSpec	The file specification whose Cos object is obtained.

Return Value

The string or dictionary Cos object corresponding to the file specification.

Header File

PDCalls.h

Related Methods

[PDFFileSpecFromCosObj](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDFFileSpecGetDIPath

```
ASInt32 PDFFileSpecGetDIPath (PDFFileSpec fileSpec,
char* buffer, ASInt32 bufLen);
```

Description

Gets the device-independent pathname from a file specification.

Parameters

fileSpec	The file specification whose device-independent pathname is obtained.
buffer	<i>(Filled by the method)</i> null-terminated device-independent pathname. If buffer is NULL , the method simply returns the length of the pathname.
bufLen	Length of buffer , in bytes. If the device-independent pathname is longer than this, only the first bufLen – 1 bytes are copied into buffer , plus a NULL at the end of the buffer.

Return Value

Number of characters (excluding the **NULL**) copied into **buffer**.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDFFileSpecGetDoc

```
PDDoc PDFFileSpecGetDoc (PDFFileSpec fileSpec);
```

Description

Gets the **PDDoc** that contains **fileSpec**.

Parameters

fileSpec	A PDFFileSpec in a document.

Return Value

A **PDDoc** or **NULL** if the file specification **CosObj** is not in a document.

Header File

`PDCalls.h`

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

PDFFileSpecGetFileSys

```
ASFileSys PDFFileSpecGetFileSys (PDFFileSpec fileSpec);
```

Description

Gets the file system that services the specified file specification.

Parameters

fileSpec	The file specification whose file system is obtained.

Return Value

The file system that services **fileSpec**.

Header File

PDCalls.h

Related Methods

[PDFFileSpecGetFileSysName](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDFFileSpecGetFileSysName

```
ASAtom PDFFileSpecGetFileSysName (PDFFileSpec fileSpec);
```

Description

Gets the name of the file system that a **PDFFileSpec** belongs to. For a simple **fileSpec** (string form), the name of the file system is the name of the document's file system—if the **CosObj** that is the **fileSpec** is contained in a document. For a complex **fileSpec** (dict form) with an **FS** key, the name of the file system is the atom associated with the **FS** key.

The file system returned by **PDFFileSpecGetFileSys** is the file system that has registered a **PDFFileSpecHandler** for **fileSpec**'s file system NAME (if there is one), and is not necessarily the same as

```
ASFileGetFileSysByName(PDFFileSpecGetFileSysName(fileSpec));
```

Parameters

fileSpec	A PDFFileSpec .

Return Value

An **ASAtom** representing the file system of **fileSpec**.

Header File

PDCalls.h

Related Methods

[PDFFileSpecGetFileSys](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

PDFFileSpecIsValid

```
ASBool PDFFileSpecIsValid (PDFFileSpec fileSpec);
```

Description

Tests whether a file specification is valid. This is intended only to ensure that the file spec has not been deleted, not to ensure that all necessary information is present and valid.

Parameters

fileSpec	The file specification whose validity is tested.

Return Value

Returns **true** if **fileSpec** is valid, **false** otherwise.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDFFileSpecNewFromASPath

```
PDFFileSpec PDFFileSpecNewFromASPath (PDDoc pdDoc,
ASFileSys fileSys, ASPathName path,
ASPathName relativeToThisPath);
```

Description

Creates a new file specification from the specified **ASPathName**, using the **PDFFileSpecNewFromASPathProc** of the specified file system's file specification handler.

Parameters

pdDoc	The document in which the new file specification will be used.
fileSys	Pointer to an ASFileSysRec specifying the file system responsible for the newly created file specification.
path	The path to convert into a file specification.
relativeToThisPath	Pathname relative to which path is interpreted. If NULL , path is interpreted as an absolute pathname, not a relative pathname.

Return Value

The newly created file spec, or an invalid file spec if the **ASPathName** cannot be converted to a **PDFFileSpec** (use **PDFFileSpecIsValid** to test whether the conversion was successful).

Header File

PDCalls.h

Related Methods

[PDFFileSpecIsValid](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDFont

PDFontAcquireEncodingArray

```
ASUns8** PDFontAcquireEncodingArray (PDFont font);
```

Description

Acquires a font's encoding array (the mapping of character codes to glyphs). When done with this array, call [PDFontEncodingArrayRelease](#) to release it.

The array contains 256 pointers. If a pointer is not **NULL**, it points to a C string containing the name of the glyph for the code point corresponding to the index. If it is **NULL**, then the name of the glyph is unchanged from that specified by the font's built-in encoding.

For a Type 3 font, all glyph names will be present in the encoding array, and **NULL** entries correspond to un-encoded code points.

For non-Roman character set viewers, it is *not* appropriate to call this method.

Parameters

font	The font whose encoding array is obtained.

Return Value

The font's encoding array. Returns **NULL** if there is no encoding array associated with the font.

Header File

PDCalls.h

Related Methods

[PDFontEncodingArrayRelease](#)
[PDFontGetEncodingIndex](#)

Product

reader, base, pro, pdfi

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDFontAcquireXlateTable

ASInt16* PDFontAcquireXlateTable (PDFont font);

Description

Increments the specified font's **XlateTable** reference count and also returns the **XlateTable**, which is a 256-entry table that maps characters from their encoding in the PDF file to host encoding. If a character cannot be mapped to host encoding, then the table entry will (for that character) contain -1. When you are done using the **XlateTable**, call [PDFontXlateTableRelease](#) to release it.

For non-Roman character set viewers, it is *not* appropriate to call this method.

Parameters

font	The font whose XlateTable is obtained.

Return Value

Pointer to the font's **XlateTable**, if any. Otherwise **NULL**.

Header File

PDCalls.h

Related Methods

[PDFontXlateTableRelease](#)
[PDFontXlateString](#)

Product

reader, base, pro, pdfi

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDFontEncodingArrayRelease

```
void PDFontEncodingArrayRelease (ASUns8\*\* array);
```

Description

Releases a font's encoding array (the mapping of character codes to glyphs). Call this method after you are done using an encoding array acquired using [PDFontAcquireEncodingArray](#).

Parameters

array	The encoding array to release.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDFontAcquireEncodingArray](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDFontEnumCharProcs

```
void PDFontEnumCharProcs (PDFont font,
    PDCharProcEnumProc proc, void* clientData);
```

Description

Enumerates a Type 3 font's character drawing procedures. The elements of a single character procedure can be enumerated using [PDCharProcEnum](#).

Parameters

font	The Type 3 font's character drawing procedures are being enumerated.
proc	User-supplied callback to call for each character in the font. Enumeration ends if proc returns false . If the font contains no characters, proc will not be called.
clientData	Pointer to user-supplied data passed to proc each time it is called.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDCharProcEnum](#)

[PDDocEnumFonts](#)

[PDDocEnumLoadedFonts](#)

Example

```
ACCB1 ASBool ACCB2 myPDCharProcEnumProc(
    char* name, PDCharProc obj,
    void* clientData)
{
    ...
}
PDFontEnumCharProcs(font,
    ASCallbackCreateProto(
        PDCharProcEnumProc,
        myPDCharProcEnumProc), NULL);
```

Product

reader, base, pro, pdfi

**Availability**

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDFontFromCosObj

PDFont PDFontFromCosObj (**CosObj** fontObj);

Description

Converts a dictionary Cos object to a font. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

fontObj	Dictionary Cos object for the font.

Return Value

The **PDFont** for **fontObj**. Returns **NULL** if there is no **CosDoc** or **PDDoc** containing **fontObj**.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDFontGetASTextName

```
void PDFontGetASTextName (PDFont font, ASBool removePrefix,
ASText NametoFill) ;
```

Description

Fills in an **ASText** object with the font name, to be used in displaying lists or menus.

In PDF 1.5, the font name can be represented with a UTF8 byte sequence. In previous versions of Acrobat the name could also be represented by host encodings such as Shift-JIS, Big5, KSC, and so on. This routine tries to return a text object that uses the correct script, but cannot always do so.

The **ASText** object is owned by the caller.

Parameters

font	The font whose name is obtained.
removePrefix	Whether to remove the subset prefix, if present. For example, when true , the name "ABCDEF+Myriad" is returned as "Myriad."
NameToFill	(Filled by the method) The ASText object for the font's name.

Return Value

None

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDFontGetBBox

```
void PDFontGetBBox (PDFont font, ASFixedRect* bboxP);
```

Description

Gets a Type 3 font's bounding box, which is the smallest rectangle that would enclose every character in the font if they were overlaid and painted.

Parameters

font	The font whose bounding box is obtained.
bboxP	<i>(Filled by the method)</i> Pointer to a rectangle specifying the font's bounding box.

Return Value

None

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDFontGetCharSet

```
PDCharSet PDFontGetCharSet (PDFont font);
```

Description

Gets the font's character set. This is derived from the "Uses Adobe standard encoding" bit in the font descriptor (if the font has a font descriptor) or from the font's name (if the font is one of the base 14 fonts and does not have a font descriptor).

For non-Roman character set viewers, call [PDFontGetEncodingName](#) instead.

Parameters

font	The font whose character set is obtained.

Return Value

The font's character set. For non-Roman character set viewers, returns **PDUnknownCharSet**.

Header File

PDFcalls.h

Related Methods

[PDFontGetEncodingName](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDFontGetCIDSystemInfo

[ASAtom](#) PDFontGetCIDSystemInfo ([PDFont](#) font);

Description

Gets an **ASAtom** representing Registry and Ordering for a CIDFont. This information resides in the **CIDSystemInfo** entry of the CIDFont dictionary, which describes a CIDFont.

PDFontGetCIDSystemInfo takes either a Type 0 font or descendant font (CIDType0 or CIDType2) as an argument. This information is always present for any Type 0 font; the actual registry ordering information is a part of the descendant font.

This method provides one way to identify a font's language.

The **CIDSystemInfo** entry uses three components to identify a character collection uniquely:

- A registry name to identify an issuer of ordering information.
- An ordering name to identify an ordered character collection.
- A supplement number to indicate that the ordered character collection for a registry, ordering, and previous supplement has been changed to add new characters assigned CIDs beginning with the next available CID.

The **PDFontGetCIDSystemInfo** method obtains the first two of these components.

A CIDFont is designed to contain a large number of glyph procedures. Instead of being accessed by a name, each glyph procedure is accessed by an integer known as a character identifier or CID. Instead of a font encoding, CIDFonts use a CMap with a Type 0 composite font to define the mapping from character codes to a font number and a character selector.

For more information on Type 0 fonts, CIDFonts, and CMaps, see Section 5.6 in the [PDF Reference](#). For detailed information on CIDFonts, see Technical Note #5092, *CID-Keyed Font Technology Overview*, and Technical Note #5014, *Adobe CMap and CIDFont Files Specification*.

Parameters

font	The font whose Registry and Ordering information is obtained.

Return Value

The **ASAtom** representing the CIDFont's Registry and Ordering information, for example, "Adobe-Japan1."

Header File

PDCalls.h

Related Methods

[PDFontGetCIDSystemSupplement](#)
[PDFontGetDescendant](#)

**Product**

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020003** or higher.

PDFontGetCIDSystemSupplement

ASInt32 PDFontGetCIDSystemSupplement (**PDFont** font);

Description

Gets the SystemSupplement number of a CIDFont. This field resides in the **CIDSystemInfo** entry of the CIDFont dictionary, which describes a CIDFont.

The **CIDSystemInfo** entry uses three components to identify a character collection uniquely:

- A registry name to identify an issuer of orderings.
- An ordering name to identify an ordered character collection.
- A supplement number to indicate that the ordered character collection for a registry, ordering, and previous supplement has been changed to add new characters assigned CIDs beginning with the next available CID.

PDFontGetCIDSystemInfo provides character collection information, and **PDFontGetCIDSystemSupplement** specifies the version of the ordering.

A CIDFont is designed to contain a large number of glyph procedures. Instead of being accessed by a name, each glyph procedure is accessed by an integer known as a character identifier or CID. Instead of a font encoding, CIDFonts use a CMap with a Type 0 composite font to define the mapping from character codes to a font number and a character selector.

For more information on Type 0 fonts, CIDFonts, and CMaps, see Section 5.6 in the [PDF Reference](#). For detailed information on CIDFonts, see Technical Note #5092, *CID-Keyed Font Technology Overview*, and Technical Note #5014, *Adobe CMap and CIDFont Files Specification*.

Parameters

font	The font whose SystemSupplement field is obtained.

Return Value

The SystemSupplement field from the CIDFont.

Header File

PDCalls.h

Related Methods

[PDFontGetDescendant](#)
[PDFontGetCIDSystemInfo](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020003** or higher.



PDFontGetCosObj

[CosObj](#) PDFontGetCosObj ([PDFont](#) font);

Description

Gets the Cos object for a font. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

font	The font whose Cos object is obtained.

Return Value

The dictionary Cos object for the font. The dictionary's contents may be enumerated with [CosObjEnum](#).

Header File

PDcalls.h

Related Methods

[PDDocEnumFonts](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDFontGetDescendant

[PDFont](#) PDFontGetDescendant ([PDFont](#) font);

Description

Gets a Type 0 font's descendant, which may be a CIDType0 or CIDType2 font.

Type 0 fonts support single-byte or multi-byte encodings and can refer to one or more *descendant fonts*. These fonts are analogous to the Type 0 or composite fonts supported by Level 2 PostScript interpreters. However, PDF Type 0 fonts only support character encodings defined by a CMap. The CMap specifies the mappings between character codes and the glyphs in the descendant fonts.

For information on Type 0 fonts, see Section 5.6 in the [PDF Reference](#). See Section 5.6.4 for more details on CMaps.

Parameters

font	The font whose descendant is obtained.

Return Value

The font's descendant font.

Header File

PDFcalls.h

Related Methods

[PDFontGetCIDSystemInfo](#)

[PDFontGetCIDSystemSupplement](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020003** or higher.

PDFontGetEncodingIndex

ASInt32 PDFontGetEncodingIndex (**PDFont** font);

Description

Gets a font's encoding index.

For non-Roman character set viewers, it is *not* appropriate to call this method. Call **PDFontGetEncodingName** instead.

Parameters

font	The font whose encoding index is obtained.

Return Value

A font encoding index. If the index is greater than **PDLastKnownEncoding**, it is a custom encoding, and is unique within the document. If the index is less than **PDLastKnownEncoding**, it must be one of the **PDFontEncoding** values.

Header File

PDCalls.h

Related Methods

PDFontAcquireEncodingArray

Product

reader, base, pro, pdfi

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDFontGetEncodingName

```
const ASUns8* PDFontGetEncodingName (PDFont font);
```

Description

Gets a string representing a font's encoding.

Use [PDFontGetEncodingIndex](#) to get encoding information for Roman viewers.

Host encoding is a platform-dependent encoding for the host machine. For non-UNIX Roman systems, it is **MacRomanEncoding** in Mac OS and **WinAnsiEncoding** in Windows. In UNIX (except HP-UX) Roman systems, it is ISO8859-1 (ISO Latin-1); for HP-UX, it is **HP-ROMAN8**. See Appendix D in the [PDF Reference](#) for descriptions of **MacRomanEncoding**, **WinAnsiEncoding**, and **PDFDocEncoding**.

For non-Roman systems, the host encoding may be a variety of encodings, which are defined by a **CMap** (character map). See Section 5.6.4 in the [PDF Reference](#) for a list of predefined **CMaps**. In this case, the encoding string contains values such as "90ms-RKSJ-H", "90msp-RKSJ-H", "Identity-V", or "90pv-RKSJ-H"; it does not return a string like "Shift-JIS".

Parameters

font	The font whose encoding name is obtained.

Return Value

String representing the font's encoding.

Header File

`PDCalls.h`

Related Methods

[PDFontGetEncodingIndex](#)
[PDGetHostEncoding](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020003** or higher.

PDFontGetFontMatrix

```
void PDFontGetFontMatrix (PDFont fontP,  
ASFixedMatrix* matrixP);
```

Description

Gets a font's matrix, which specifies the transformation from character space to text space. See Section 5.5.4 in the [PDF Reference](#). This is only valid for Type 3 fonts.

Parameters

font	The font whose matrix is obtained.
matrixP	<i>(Filled by the method)</i> Pointer to the font's matrix.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocEnumFonts](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDFontGetMetrics

```
void PDFontGetMetrics (PDFont font, PDFontMetricsP metricsP,
os_size_t sizeMetrics);
```

Description

Gets a font’s metrics, which provide the information needed to create a substitute multiple master font when the original font is unavailable. See Section 5.7 in the [PDF Reference](#) for a discussion of font descriptors.

Parameters

font	The font whose metrics are being obtained.
metricsP	<i>(Filled by the method)</i> Pointer to a PDFontMetrics structure containing the font’s metrics . The font metric values may be patched before being returned. If the actual values in the PDF file are required, use Cos instead to get trustworthy metrics.
sizeMetrics	Must be <code>sizeof (PDFontMetrics)</code> .

Return Value

None

Header File

PDCalls.h

Related Methods

[PDFontGetWidths](#)
[PDFontSetMetrics](#)

Example

```
PDWordFinder wObj;
PDWord pdWord;
PDStyle style;
PDFont font;
PDFontMetrics fMetrics;

/* Get font metric info */
style = PDWordGetNthCharStyle(wObj, pdWord, 0 );
font = PDStyleGetFont(style);
PDFontGetMetrics (font, &fMetrics, sizeof (PDFontMetrics));
```

Product

reader, base, pro, pdf

**Availability**

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDFontGetName

```
ASInt32 PDFontGetName (PDFont font, char* buffer,
ASInt32 bufSize);
```

Description

Gets the name of a font. The behavior depends on the font type; for a Type 3 font it gets the value of the **Name** key in a PDF Font resource. See Section 5.5.4 in the [PDF Reference](#). For other types it gets the value of the **BaseFont** key in a PDF font resource.

Parameters

font	The font whose name is obtained.
buffer	<i>(Filled by the method)</i> Buffer into which the font's name is stored. The client may pass in NULL to obtain the buffer size, then call the method with a buffer of the appropriate size.
bufSize	Length of buffer , in bytes. The maximum font name length that the Acrobat viewer will return is PSNAME_SIZE (see <code>PDExpT.h</code>).

Return Value

The number of characters in the font name. If the font name is too long to fit into **buffer**, **bufSize** - 1 bytes are copied into **buffer**, and **buffer** is null-terminated.

Header File

```
PDCalls.h
```

Related Methods

```
PDDocEnumFonts
PDFontGetEncodingName
```

Example

```
PDWord pdWord;
PDFont font;
PDStyle style = PDWordGetNthCharStyle(pdWF, pdWord, 0);
size = PDStyleGetFontSize(style);
PDFontGetName(font, mbuf, sizeof(mbuf));
if(size == fixedTen && !strstr(mbuf,"Italic"))
{ /* process the 10 pt. Italic */
    ...
}
```

Product

```
reader, base, pro, pdf
```



Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDFontGetSubtype

```
ASAtom PDFontGetSubtype (PDFont font);
```

Description

Gets a font's subtype.

Parameters

font	The font whose subtype is obtained.

Return Value

subtype

The font's subtype. The **ASAtom** returned can be converted to a string using [ASAtomGetString](#).

Must be one of the [Font Subtypes](#).

Header File

PDFcalls.h

Related Methods

[PDDocEnumFonts](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDFontGetWidths

```
void PDFontGetWidths (PDFont font, ASInt16* widths);
```

Description

Gets the advance width of every glyph in a font. The advance width is the amount by which the current point advances when the glyph is drawn. The advance width may not correspond to the visible width of the glyph (for example, a glyph representing an accent mark might have an advance width of zero so that characters can be drawn under it). For this reason, the advance width cannot be used to determine the glyphs' bounding boxes.

For non-Roman character set viewers, this method gets the width for a single byte range (0 through 255).

Parameters

font	The font whose glyph advance widths are obtained.
widths	<i>(Filled by the method)</i> An array of glyph advance widths, measured in character space units. Un-encoded code points will have a width of zero. For non-Roman character set viewers, an array for a single byte range (0 through 255).

Header File

PDCalls.h

Related Methods

[PDFontGetMetrics](#)
[PDFontSetMetrics](#)
[PDFontXlateWidths](#)

Example

```
PDWordFinder wObj;  
PDWord pdWord;  
PDStyle style;  
PDFont font;  
PDFontMetrics fMetrics;  
ASInt16 widths[256];  
/* Get font width info */  
style = PDWordGetNthCharStyle(wObj, pdWord, 0);  
font = PDStyleGetFont(style);  
PDFontGetWidths(font, widths);
```

Product

reader, base, pro, pdf

**Availability**

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDFontIsEmbedded

ASBool PDFontIsEmbedded (**PDFont** font);

Description

Tests whether the specified font is embedded in the PDF file (that is, the font is stored as a font file, which is a stream embedded in the PDF file). Only Type 1 and TrueType fonts can be embedded.

Parameters

font	The font to test.

Return Value

Returns **true** if the font is embedded in the file, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDDocEnumFonts](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDFontSetMetrics

```
void PDFontSetMetrics (PDFont font, PDFontMetricsP metricsP,
os_size_t sizeMetrics);
```

Description

Sets a font’s metrics, which provide the information needed to create a substitute multiple master font when the original font is unavailable. See Section 5.7 in the [PDF Reference](#) for a discussion of font descriptors. This method can only be used on Type 1, multiple master Type 1, and TrueType fonts; it cannot be used on Type 3 fonts.

Parameters

font	The font whose metrics are being set.
metricsP	Pointer to a PDFontMetrics structure containing the font’s metrics.
sizeMetrics	Must be <code>sizeof(PDFontMetrics)</code> .

Return Value

None

Header File

PDCalls.h

Related Methods

[PDFontGetMetrics](#)

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDFontXlateString

```
ASBool PDFontXlateString (PDFont font, ASUns8* inP,
ASUns8* outP, ASInt32 len);
```

Description

Translates a string from the **PDFont**'s encoding into host encoding. If any characters cannot be represented in host encoding, they are replaced with space characters. If no **XlateTable** exists in the font, the function returns **false** and **outP** is not written.

For non-Roman character set viewers, it is *not* appropriate to call this method. Instead call one of the following: **PDFontXlateToHost**, **PDFontXlateToUCS**, **PDXlateToHostEx**, or **PDXlateToPDFDocEncEx**.

Parameters

font	The font (and hence, encoding) that inP uses.
inP	The string to translate.
outP	<i>(Filled by the method)</i> The translated string. outP may point to the same buffer as inP , to allow in-place translation.
len	The length of inP and outP .

Return Value

Returns **true** if an **XlateTable** exists in the font, **false** otherwise. If no **XlateTable** exists in the font, **outP** is not written.

Header File

PDCalls.h

Related Methods

PDFontXlateToHost
PDFontAcquireXlateTable
PDFontXlateToUCS
PDXlateToHostEx
PDXlateToPDFDocEncEx

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDFontXlateTableRelease

```
void PDFontXlateTableRelease (ASInt16* table);
```

Description

Decrements the specified font's **XlateTable** reference count. The **XlateTable** is a 256-entry table that maps characters from their encoding in the PDF file to host encoding. If a character cannot be mapped to host encoding, then the table entry will (for that character) contain -1.

Parameters

table	The XlateTable to release.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDFontAcquireXlateTable](#)
[PDFontXlateString](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDFontXlateToHost

```
ASInt32 PDFontXlateToHost (PDFont fontP, ASUns8* inP,
ASInt32 inLen, ASUns8* outP, ASInt32 outLen);
```

Description

Translates a string from the **PDFont**'s encoding to host encoding. This is useful for converting the text from a **PDWord** into host encoding. In the same way that **PDxlateToHostEx** converts text from bookmark titles to host encoding, **PDFontXlateToHost** converts text from a page contents stream to host encoding. Use **PDFontXlateToUCS** to translate from the **PDFont**'s encoding to Unicode.

Non-Roman fonts, such as PostScript composite fonts, can be encoded in different ways, such as SHIFT-JIS, RKSJ, and so on. To use **PDFontXlateToHost**, the caller does not need to know which encoding they're converting from, since that information is contained in the **PDFont**.

Host encoding is a platform-dependent encoding for the host machine. For non-UNIX Roman systems, it is **MacRomanEncoding** in Mac OS and **WinAnsiEncoding** in Windows. In UNIX (except HP-UX) Roman systems, it is ISO8859-1 (ISO Latin-1); for HP-UX, it is **HP-ROMAN8**. See Appendix D in the [PDF Reference](#) for descriptions of **MacRomanEncoding**, **WinAnsiEncoding**, and **PDFDocEncoding**.

For non-Roman systems, the host encoding may be a variety of encodings, which are defined by a **CMap** (character map). See Section 5.6.4 in the [PDF Reference](#) for a list of predefined **CMaps**.

Use **PDGetHostEncoding** to determine if a system's host encoding is Roman or not.

Parameters

fontP	The font used in the input string inP .
inP	Pointer to the string to translate.
inLen	Length of the inP buffer, in bytes.
outP	(Filled by the method) Pointer to the translated string.
outLen	The length of the outP buffer, in bytes.

Return Value

Number of bytes in the translated string **outP**.

Header File

PDCalls.h

Related Methods

[PDFontXlateString](#)

PDFontXlateToUCS
PDGetHostEncoding
PDXlateToHostEx
PDXlateToPDFDocEncEx

Example

```
ASInt16 newlen;
ASUns8 stackBuffer[1024];
ASUns8* buffer = stackBuffer;

newlen = PDFontXlateToHost(font, string, stringlen, buffer,
sizeof(stackBuffer) - 1);

if (newlen == sizeof(stackBuffer) - 1) {
    newlen = PDFontXlateToHost(font, string,
        stringlen, NULL, 0);
    buffer = ASmalloc(newlen + 1);
    newlen = PDFontXlateToHost(font, string,
        stringlen, buffer, newlen);
}

buffer[newlen] = '\0';
...
if (buffer != stackBuffer)
    ASfree(buffer);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020003** or higher.

PDFontXlateToUCS

```
ASInt32 PDFontXlateToUCS (PDFont fontP, ASUns8* inP,
ASInt32 inLen, ASUns8* outP, ASInt32 outLen);
```

Description

Translates a string from whatever encoding the **PDFont** uses to Unicode encoding. This is useful for converting the text from a **PDWord** into Unicode. Use **PDFontXlateToHost** to translate from the **PDFont**'s encoding to host encoding.

Non-Roman fonts, like PostScript composite fonts, can be encoded in different ways, such as SHIFT-JIS, RKSJ, and so on. The caller does not need to know which encoding they're converting from, since that information is contained in the **PDFont**.

Host encoding is a platform-dependent encoding for the host machine. For non-UNIX Roman systems, it is **MacRomanEncoding** in Mac OS and **WinAnsiEncoding** in Windows. In UNIX (except HP-UX) Roman systems, it is ISO8859-1 (ISO Latin-1); for HP-UX, it is **HP-ROMAN8**. See Appendix D in the *PDF Reference* for descriptions of **MacRomanEncoding**, **WinAnsiEncoding**, and **PDFDocEncoding**.

For non-Roman systems, the host encoding may be a variety of encodings, which are defined by a **CMap** (character map). See Section 5.6.4 in the *PDF Reference* for a list of predefined **CMaps**.

Use **PDGetHostEncoding** to determine if a system's host encoding is Roman or not.

Parameters

fontP	The font of the input string inP .
inP	Pointer to the string to translate.
inLen	Length of the inP buffer, in bytes.
outP	(Filled by the method) Pointer to the translated string. If NULL , the method returns the size of the translated string.
outLen	The length of the outP buffer, in bytes. If 0, the method returns the size of the translated string.

Return Value

Number of bytes in the translated string in **outP**.

Header File

PDCalls.h

Related Methods

[PDFontXlateToHost](#)
[PDGetHostEncoding](#)

PDFontXlateToHostEx **PDFontXlateToPDFDocEncEx**

Example

```
ASInt32 newlen;
ASUns8 stackBuffer[1024];
ASUns8* buffer = stackBuffer;

newlen = PDFontXlateToUCS(font, string, stringlen, buffer,
sizeof(stackBuffer) - 2);

if (newlen == sizeof(stackBuffer) - 2) {
    newlen = PDFontXlateToUCS(font, string,
        stringlen, NULL, 0);
    buffer = ASmalloc(newlen + 2);
    newlen = PDFontXlateToUCS(font, string,
        stringlen, buffer, newlen);
}

buffer[newlen] = '\\0';
buffer[newlen+1] = '\\0';
...
if (buffer != stackBuffer)
    ASfree(buffer);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020003** or higher.

PDFontXlateWidths

```
void PDFontXlateWidths (PDFont font, ASInt16* inP,
ASInt16* outP);
```

Description

Translates an array of 256 glyph advance widths (obtained from [PDFontGetWidths](#)) from their order in the PDF file into host encoding order. If the widths are already in host encoding order, the widths are merely copied. All un-encoded code points are given a width of zero.

For non-Roman character set viewers, it is *not* appropriate to call this method.

Parameters

font	The font whose glyph widths are translated.
inP	Array of glyph advance widths to rearrange.
outP	(Filled by the method) Rearranged array of glyph advance widths.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDFontGetWidths](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDForm

PDFormEnumPaintProc

```
void PDFormEnumPaintProc (PDXObject obj,
    PDGraphicEnumMonitor mon, void* clientData);
```

Description

(Obsolete, provided only for backwards compatibility) Enumerates a form's drawing operations.

Parameters

obj	The form whose drawing operations are enumerated.
mon	Structure containing user-supplied callbacks that are called for each drawing operator on a page. Enumeration ends if any procedure returns false .
clientData	Pointer to user-supplied data to pass to mon each time it is called.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDFormEnumResources](#)
[PDFormEnumPaintProcWithParams](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDFormEnumPaintProcWithParams

```
void PDFormEnumPaintProcWithParams (PDXObject obj,  
PDGraphicEnumParams params)
```

Description

Enumerates a form's drawing operations for those contents that are visible in a given optional-content context. The parameters include both the monitor and data you would pass to [PDFormEnumPaintProc](#), and an optional-content context that determines which contents are visible.

Parameters

obj	The form whose drawing operations are enumerated.
params	The parameters, including the optional-content context to use for content visibility.

Return Value

None

Known Exceptions

[pdPErrUnableToCreateRasterPort](#)

Header File

PDCalls.h

Related Methods

[PDFormEnumPaintProc](#)
[PDCharProcEnumWithParams](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDFormEnumResources

```
void PDFormEnumResources (PDXObject obj,  
    PDResourceEnumMonitor mon, void* clientData);
```

Description

(*Obsolete, provided only for backwards compatibility*) Enumerates the resources used by a form.

Parameters

obj	The form whose resources are enumerated.
mon	Structure containing user-supplied callbacks that are called for each of the form's resources. Enumeration ends if any procedure returns false .
clientData	Pointer to user-supplied data to pass to mon each time it is called.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDFormEnumPaintProc](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDFormGetBBox

```
void PDFormGetBBox (PDXObject obj, ASFixedRect* bboxP);
```

Description

(Obsolete, provided only for backwards compatibility) Gets a form's bounding box.

Parameters

obj	The form whose bounding box is obtained.
bboxP	<i>(Filled by the method)</i> Pointer to a rectangle containing the form's bounding box, specified in user space coordinates.

Return Value

None

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDFormGetFormType

```
ASInt32 PDFormGetFormType (PDXObject obj);
```

Description

*(Obsolete, provided only for backwards compatibility)*Gets the value of a form's **FormType** attribute.

Parameters

obj	Form whose type is obtained.

Return Value

Form type (value of the PDF **FormType** key). This value is 1 for PDF 1.0, 1.1, and 1.2.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDFormGetMatrix

```
void PDFormGetMatrix (PDXObject obj, ASFixedMatrix* matrixP);
```

Description

(Obsolete, provided only for backwards compatibility) Gets the specified form's transformation matrix.

Parameters

obj	The form whose transformation matrix is obtained.
matrixP	<i>(Filled by the method)</i> Pointer to a matrix containing the form's transformation matrix, which specifies the transformation from form space to user space. See Section 4.9 in the PDF Reference .

Return Value

None

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDFormGetXUIDCosObj

```
CosObj PDFormGetXUIDCosObj (PDXObject obj);
```

Description

(Obsolete, provided only for backwards compatibility) Gets the array Cos object corresponding to a form's **XUID**. An **XUID** is an array of numbers that uniquely identify the form in order to allow it to be cached.

Parameters

obj	The form whose XUID is obtained.

Return Value

The array Cos object for the form's **XUID**.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDGraphic

PDGraphicGetBBox

```
void PDGraphicGetBBox (PDGraphic obj, ASFixedRect* bboxP);
```

Description

Gets a bounding box for the specified graphic object.

Parameters

obj	The graphic object whose bounding box is obtained.
bboxP	(Filled by the method) Pointer to a rectangle containing the bounding box for obj.If called during PDFormEnumPaintProc or PDCharProcEnum, the coordinates are specified in the object space (that is, relative to the object’s matrix).

Return Value

None

Header File

PDCalls.h

Related Methods

- PDCharProcEnum
- PDFormEnumPaintProc

Product

reader, base, pro, pdf

Availability

Available if PI_PDMODEL_VERSION (in PIRequir.h) is set to 0x00020000 or higher.



PDGraphicGetCurrentMatrix

```
void PDGraphicGetCurrentMatrix (PDGraphic obj,
ASFixedMatrix* matrix);
```

Description

Gets the current transformation matrix in effect for a graphic object; the matrix is relative to user space.

Parameters

obj	The graphic object for which transformation matrix is obtained.
matrix	<i>(Filled by the method)</i> Pointer to a matrix containing the transformation matrix for obj .

Return Value

None

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDGraphicGetState

```
void PDGraphicGetState (PDGraphic obj, PDGraphicStateP stateP,  
ASInt32 stateLen);
```

Description

Gets the graphics state associated with a graphic object. See Section 4.3 in the [PDF Reference](#) for a discussion of the graphics state parameters.

Parameters

obj	The graphic object whose graphics state is obtained.
stateP	(Filled by the method) Pointer to a PDGraphicState structure containing the graphics state.
stateLen	Must be sizeof (PDGraphicsState).

Return Value

None

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDImage

PDImageColorSpaceGetIndexLookup

```
void PDImageColorSpaceGetIndexLookup (PDXObject obj,
ASUns8* data, ASInt32 dataLen);
```

Description

(Obsolete, provided only for backwards compatibility) Gets the lookup table for an indexed color space. The table will contain the number of entries specified by the index size, and there will be 1 byte for each color component for each entry. The number of color components depends on the color space: gray→1, RGB→3, CMYK→4, Lab→3. For additional information on indexed color space, see Section 4.5.5 in the [PDF Reference](#). There is also some useful discussion in the *PostScript Language Reference Manual, Third Edition*, under indexed color spaces.

Parameters

obj	The image whose lookup table is obtained.
data	<i>(Filled by the method)</i> Array for returned color space information.
dataLen	Length of data , in bytes.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDInlineImageColorSpaceGetIndexLookup](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDImageGetAttrs

```
void PDImageGetAttrs (PDXObject obj, PDImageAttrsP attrsP,
ASInt32 attrsLen);
```

Description

(Obsolete, provided only for backwards compatibility. Use PDEImageGetAttrs and/or Cos-level calls instead.) Gets the attributes of an image (for example, Type, Subtype, Name, Width, Height, BitsPerComponent, ColorSpace, Decode, Interpolate, ImageMask).

Parameters

obj	The image whose attributes are obtained.
attrsP	<i>(Filled by the method)</i> Pointer to a PDImageAttrs structure containing the image attributes.
attrsLen	Must be <code>sizeof (PDImageAttrs)</code> .

Return Value

None

Header File

PDCalls.h

Related Methods

[PDInlineImageGetAttrs](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDImageSelectAlternate

```
CosObj PDImageSelectAlternate (CosObj image, ASBool print,
ASUns32 tickLimit, ASBool* cacheImageP, void* callData);
```

Description

Selects which Alternate image to use. This method can do one of three things:

- Return an existing Alternate
- Create an image XObject and return that
- Indicate that this XObject should be skipped

This method is called each time the Acrobat viewer draws an XObject image, regardless of whether the image XObject has an Alternates key.

You can replace this method with your own version, using [HFTReplaceEntry](#).

Parameters

image	The Cos object for this image XObject's base image. Under some circumstances, PDImageSelectAlternate can be called with an XObject type other than an image; for example, a form. Because of this, your code <i>must</i> check the XObject 's subtype (you can use CosDictGet to read the value of the XObject 's Subtype key). If the subtype is not Image , your code must not modify the XObject , but simply return the CosObj that was passed to you.
print	true if printing, false if displaying.
tickLimit	Amount of time, in ticks, before the image selector must return control to the Acrobat viewer. This parameter is not relevant for image selectors that simply choose an existing alternate or create a new image XObject using Cos methods, but it is relevant for image selectors that create a new image XObject by calculating or reading data from a network or a slow device. • If tickLimit is zero, the image selector must not return control to the Acrobat viewer until the selector can provide the alternate image to use. • If tickLimit is nonzero, the image selector does not have to provide the image XObject within tickLimit, but it must raise the fileErrBytesNotReady exception if it cannot. This returns control to the Acrobat viewer and informs it that the data is not ready. The Acrobat viewer then calls the image selector periodically until the image selector returns the selected image XObject.

cacheImageP	If true, the image data returned to the Acrobat viewer is cached for future use. If false, it is not. Pass true if you expect your image data to remain valid for at least several calls to your client. Pass false if you expect your image data to change on the next call to your client. If you create and return a Cos stream object with an external file system and the stream's data will not always be the same, you must set cacheImageP to false. If you fail to do this, the Acrobat viewer may use cached image data when it should not.
callData	An opaque pointer containing data that must be passed in other image selector calls.

Return Value

The image XObject to use. Returning a **NULL** Cos object (obtained using [CosNewNull](#)) tells the Acrobat viewer to skip this XObject entirely.

Known Exceptions

[fileErrBytesNotReady](#)

Header File

PDCalls.h

Related Methods

[PDImageSelGetGeoAttr](#)
[PDImageSelGetDeviceAttr](#)
[PDImageSelAdjustMatrix](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDImageSelAdjustMatrix

```
void PDImageSelAdjustMatrix (void* callData,
    ASFixedMatrix newUserMatrix);
```

Description

Allows an image selector client to change the region of the page occupied by an image. It must only be used by image selector clients that return data for *only the visible part* of an image—to set the region of the page that the sub-image occupies. It must not be used otherwise.

This method only has an effect while displaying on screen. It does nothing when printing.

The matrix set by this call remains in effect only for the current image. The Acrobat viewer automatically replaces it after the image has been drawn.

Parameters

callData	The value passed to the image selector as a parameter to PDImageSelectAlternate .
newUserMatrix	The matrix that will replace the imageToUserMatrix (see PDImageSelGetGeoAttr). The imageToDevMatrix is automatically calculated from newUserMatrix .

Return Value

None

Header File

PDCalls.h

Related Methods

[PDImageSelGetGeoAttr](#)
[PDImageSelGetDeviceAttr](#)
[PDImageSelectAlternate](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDImageSelGetDeviceAttr

```
void PDImageSelGetDeviceAttr (void* callData,
    PDColorSpace* colorSpaceP, ASUns32* bitsPerPixelP,
    ASAtom* deviceTypeP);
```

Description

Gets device-related attributes for a particular image XObject. It must only be used from within an image selector client, since it returns information that is only valid in that context. This method can be used by an image selector client to obtain additional information to help the selector determine which alternate image to choose.

If an image is displayed on two devices simultaneously (for example, if the window containing the image is split across two monitors in a multi-monitor system), the values returned for **colorSpaceP** and **bitsPerPixelP** are each the maximum for the devices on which the image is currently displayed. For example, if the image is currently split across the following devices—

- 8-bit gray scale monitor
- 4-bit RGB color monitor

colorSpaceP would be **DeviceRGB**, because that is the “highest” colorspace on which the image is currently displayed.

bitsPerPixelP would be 8, because that is the highest bit depth on which the image is currently displayed.

Parameters

callData	<i>(Filled by the method)</i> The value passed to the image selector as a parameter to PDImageSelectAlternate .
colorSpaceP	<i>(Filled by the method)</i> Destination device’s colorspace. Will be one of the following: PDDeviceGray — Grayscale device PDDeviceRGB — RGB device PDDeviceCMYK — CMYK device If the device has some other color space or its color space cannot be determined, PDDeviceRGB is returned.
bitsPerPixelP	<i>(Filled by the method)</i> Number of bits used for each pixel. For example, a device with 8 bits red, 8 bits green, and 8 bits blue would have a bitsPerPixel of 24. If bitsPerPixelP has a value of 0 (instead of the more standard 1, 8, or 24), the number of bits per pixel on the output device could not be determined.

deviceTypeP	(Filled by the method) Output device type. Will be one of the following: Display — A display device such as a monitor PostScript — A PostScript printer or PostScript file nonPostScriptPrinter — a non-PostScript printer

Return Value

None

Header File

PDCalls.h

Related Methods

[PDImageSelAdjustMatrix](#)
[PDImageSelGetDeviceAttr](#)
[PDImageSelectAlternate](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDImageSelGetGeoAttr

```
void PDImageSelGetGeoAttr (void* callData,
    ASFixedRect* updateBBoxP,
    ASFixedMatrix* imageToDefaultMatrixP,
    ASFixedMatrix* imageToDevMatrixP);
```

Description

Requests geometry-related attributes of an image **XObject**. This method can be used by an image selector client to obtain additional information to help the selector determine which alternate image to choose.

Parameters

callData	(Filled by the method) The value passed to the image selector as a parameter to PDImageSelectAlternate .
updateBBoxP	(Filled by the method) Rectangle bounding the region of the page to update. This is the intersection of the visible portion of the page, any update regions, and any clipping paths that have been explicitly set in the PDF file. Its coordinates are specified in default user space.
imageToDefaultMatrix	(Filled by the method) Matrix specifying the transformation from image space (the space in which all images are 1x1 units) to default user space (the space with 72 units per inch). It contains sufficient information for the image selector plug-in to determine the image's size and location on the page. For a "normal page and image" (an "upright" image on a non-rotated page), the following is true: imageToDefaultMatrixP->a = image horizontal size in 1/72 of an inch units imageToDefaultMatrixP->b = 0 imageToDefaultMatrixP->c = 0 imageToDefaultMatrixP->d = image vertical size in 1/72 of an inch units imageToDefaultMatrixP->h = left edge of image imageToDefaultMatrixP->v = bottom edge of image

imageToDefaultMatrix (cont'd)	In other words, this matrix provides the image's height, width, and position on the page, all in units of points (compare to imageToDefaultMatrixP). The intersection of the rectangle obtained by transforming a 1x1 unit rectangle (the image) through imageToDeviceMatrixP and the updateBBoxP rectangle is the region of the image that is actually drawn. This is the region of the image for which data is required.
imageToDevMatrixP	<i>(Filled by the method)</i> Matrix specifying the transformation from image space (the space in which all images are 1x1 unit) to device space (the space in which one unit is one pixel). This matrix provides the image's height, width, and position on the page, all in pixels (compare to imageToDefaultMatrixP).

Return Value

None

Header File

PDCalls.h

Related Methods

[PDImageSelAdjustMatrix](#)
[PDImageSelGetDeviceAttr](#)
[PDImageSelectAlternate](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDInlineImage

PDInlineImageColorSpaceGetIndexLookup

```
void PDInlineImageColorSpaceGetIndexLookup
(PDInlineImage image, ASUns8* data, ASInt32 dataLen);
```

Description

Gets the lookup table for an indexed color space. The table will contain the number of entries specified by the index size, and there will be 1 byte for each color component for each entry. The number of color components depends on the color space: gray→1, RGB→3, CMYK→4, Lab→3. For additional information on indexed color space, see Section 4.5.5 in the [PDF Reference](#). There is also some useful discussion in the *PostScript Language Reference Manual, Third Edition* under indexed color spaces.

Parameters

image	The inline image whose lookup table is obtained.
data	(Filled by the method) The lookup table for image .
dataLen	Length of data , in bytes.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDImageColorSpaceGetIndexLookup](#)

Product

reader, base, pro, pdfi

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDInlineImageGetAttrs

```
void PDInlineImageGetAttrs (PDInlineImage obj,
    PDImageAttrsP attrsP, ASInt32 attrsLen);
```

Description

Gets an inline image’s attributes.

NOTE: This method is provided only for backwards compatibility. It has not been updated beyond PDF Version 1.1 and may not work correctly for newly created PDF 1.2 or later files. You should use the PDFEdit API to enumerate page contents.

NOTE: The attribute for a color space is a **CosObj**. Cos objects that are the result of parsing inline dictionaries in the PDF page contents are a special class of Cos objects. You should never depend on these objects lasting the lifetime of the document. You should extract the information you need from the object immediately and refer to it no further in your code.

Parameters

obj	The inline image whose attributes are obtained.
attrsP	(Filled by the method) Pointer to a PDImageAttrs structure containing the image attributes. NOTE: This structure contains a Cos object that is subject to the warning above.
attrsLen	Must be sizeof (PDImageAttrs) .

Return Value

None

Header File

PDCalls.h

Related Methods

- [PDImageGetAttrs](#)
- [PDInlineImageGetData](#)



Example

```
PDImageAttrs iattrs;
ASUns8* bitsData;

PDInlineImageGetAttrs(obj,&iattrs,
sizeof(PDImageAttrs));

if(NULL == (bitsData =
(ASUns8 *)ASmalloc(iattrs.dataLen)) ){
    AVAlertNote("Serious Memory Allocation Error");
    return false;
}

PDInlineImageGetData(obj,bitsData,
iattrs.dataLen);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDInlineImageGetData

```
void PDInlineImageGetData (PDInlineImage obj, ASUns8* data,
ASInt32 dataLen);
```

Description

Gets the image data for an inline image.

Parameters

obj	The inline image whose data is obtained.
data	<i>(Filled by the method)</i> A buffer into which the image data will be placed.
dataLen	Number of bytes that data can hold. Must be large enough to hold the entire inline image. Use PDInlineImageGetAttrs to determine how much data is in the image.

Return Value

None

Known Exceptions

Raises [genErrBadParm](#) if **dataLen** < the amount of data in the image.

Header File

PDCalls.h

Related Methods

[PDInlineImageGetAttrs](#)

Example

```
PDImageAttrs iattrs;
ASUns8* bitsData;

PDInlineImageGetAttrs(obj,&iattrs,sizeof(PDImageAttrs));

if(NULL == (bitsData = (ASUns8 *)ASmalloc(iattrs.dataLen)) )
{
    AAlertNote("Serious Memory Allocation Error");
    return false;
}
PDInlineImageGetData(obj,bitsData iattrs.dataLen);
```

Product

reader, base, pro, pdf



Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDLLinkAnnot

PDLLinkAnnotGetAction

```
PDAction PDLLinkAnnotGetAction (PDLLinkAnnot aLinkAnnot);
```

Description

Gets a link annotation's action. After you obtain the action, you can execute it with [AVDocPerformAction](#).

Parameters

aLinkAnnot	The link annotation whose action is obtained.

Return Value

The link annotation's action.

Header File

PDCalls.h

Related Methods

[AVDocPerformAction](#)
[PDLLinkAnnotSetAction](#)

Example

```
PDLLinkAnnot pdAnnot;
PDAction linkAction;
PDViewDestination viewDest;
ASInt32 page;
ASAtom fit;
ASFixedRect destRect;
ASFixed zoom;

linkAction = PDLLinkAnnotGetAction(pdAnnot);
viewDest = PDActionGetDest(linkAction);
PDViewDestGetAttr(viewDest, &page, &fit, &destRect, &zoom);
AVDocPerformAction(AVAppGetActiveDoc(), linkAction);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDLinkAnnotGetBorder

```
void PDLinkAnnotGetBorder (PDLinkAnnot aLinkAnnot,
PDLinkAnnotBorder* border);
```

Description

Gets the border of a link annotation.

Parameters

aLinkAnnot	The link annotation whose border is obtained.
border	<i>(Filled by the method)</i> Pointer to a structure containing the link border. Link corner radii are ignored by the Acrobat viewers.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDLinkAnnotSetBorder](#)

Example

```
pdAnnot = PDPageAddNewAnnot (conPage,
    PDPageGetNumAnnots (conPage) -1,
    ASAtomFromString ("Link"), &dstRect);
linkBorder.hCornerRadius = 16;
linkBorder.vCornerRadius = 16;
linkBorder.dashArrayLen = 0;
linkBorder.width = 1;
PDLinkAnnotSetBorder (pdAnnot, &linkBorder);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDLINKAnnotRemoveAction

```
void PDLINKAnnotRemoveAction (PDLINKAnnot aLinkAnnot);
```

Description

Removes a link annotation's action.

Parameters

aLinkAnnot	The link annotation whose action is removed.

Return Value

None

Known Exceptions

[pdErrBadAction](#)

Notifications

[PDAnnotWillChange](#)

[PDAnnotDidChange](#)

Header File

PDCalls.h

Related Methods

[PDLINKAnnotGetAction](#)

[PDLINKAnnotSetAction](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDLinkAnnotSetAction

```
void PDLinkAnnotSetAction (PDLinkAnnot aLinkAnnot,
    PDAction action);
```

Description

Sets a link annotation’s action.

Parameters

aLinkAnnot	The link annotation whose action is set.
action	New action for the link annotation.

Return Value

None

Notifications

[PDAnnotWillChange](#)
[PDAnnotDidChange](#)

Header File

PDCalls.h

Related Methods

[PDLinkAnnotGetAction](#)

Example

```
PDLinkAnnot pdAnnot;
PDAction oldAction;
PDViewDestination viewDest;
ASInt32 page;
ASAtom fit;
ASFixedRect destRect;
ASFixed zoom;
PDPage tmp;

oldAction = PDLinkAnnotGetAction(pdAnnot);
viewDest = PDActionGetDest(oldAction);
PDViewDestGetAttr(viewDest, &page, &fit,&destRect, &zoom);
PDActionDestroy(oldAction);
tmp = PDDocAcquirePage(pdDoc, page);
PDAnnotGetTitle(pdAnnot, title,sizeof(title));
```

Product

reader, base, pro, pdf



Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDLinkAnnotSetBorder

```
void PDLinkAnnotSetBorder (PDLinkAnnot aLinkAnnot,
    const PDLinkAnnotBorder* border);
```

Description

Sets a link annotation's border.

Parameters

aLinkAnnot	The link annotation whose border is set.
border	Pointer to a structure containing the link border. Link corner radii are ignored by the Acrobat viewers.

Return Value

None

Known Exceptions

- PDAnnotDidChange
- PDAnnotWillChange

Notifications

- PDAnnotWillChange
- PDAnnotDidChange

Header File

PDCalls.h

Related Methods

PDLinkAnnotGetBorder

Example

```
pdAnnot = PDPageAddNewAnnot (conPage,
    PDPageGetNumAnnots (conPage) - 1,
    ASAtomFromString ("Link"), &dstRect);
linkBorder.hCornerRadius = 16;
linkBorder.vCornerRadius = 16;
linkBorder.dashArrayLen = 0;
linkBorder.width = 1;
PDLinkAnnotSetBorder (pdAnnot, &linkBorder);
```

Product

reader, base, pro, pdf



Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDNameTree

PDNameTreeEnum

```
void PDNameTreeEnum (PDNameTree theTree, CosObjEnumProc proc,  
void* clientData);
```

Description

Enumerates the entries in the tree.

Parameters

theTree	A name tree.
proc	A procedure to call once for each name/object pair in theTree . The obj/value pair in proc correspond to the Cos string and CosObj values of each leaf in the tree.
clientData	Data used by the enumeration procedure. clientData is passed to the enumeration procedure proc each time an entry is encountered.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDNameTreeGet](#)
[PDNameTreePut](#)
[PDNameTreeRemove](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDNameTreeEqual

```
ASBool PDNameTreeEqual (PDNameTree tree1, PDNameTree tree2);
```

Description

Compares two name trees to determine if they are the same object.

Parameters

tree1	A name tree.
tree2	Another name tree.

Return Value

Returns **true** if the two name trees are equivalent, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDNameTreeIsValid](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDNameTreeFromCosObj

```
PDNameTree PDNameTreeFromCosObj (CosObj obj);
```

Description

Creates a type cast of the `cosObj` to a name tree. This does not copy the object.

Parameters

<code>obj</code>	The <code>CosObj</code> for which a <code>PDNameTree</code> representation is desired.

Return Value

A `PDNameTree` representation of `obj`.

Header File

`PDCalls.h`

Related Methods

[PDNameTreeNew](#)

[PDNameTreeGetCosObj](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.

PDNameTreeGet

```
ASBool PDNameTreeGet (PDNameTree theTree, const char* name,
ASInt32 nameLen, CosObj* value);
```

Description

Retrieves an object from the name tree.

Parameters

theTree	The PDNameTree from which an object is retrieved.
name	The name of the object within theTree to get. This is a Cos-style string, not a C string.
nameLen	The length of name .
value	(Filled by the method) The Cos object corresponding to name within theTree .

Return Value

Returns true if the object was retrieved, false if no object with this name exists.

Header File

PDCalls.h

Related Methods

```
PDNameTreePut
PDNameTreeRemove
```

Product

reader, base, pro, pdf

Availability

Available if PI_PDMODEL_VERSION (in PIRequir.h) is set to 0x00040000 or higher.



PDNameTreeGetCosObj

```
CosObj PDNameTreeGetCosObj (PDNameTree theTree);
```

Description

Creates a type cast of the name tree to a **cosObj**. This does not copy the object.

Parameters

theTree	The PDNameTree for which a CosObj representation is desired.

Return Value

A **CosObj** representation of **theTree**.

Header File

PDCalls.h

Related Methods

[PDNameTreeNew](#)

[PDNameTreeFromCosObj](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDNameTreesValid

```
ASBool PDNameTreeIsValid (PDNameTree theTree);
```

Description

Validates whether a **PDNameTree** is a **CosDict** Cos object or not

Parameters

theTree	The PDNameTree whose validity is desired.

Return Value

Returns **true** if the name tree is a **CosDict**, **false** otherwise.

Header File

PDCalls.h

Related Methods

- [PDDocCreateNameTree](#)
- [PDDocGetNameTree](#)
- [PDNameTreeEqual](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDNameTreeLookup

```
CosObj PDNameTreeLookup (CosObj nameTree, char* name,
ASInt32 nameLen) ;
```

Description

Given a name tree (such as the Dests tree in the **Names** dictionary) and a string, find the **CosObj** in the tree that matches the string.

See Section 3.8.5 in the [PDF Reference](#) for more information on name trees.

Parameters

nameTree	The name tree in which to search.
name	The name to search for. The name tree uses Cos-style strings, which may use Unicode encoding, and these may contain bytes with zeroes in them (high bytes of ASCII characters). NOTE: name is not a C-style string. Cos string objects can contain NULL chars. Standard C string-handling functions may not work as expected.
nameLen	Length of name , in bytes.

Return Value

The Cos object associated with the specified name, which is the array element following the name.

Header File

PDCalls.h

Related Methods

[PDNameTreeGet](#)
[PDNameTreePut](#)



Example

```
CosObj result = CosNewNull();
CosObj nameTree, name;
char* pName, *p;
ASInt32 nameLen;

/* Assume nameTree and name have appropriate values */
p = CosStringValue(name, &nameLen);
pName = (char *) ASmalloc(nameLen);
p = CosStringValue(name, &nameLen); /* acquire again in case malloc
caused a flush */
memcpy(pName, p, nameLen);

result = PDNameTreeLookup(nameTree, pName, nameLen);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

PDNameTreeNew

```
PDNameTree PDNameTreeNew (PDDoc pdDoc);
```

Description

Creates a new name tree in the document.

Parameters

pdDoc	The document for which a new name tree is desired.

Return Value

The newly created name tree or a **NULL CosObj** if Acrobat is unable to create a **PDNameTree** for the document specified by **pdDoc**.

[PDNameTreeIsValid](#) should be called to determine if the name tree returned by **PDNameTreeNew** is usable.

Header File

PDCalls.h

Related Methods

[PDNameTreeFromCosObj](#)
[PDNameTreeGetCosObj](#)
[PDNameTreeIsValid](#)
[PDNameTreeRemove](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDNameTreeNotifyNameAdded

```
void PDNameTreeNotifyNameAdded (PDNameTree theTree,  
CosObj key, CosObj value);
```

Description

Sends a PDNameTreeNameAdded notification.

Parameters

theTree	The PDNameTree to which a name had been added.
key	The name of the object within theTree that was added. This is a Cos string, not a C string.
value	(Filled by the method) The Cos object corresponding to the object name that was added to theTree.

Notifications

PDNameTreeNameAdded

Related Methods

PDNameTreeNotifyNameRemoved

Product

base, pro, pdf

Availability

Available if PI_PDMODEL_VERSION (in PIRequir.h) is set to 0x00050000 or higher.



PDNameTreeNotifyNameRemoved

```
ACCB1 void ACCB2 PDNameTreeNotifyNameRemoved
(PDNameTree theTree, CosObj removedName);
```

Description

Sends a [PDNameTreeNameRemoved](#) notification.

Parameters

theTree	The PDNameTree from which the name had been removed.
removedName	The name within theTree that was removed.

Notifications

[PDNameTreeNameRemoved](#)

Related Methods

[PDNameTreeNotifyNameAdded](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDNameTreePut

```
void PDNameTreePut (PDNameTree theTree, CosObj key,
CosObj value);
```

Description

Puts a new entry in the name tree. If an entry with this name is already in the tree, it is replaced.

Parameters

theTree	The name tree for which a new entry is added
key	The name of the object to put in the tree. This is a Cos-style string, not a C string. This allows the use of an existing indirect object for the key rather than forcing the creation of a new object.
value	The Cos object to be associated with key .

Return Value

None

Notifications

[PDNameTreeNameAdded](#)

Header File

PDCalls.h

Related Methods

[PDNameTreeGet](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDNameTreeRemove

```
void PDNameTreeRemove (PDNameTree theTree, const char* key,
ASInt32 keyLen) ;
```

Description

Removes the specified object from the tree. Does nothing if no object with that name exists.

Parameters

theTree	The name tree from which an entry is removed.
key	The name of the entry to remove. This is a Cos-style string, not a C string.
keyLen	Length of key name, in bytes.

Return Value

None

Notifications

[PDNameTreeNameRemoved](#)

Header File

PDCalls.h

Related Methods

[PDNameTreeGet](#)
[PDNameTreePut](#)

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.



PDNumTree

PDNumTreeEnum

```
void PDNumTreeEnum (PDNumTree theTree, CosObjEnumProc proc, void* clientData);
```

Description

Enumerates the entries in the tree.

Parameters

theTree	A number tree.
proc	A procedure to call once for each number destination pair in theTree.
clientData	Data used by the enumeration procedure. clientData is passed to the enumeration procedure proc each time a number tree is encountered.

Return Value

None

Header File

PDCalls.h

Related Methods

- PDNumTreeGet
- PDNumTreePut

Product

reader, base, pro, pdf

Availability

Available if PI_PDMODEL_VERSION (in PIRequir.h) is set to 0x00040000 or higher.

PDNumTreeEqual

```
ASBool PDNumTreeEqual (PDNumTree tree1, PDNumTree tree2);
```

Description

Compares two number trees to determine if they are the same object.

Parameters

tree1	A number tree.
tree2	Another number tree.

Return Value

Returns **true** if the two number trees are equivalent, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDNumTreeIsValid](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDNumTreeFromCosObj

```
PDNumTree PDNumTreeFromCosObj (CosObj obj);
```

Description

Creates a type cast of the `cosObj` to a number tree.

Parameters

<code>obj</code>	The <code>CosObj</code> for which a <code>PDNumTree</code> representation is desired.

Return Value

A `PDNumTree` representation of `obj`.

Header File

`PDCalls.h`

Related Methods

[PDNumTreeNew](#)

[PDNumTreeGetCosObj](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.



PDNumTreeGet

```
ASBool PDNumTreeGet (PDNumTree theTree, ASInt32 key,
CosObj* value);
```

Description

Retrieves an object from the number tree.

Parameters

theTree	The PDNumTree requested.
key	The number of the entry to retrieve.
value	(Filled by the method) The value associated with key.

Return Value

Returns **true** if the object was retrieved, **false** if no object with this number exists.

Header File

PDCalls.h

Related Methods

- PDNumTreePut
- PDNumTreeRemove

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in PIRequir.h) is set to **0x00040000** or higher.

PDNumTreeGetCosObj

```
CosObj PDNumTreeGetCosObj (PDNumTree theTree);
```

Description

Creates a type cast of the number tree to a **cosObj**.

Parameters

theTree	The PDNumTree for which a CosObj representation is desired.

Return Value

A **CosObj** representation of **theTree**.

Header File

PDCalls.h

Related Methods

[PDNumTreeNew](#)

[PDNumTreeFromCosObj](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDNumTreesValid

```
ASBool PDNumTreeIsValid (PDNumTree theTree);
```

Description

Validates whether a **PDNumTree** is a **CosDict** Cos object or not

Parameters

theTree	The PDNumTree whose validity is desired.

Return Value

Returns **true** if the number tree is a **CosDict**, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDNumTreeEqual](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDNumTreeNew

```
PDNumTree PDNumTreeNew (PDDoc pdDoc) ;
```

Description

Creates a new number tree in the document.

Parameters

pdDoc	The document for which a new number tree is desired.

Return Value

The newly created number tree or a **NULL CosObj** if Acrobat is unable to create a **PDNumTree** for the document specified by **pdDoc**.

PDNumTreeIsValid should be called to determine if the number tree returned by **PDNumTreeNew** is usable.

Header File

PDCalls.h

Related Methods

[PDNumTreeFromCosObj](#)
[PDNumTreeGetCosObj](#)
[PDNumTreeIsValid](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDNumTreePut

```
void PDNumTreePut (PDNumTree theTree, ASInt32 key,
CosObj value);
```

Description

Puts a new entry in the number tree. If an entry with this number is already in the tree, it is replaced.

Parameters

theTree	The number tree for which a new entry is added
key	The number of the entry.
value	The value associated with key .

Return Value

None

Notifications

[PDNumTreeNumAdded](#)

Header File

PDCalls.h

Related Methods

[PDNumTreeGet](#)
[PDNumTreeRemove](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDNumTreeRemove

```
void PDNumTreeRemove (PDNumTree theTree, ASInt32 key);
```

Description

Removes the specified object from the tree. Does nothing if no object with that number exists.

Parameters

theTree	The number tree from which an entry is removed.
key	The number of the entry to remove.

Return Value

None

Notifications

PDNumTreeNumRemoved

Header File

PDCalls.h

Related Methods

PDNumTreeGet
PDNumTreePut

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.



PDOCConfig

PDOCConfigCreate

```
PDOCConfig PDOCConfigCreate (PDDoc pdDoc);
```

Description

Creates a new optional-content configuration object.

Parameters

<code>doc</code>	The document in which the configuration is used.

Return Value

The newly created configuration object.

Header File

PDCalls.h

Related Methods

[PDOCConfigDestroy](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCConfigDestroy

```
void PDOCConfigDestroy (PDOCConfig pdOCCfg) ;
```

Description

Removes an optional-content configuration object and destroys the Cos objects associated with it. If you pass this method the document's default configuration object (as returned by [PDDocGetOCConfig](#)), nothing happens.

Parameters

<code>pdOCCfg</code>	The configuration to destroy.

Return Value

None

Header File

`PDCalls.h`

Related Methods

[PDOCConfigCreate](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

PDOCConfigGetAllRadioButtonGroups

```
PDOC** PDOCConfigGetAllRadioButtonGroups
(PDOCConfig pdOCCfg) ;
```

Description

Returns an array of pointers to sets of optional-content groups in the configuration that are configured to be mutually exclusive. A set behaves like a radio button group, where only one member can be ON at one time.

Parameters

pdOCCfg	The configuration.

Return Value

A **NULL**-terminated array of pointers to **NULL**-terminated arrays of optional-content groups (OCGs). The client is responsible for freeing all arrays using [ASfree](#).

Header File

PDCalls.h

Related Methods

[PDOCConfigGetRadioButtonGroupForOCG](#)
[PDOCConfigMakeRadioButtonGroup](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCConfigGetCosObj

```
CosObj PDOCConfigGetCosObj (PDOCConfig pdOCCfg);
```

Description

Gets the Cos object associated with the optional-content configuration.

Parameters

pdOCCfg	The configuration for which a CosObj representation is desired.

Return Value

A **CosObj** representation of **pdOCCfg**.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



PDOCConfigGetCreator

```
ASText PDOCConfigGetCreator (PDOCConfig pdOCCfg);
```

Description

Gets the creator property for an optional-content configuration.

Parameters

pdOCCfg	The configuration for which a creator is desired.

Return Value

An **ASText** object containing the creator string from the **Creator** entry in the configuration's Cos dictionary, or **NULL** if there is no such entry. The client is responsible for freeing the **ASText** using **ASTextDestroy**.

Header File

PDCalls.h

Related Methods

[PDOCConfigSetCreator](#)
[PDOCConfigGetName](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCConfigGetInitState

```
void PDOCConfigGetInitState (PDOCConfig pdOCCfg,  
                             PDOCConfigBaseState *bs, PDOCG** onOCGs, PDOCG** offOCGs);
```

Description

Gets the initial ON-OFF states of optional-content groups in an optional-content configuration.

The client is responsible for allocating storage for the arrays using [ASmalloc](#) and freeing them using [ASfree](#).

Parameters

pdOCCfg	The configuration for which the initial state is desired.
bs	<i>(Filled by the method)</i> An existing PDOCConfigBaseState structure in which to store the initialization information.
onOCGs	<i>(Filled by the method)</i> A NULL -terminated array of OCGs that have an initial state of ON, or NULL if there are no such groups.
offOCGs	<i>(Filled by the method)</i> A NULL -terminated array of OCGs that have an initial state of OFF, or NULL if there are no such groups.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCConfigSetInitState](#)
[PDOCGGetInitialState](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCConfigGetIntent

```
ASAtom* PDOCConfigGetIntent (PDOCConfig pdOCCfg);
```

Description

Gets the **Intent** entry for an optional-content configuration. An intent is an **ASAtom** value broadly describing the intended use, either **View** or **Design**. A group's content is considered to be optional (that is, the group's state is considered in its visibility) if any intent in its list matches an intent of the context. The intent list of the context is usually set from the intent list of the document configuration.

The intent array contains entries (atoms) terminated by **ASAtomNull**.

If the configuration has no **Intent** entry, the default value of **View** is used. In this case, optional content is disabled for contexts initialized with this configuration.

Parameters

pdOCCfg	The configuration for which an intent list is desired.

Return Value

The **ASAtomNull**-terminated intent array. The client is responsible for freeing it using **ASFree**.

Header File

PDCalls.h

Related Methods

[PDOCConfigSetIntent](#)
[PDOCContextGetIntent](#)
[PDOCGetIntent](#)
[PDOCUsedInOCConfig](#)
[PDOCUsedInOCContext](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



PDOCConfigGetName

```
ASText PDOCConfigGetName (PDOCConfig pdOCCfg) ;
```

Description

Gets the name of an optional-content configuration.

Parameters

pdOCCfg	The configuration for which a name is desired.

Return Value

An **ASText** object containing the name string from **Name** entry of the configuration's Cos dictionary, or **NULL** if there is no **Name** entry. The client is responsible for freeing the **ASText** object using **ASTextDestroy**.

Header File

PDCalls.h

Related Methods

[PDOCConfigSetName](#)
[PDOCConfigGetCreator](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCConfigGetOCGOrder

```
ASBool PDOCConfigGetOCGOrder (PDOCConfig pdOCConfig
CosObj *orderArray);
```

Description

Gets the UI-display order of optional-content groups (OCGs) in an optional-content configuration. This is the order in which the group names are displayed in the **Layers** panel of Acrobat 6.0.

Parameters

pdOCConfig	The configuration for which an OCG display order is desired.
orderArray	<i>(Filled by the method)</i> A pointer to the Cos object containing the OCG order array. For more information, see the PDF Reference .

Return Value

Returns **true** if the order belongs directly to this configuration, **false** if it is inherited from the document's default configuration.

Header File

PDCalls.h

Related Methods

[PDOCConfigSetOCGOrder](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCConfigGetPDDoc

```
PDDoc PDOCConfigGetPDDoc (PDOConfig pdOCCfg) ;
```

Description

Gets the document to which the optional-content configuration belongs.

Parameters

pdOCCfg	The configuration for which a document is desired.

Return Value

The document object.

Header File

PDCalls.h

Related Methods

[PDDocGetOCConfig](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCConfigGetRadioButtonGroupForOCG

```
PDOCG* PDOCConfigGetRadioButtonGroupForOCG
(PDOCConfig pdOCCfg, PDOCG ocg);
```

Description

Returns an array of optional-content groups in the configuration that contains the specified group, and is configured to behave like a radio button group, where only one member of the set can be ON at one time.

Parameters

pdOCCfg	The optional-content configuration.
ocg	The optional-content group for which to obtain the radio-button group.

Return Value

A **NULL**-terminated array of **PDOCG** objects, or **NULL** if the specified group does not belong to any radio button group. The client is responsible for freeing the array using **ASfree**.

Header File

PDCalls.h

Related Methods

```
PDOCConfigGetAllRadioButtonGroups
PDOCConfigMakeRadioButtonGroup
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCConfigMakeRadioButtonGroup

```
void PDOCConfigMakeRadioButtonGroup (PDOCConfig pdOCCfg,
PDOC* ocs);
```

Description

Configures a mutually exclusive set of optional-content groups in an optional-content configuration. The set behaves like a radio button group, where only one OCG from the set can be ON at a time. A client must enforce this in the UI-level code, not the PD-level code.

Parameters

pdOCCfg	The optional-content configuration.
ocs	A NULL -terminated array of optional-content groups to be included in the group.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCConfigGetAllRadioButtonGroups](#)
[PDOCConfigGetRadioButtonGroupForOCG](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCConfigSetCreator

```
void PDOCConfigSetCreator (PDOCConfig pdOCCfg,  
    AText creator);
```

Description

Sets the creator property of an optional-content configuration. Stores the specified string as the **Creator** entry in the configuration's Cos dictionary.

Parameters

pdOCCfg	The configuration for which to set a creator.
creator	The new creator string.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCConfigGetCreator](#)
[PDOCConfigSetName](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCConfigSetInitState

```
void PDOCConfigSetInitState (PDOCConfig pdOCCfg,  
                             PDOCConfigBaseState bs, PDOCG* onOCGs, PDOCG* offOCGs);
```

Description

Sets the initial ON-OFF states of optional-content groups to be saved in an optional-content configuration.

Parameters

pdOCCfg	The configuration for which to set the initial state.
bs	An existing PDOCConfigBaseState structure containing the initialization information.
onOCGs	A NULL -terminated array of optional-content groups (OCGs) that have an initial state of ON when that is not the base state, or NULL .
offOCGs	A NULL -terminated array of OCGs that have an initial state of OFF when that is not the base state, or NULL .

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCConfigGetInitState](#)
[PDOCGSetInitialState](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCConfigSetIntent

```
void PDOCConfigSetIntent (PDOConfig pdOCCfg, ASAtom *intent);
```

Description

Sets the **Intent** entry in an optional-content configuration's Cos dictionary. An intent is an **ASAtom** value broadly describing the intended use, either **View** or **Design**.

A group's content is considered to be optional (that is, the group's state is considered in its visibility) if any intent in its list matches an intent of the context. The intent list of the context is usually set from the intent list of the document configuration.

If the configuration has no **Intent** entry, the default value of **View** is used. In this case, optional content is disabled for contexts initialized with this configuration.

Parameters

pdOCCfg	The configuration for which to set an intent.
intent	The new Intent entry value, an array of atoms terminated with ASAtomNull . To remove the Intent entry, pass an array with only one element, ASAtomNULL .

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOConfigGetIntent](#)
[PDOContextSetIntent](#)
[PDOCGSetIntent](#)
[PDOCGUsedInOCConfig](#)
[PDOCGUsedInOCContext](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCConfigSetName

```
void PDOCConfigSetName (PDOCConfig pdOCCfg, AText name);
```

Description

Sets the name of an optional-content configuration. Stores the specified string as the **Name** entry in the configuration's Cos dictionary.

Parameters

pdOCCfg	The configuration for which to set the name.
name	The new name string.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCConfigGetName](#)
[PDOCConfigSetCreator](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCConfigSetOCGOrder

```
void PDOCConfigSetOCGOrder (PDOCConfig pdOCCfg,
                             CosObj orderArray);
```

Description

Sets the UI-display order of optional-content groups (OCGs) in an optional-content configuration. This is the order in which the group names are displayed in the **Layers** panel of Acrobat 6.0.

Parameters

pdOCCfg	The configuration for which a OCG is desired.
orderArray	The Cos object containing the OCG order array. For more information, see the PDF Reference . Pass NULL to remove any existing order entry.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCConfigGetOCGOrder](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



PDOCContext

PDOCContextApplyAutoStateChanges

```
ASBool PDOCContextApplyAutoStateChanges (PDOCContext ctx,
PDOCConfig cfg, ASAtom event)
```

Description

Calls [PDOCContextFindAutoStateChanges](#) to find optional-content groups whose ON-OFF states should be toggled, based on usage application directives contained in the configuration's **AS** array, and applies the changes within the given context.

The **AS** array defines how usage entries are used to automatically manipulate the OCG states. It associates an *event* (**View**, **Print**, or **Export**) with a list of OCGs and a *category*, or list of usage keys identifying OCG usage dictionary entries. See the [PDF Reference](#), and the `OCTextAutoStateSnip` example in the Snippet Runner.

Parameters

ctx	The context for which the visibility state is changed.
cfg	The configuration whose usage directives are used.
event	The event for which an ON-OFF state is automatically changed. Events are View , Export , and Print .

Return Value

true if successful, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDOCContextFindAutoStateChanges](#)
[PDOCContextMakeCopyWithAutoStateChanges](#)
[PDOCContextClearAllUserOverrides](#)
[PDOCGGetUserOverride](#)
[PDOCGSetUserOverride](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

PDOCContextClearAllUserOverrides

```
void PDOCContextClearAllUserOverrides (PDOCContext ctx)
```

Description

Removes usage override marks in all optional-content groups in the given context.

When an optional-content group is marked as having had its state set explicitly in a specified context, automatic state changes caused by the **View** event are prevented. When a group's automatic state change is caused by the **Export** or **Print** event, the user-override setting for the group is ignored.

A configuration's **AS** array defines how usage entires are used to automatically manipulate the OCG states. It associates an *event* (**View**, **Print**, or **Export**) with a list of OCGs and a *category*, or list of usage keys identifying OCG usage dictionary entries. See the [PDF Reference](#), and the OCTextAutoStateSnip example in the Snippet Runner.

Parameters

ctx	The context for which the user override marks are removed.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCContextGetUserOverride](#)
[PDOCContextSetUserOverride](#)
[PDOCContextApplyAutoStateChanges](#)
[PDOCContextFindAutoStateChanges](#)
[PDOCContextMakeCopyWithAutoStateChanges](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCContextContentIsVisible

```
ASBool PDOCContextContentIsVisible (PDOCContext ocContext);
```

Description

Tests whether content is visible in the optional-content context. The method considers the context's current OCMD stack, the group ON-OFF states, the non-OC drawing status, the drawing and enumeration type, and the intent.

Use this method in conjunction with the OCMD stack methods.

Parameters

ocContext	The context for which the visibility state is desired.

Return Value

Returns **true** if the content is visible, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDOCContextPopOCMD](#)
[PDOCContextPushOCMD](#)
[PDOCContextResetOCMDStack](#)
[PDOCContextXObjectIsVisible](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCContextFindAutoStateChanges

PDOCContextFindAutoStateChanges (**PDOCContext** ctx, **PDOCConfig** cfg, **ASAtom** event)

Description

Finds optional-content groups whose ON-OFF states should be toggled in the context, based on usage application directives contained in the configuration's **AS** array.

The **AS** array defines how usage entries are used to automatically manipulate the OCG states. It associates an event (**View**, **Print**, or **Export**) with a list of OCGs and a *category*, or list of usage keys identifying OCG usage dictionary entries. See the [PDF Reference](#), and the `OCTextAutoStateSnip` example in the Snippet Runner.

Parameters

ctx	The context for which the visibility state should be changed.
cfg	The configuration whose usage directives are used.
event	The event for which an ON-OFF state is automatically changed. Events are View , Export , and Print .

Return Value

A **NULL**-terminated array of optional-content group objects. The client is responsible for freeing it using [ASfree](#).

Header File

PDCalls.h

Related Methods

[PDOCContextApplyAutoStateChanges](#)
[PDOCContextMakeCopyWithAutoStateChanges](#)
[PDOCContextClearAllUserOverrides](#)
[PDOCContextGetUserOverride](#)
[PDOCContextSetUserOverride](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDOCContextFree

```
void PDOCContextFree (PDOCContext ocContext);
```

Description

Destroys an optional-content context object and frees the associated memory as needed.

Parameters

ocContext	The context object to free.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCContextInit](#)

[PDOCContextNew](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCContextGetIntent

```
ASAtom* PDOCContextGetIntent ( PDOCContext ocContext );
```

Description

Gets the intent list for an optional-content context. An intent is an **ASAtom** value broadly describing the intended use, either **View** or **Design**.

A group's content is considered to be optional (that is, the group's state is considered in its visibility) if any intent in its list matches an intent of the context. The intent list of the context is usually set from the intent list of the document configuration.

Parameters

ocContext	The context for which an intent is desired.

Return Value

An array containing intent entries (**ASAtoms**) terminated by **ASAtomNull**. The client is responsible for freeing it using **ASfree**.

Header File

PDCalls.h

Related Methods

[PDOCContextSetIntent](#)
[PDOCConfigGetIntent](#)
[PDOCGetIntent](#)
[PDOCUsedInOCConfig](#)
[PDOCUsedInOCContext](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCContextGetNonOCDrawing

```
ASBool PDOCContextGetNonOCDrawing (PDOCContext ocContext);
```

Description

Gets the non-OC drawing status for an optional-content context. Content that is *not* marked as optional content is drawn when **NonOCDrawing** is **true**, and not drawn when **NonOCDrawing** is **false**.

Together, this value and the **PDOCDrawEnumType** value of the context determine how both optional and non-optional content on a page is drawn or enumerated. See [PDOCDrawEnumType](#).

Parameters

ocContext	The context for which the non-OC drawing status is desired.

Return Value

Returns **true** if the context's **NonOCDrawing** is **true**, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDOCContextSetNonOCDrawing](#)
[PDOCContextGetOCDDrawEnumType](#)
[PDOCContextNewWithOCDDisabled](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDOCContextGetOCDrawEnumType

```
PDOCDrawEnumType PDOCContextGetOCDrawEnumType
(PDOCContext ocContext);
```

Description

Gets the drawing and enumeration type for an optional-content context. This type, together with the visibility determined by the OCG and OCMD states, controls whether content that is marked as optional content is drawn or enumerated.

Together, this value and the **NonOCDrawing** value of the context determine how both optional and non-optional content on a page is drawn or enumerated. See [PDOCDrawEnumType](#).

Parameters

ocContext	The context for which the drawing and enumeration type is desired.

Return Value

The drawing and enumeration type value.

Header File

PDCalls.h

Related Methods

[PDOCContextGetNonOCDrawing](#)
[PDOCContextSetOCDrawEnumType](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCContextGetOCGStates

```
void PDOCContextGetOCGStates (PDOCContext ocContext,  
                               PDOCG *pdocgs, ASBool *states);
```

Description

Gets the ON-OFF states for the given optional-content groups (OCGs) in the given optional-content context. Returns the states in the **states** array, which must be large enough to hold as many **ASBool** values as there are OCGs.

Parameters

ocContext	The context for which the OCG states are desired.
pdocgs	A NULL -terminated array of optional-content groups whose states are obtained.
states	<i>(Filled by the method)</i> An array of OCG states corresponding to the array of OCGs, true for ON and false for OFF. The array must be large enough to hold as many states as there are non- NULL OCGs.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCContextSetOCGStates](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



PDOCContextGetPDDoc

```
PDDoc PDOCContextGetPDDoc (PDOCContext ocContext);
```

Description

Gets the document that contains an optional-content context.

Parameters

ocContext	The context for which a document is desired.

Return Value

The document object.

Header File

PDCalls.h

Related Methods

[PDDocGetOCContext](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCContextInit

```
void PDOCContextInit (PDOCContext ocContext,
PDOCContextChangeType policy, PDOCContext otherCtx,
PDOCConfig pdOCCfg);
```

Description

Initializes the ON-OFF states of all optional-content groups (OCGs) within an existing context.

Parameters

ocContext	The context to initialize.
policy	The initialization policy for the context. This value determines whether optional-content groups are initially ON or OFF.
otherCtx	Another context from which to take initial OCG states when the policy is kPDOCInit_FromOtherContext . Ignored for other policies.
pdOCCfg	A configuration from which to take initial OCG states when the policy is kPDOCInit_FromConfig . Ignored for other policies.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCContextNew](#)
[PDOCContextFree](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCContextMakeCopy

```
PDOCContext PDOCContextMakeCopy (PDOCContext ocContext);
```

Description

Creates a new context object to represent an optional-content state of the document, using an existing context as a template. This is the same as calling:

```
PDOCContextNew(kOCCInit_FromOtherContext, ocContext, NULL,
PDOContextGetPDDoc(ocContext));
```

Parameters

ocContext	The context to copy.

Return Value

The new **PDOCContext** object. The client is responsible for freeing the context using **PDOCContextFree**.

Header File

PDCalls.h

Related Methods

[PDDocGetOCContext](#)
[PDOContextInit](#)
[PDOContextNew](#)
[PDOContextFree](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCContextMakeCopyWithAutoStateChanges

[PDOCContext](#) [PDOCContextMakeCopyWithAutoStateChanges](#)
([PDOCContext](#) inCtx, [PDOCConfig](#) cfg, [ASAtom](#) event)

Description

Creates a new context object that represents an optional-content state of the document, using an existing context as a template, but applying an automatic state change for the specified event. An automatic state change toggles all groups' ON-OFF states when the triggering event occurs.

A configuration's **AS** array defines how usage entries are used to automatically manipulate the OCG states. It associates an *event* (**View**, **Print**, or **Export**) with a list of OCGs and a *category*, or list of usage keys identifying OCG usage dictionary entries. See the [PDF Reference](#), and the OTextAutoStateSnip example in the Snippet Runner.

Parameters

inCtx	The context to copy.
cfg	The configuration in which the automatic state change applies.
event	The event for which state will automatically change. Events are View , Export , and Print .

Return Value

The new [PDOCContext](#) object. The client is responsible for freeing the context using [PDOCContextFree](#).

Header File

PDCalls.h

Related Methods

[PDOCContextApplyAutoStateChanges](#)
[PDOCContextFindAutoStateChanges](#)
[PDOCContextClearAllUserOverrides](#)
[PDOCContextGetUserOverride](#)
[PDOCContextSetUserOverride](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in [PIRequir.h](#)) is set to **0x00060000** or higher.

PDOCContextNew

```
PDOCContext PDOCContextNew (PDOCContextChangeType policy,
PDOCContext otherCtx, PDOCConfig pdOCCfg, PDDoc pdDoc);
```

Description

Creates a context object that represents an optional-content state of the document, initializing it in the same way as [PDOCContextInit](#).

Parameters

policy	The initialization policy for the new context. This value determines whether optional-content groups (OCGs) are initially ON or OFF.
otherCtx	Another context from which to take initial OCG states when the policy is <code>kPDOCInit_FromOtherContext</code> . Ignored for other policies.
pdOCCfg	A configuration from which to take initial OCG states when the policy is <code>kPDOCInit_FromConfig</code> . Ignored for other policies.
pdDoc	The document for which to create a context.

Return Value

The new `PDOCContext` object. The client is responsible for freeing the context using [PDOCContextFree](#).

Header File

PDCalls.h

Related Methods

```
PDOCContextInit
PDOCContextFree
```

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

PDOCContextNewWithInitialState

```
PDOCContext PDOCContextNewWithInitialState (PDDoc pdDoc);
```

Description

Creates a context object that represents an optional-content state of the document, using the current state as the initial state for each group (OCG), as determined by the document's optional-content configuration (returned by **PDDocGetOCConfig**(pdDoc)).

Parameters

pdDoc	The document for which to create a context.

Return Value

The new **PDOCContext** object. The client is responsible for freeing the context using **PDOCContextFree**.

Header File

PDCalls.h

Related Methods

[PDDocGetOCConfig](#)
[PDOCContextInit](#)
[PDOCContextNew](#)
[PDOCContextNewWithOCDisabled](#)
[PDOCContextFree](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCContextNewWithOCDisabled

```
PDOCContext PDOCContextNewWithOCDisabled (PDDoc pdDoc);
```

Description

Creates a context object that represents an optional-content state of the document, with the `PDOCDrawEnumType` property set to `kPDOC_NoOC`, so that no content marked as optional content is drawn, regardless of the visibility according to the OCGs and OCMDs.

Content that is not marked as optional content may still be drawn, depending on the `NonOCDrawing` property.

Parameters

<code>pdDoc</code>	The document for which to create a context.

Return Value

The new `PDOCContext` object. The client is responsible for freeing the context using `PDOCContextFree`.

Header File

`PDCalls.h`

Related Methods

- `PDOCContextInit`
- `PDOCContextNew`
- `PDOCContextNewWithInitialState`
- `PDOCContextFree`
- `PDOCContextGetNonOCDrawing`
- `PDOCContextGetOCDrawEnumType`
- `PDOCContextSetNonOCDrawing`
- `PDOCContextSetOCDrawEnumType`

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

PDOCContextPopOCMD

```
void PDOCContextPopOCMD (PDOCContext ocContext);
```

Description

Pops the optional-content membership dictionary (OCMD) stack for an optional-content context. The stack is used to track nesting of optional-content states as contents are enumerated or drawn.

- Call the [PDOCContextPushOCMD](#) method when entering **BDC** for optional content or beginning to process a form or annotation that has a **OC** entry.
- Call this method to pop the stack when encountering **EMC**, or finishing the processing of a form or annotation appearance.

To track nested content that is not for optional content, pass in **NULL** for **pdOCMD** when pushing the OCMD stack for **BMC**, patterns, and charprocs, for **BDC** with no optional content, or for forms or annotations that do not have an **OC** entry. When finished processing any of these objects, you can call [PDOCContextPopOCMD](#) without worrying about whether the content was optional or not.

Parameters

ocContext	The context for which to pop the OCMD stack.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCContextPushOCMD](#)
[PDOCContextResetOCMDStack](#)
[PDOCContextContentIsVisible](#)
[PDOCContextXObjectIsVisible](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOContextPushOCMD

```
void PDOContextPushOCMD (PDOContext ocContext,
    PDOCMD pdOCMD) ;
```

Description

Pushes a new optional-content membership dictionary (OCMD) onto the stack for an optional-content context.

The stack is used to track nesting of optional-content states as contents are enumerated or drawn. Call this method when entering the **BDC** for optional content or beginning to process a form or annotation that has a **OC** entry. Call [PDOContextPopOCMD](#) when encountering an **EMC**, or finishing the processing of a form or annotation appearance.

To make it easier to track nested content that is not for optional content, pass **NULL** for **pdOCMD** when encountering **BMC**, patterns, and charprocs. Also pass **NULL** for a **BDC** with no optional content or for forms or annotations that do not have an **OC** entry. When finished processing any of these objects, you can call [PDOContextPopOCMD](#) without worrying about whether the content was optional or not.

Parameters

ocContext	The context containing the OCMD stack.
pdOCMD	The OCMD to push onto the stack.

Return Value

None

Header File

PDCalls.h

Related Methods

- [PDOContextPopOCMD](#)
- [PDOContextResetOCMDStack](#)
- [PDOContextContentIsVisible](#)
- [PDOContextXObjectIsVisible](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCContextResetOCMDStack

```
void PDOCContextResetOCMDStack (PDOCContext ocContext) ;
```

Description

Clears the OCMD stack for an optional-content context, and resets the current visibility for the context based on the context's non-OC drawing setting (see [PDOCContextSetNonOCDrawing](#)). Call this method at the start of an enumeration or drawing operation that uses a given context.

The OCMD stack contains optional-content membership dictionary objects. The OCMD stack methods and the various methods that test visibility (such as [PDAnnotIsCurrentlyVisible](#)) work together as content is being enumerated or drawn to determine whether particular graphical elements are visible or not. Visibility is based on the context's collection of ON-OFF states for optional-content groups, the context's current settings for **NonOCDrawing** and **PDOCDrawEnumType**, and the state of the OCMD stack.

Any custom drawing or enumerating code that needs to keep track of visibility of content must make a private copy of the **PDOCContext** if that context could be accessed by some other client, in order to avoid conflicting state changes. In particular, you must copy the document's default context (as returned by [PDDocGetOCContext](#)). To enforce this, this reset method does nothing when given a document's default context. Similarly, the push and pop stack operations raise an error for the default context.

If you are using the PD-level draw and enumeration methods, you do not need to copy the context or explicitly call the OCMD stack methods, as the PD-level methods do this internally.

Clients of PDFEdit and other libraries that enumerate contents need to use these three methods when traversing the **PDEContent** structure. When entering a new **PDEContent**, call [PDOCContextPushOCMD](#) (passing an OCMD object or **NULL**). Upon finishing the traversal, call [PDOCContextPopOCMD](#).

Parameters

ocContext	The context for which to reset the OCMD stack.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCContextPopOCMD](#)
[PDOCContextPushOCMD](#)
[PDOCContextContentIsVisible](#)



`PDOContextXObjectIsVisible`

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

PDOCContextSetIntent

```
void PDOCContextSetIntent (PDOCContext ocContext,  
ASAtom *intent);
```

Description

Sets the **Intent** entry in an optional-content context's Cos dictionary. An intent is an **ASAtom** value broadly describing the intended use, either **View** or **Design**.
A group's content is considered to be optional (that is, the group's state is considered in its visibility) if any intent in its list matches an intent of the context. The intent list of the context is usually set from the intent list of the document configuration.
Raises an exception if the context is busy.

Parameters

ocContext	The context for which to set the intent.
intent	The new Intent entry value, an array of atoms terminated with ASAtomNull .

Return Value

None

Header File

PDCalls.h

Related Methods

- PDOCContextGetIntent
- PDOCConfigSetIntent
- PDOCGSetIntent
- PDOCGUsedInOCConfig
- PDOCGUsedInOCContext

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCContextSetNonOCDrawing

```
void PDOCContextSetNonOCDrawing (PDOCContext ocContext,
ASBool drawNonOC) ;
```

Description

Sets the non-OC status for an optional-content context. Content that is *not* marked as optional content is drawn when **NonOCDrawing** is **true**, and not drawn when **NonOCDrawing** is **false**. Together, this value and the **PDOCDrawEnumType** value of the context determine how both optional and non-optional content on a page is drawn or enumerated. See [PDOCDrawEnumType](#).

Parameters

ocContext	The context for which to set the non-OC drawing status.
drawNonOC	The new value for the non-OC drawing status, true or false .

Return Value

None

Header File

PDCalls.h

Related Methods

- [PDOCContextGetNonOCDrawing](#)
- [PDOCContextGetOCDrawEnumType](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOContextSetOCDrawEnumType

```
void PDOContextSetOCDrawEnumType (PDOContext ocContext,
PDODrawEnumType dt);
```

Description

Sets the drawing and enumeration type for an optional-content context. This type, together with the visibility determined by the OCG and OCMD states, controls whether content that is marked as optional content is drawn or enumerated.

Together, this value and the **NonOCDrawing** value of the context determine how both optional and non-optional content on a page is drawn or enumerated. See [PDODrawEnumType](#).

Parameters

ocContext	The context for which the drawing and enumeration type is desired.
dt	The new drawing and enumeration type.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOContextGetOCDrawEnumType](#)
[PDOContextSetNonOCDrawing](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCContextSetOCGStates

```
void PDOCContextSetOCGStates (PDOCContext ocContext,  
                               PDOCG *pdocgs, ASBool *newStates);
```

Description

Sets the ON-OFF states for the given optional-content groups (OCGs) in the given optional-content context. The **newStates** array must be large enough to hold as many **ASBool** values as there are OCGs.

Parameters

ocContext	The context for which the OCG states are set.
pdocgs	A NULL -terminated array of optional-content groups.
newStates	An array of new OCG states corresponding to the array of OCGs, true for ON and false for OFF. The array must contain as many states as there are non- NULL OCGs in the pdocgs array.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCContextGetOCGStates](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCContextXObjectIsVisible

```
ASBool PDOCContextXObjectIsVisible (PDOCContext ocContext,
CosObj obj);
```

Description

Tests whether an XObject form or image contained in **obj** is visible in the optional-content context. The method considers the context's current OCMD stack, optional-content group ON-OFF states, the non-OC drawing status, the drawing and enumeration type, the intent, and the specific OCG.

Use this method in conjunction with the OCMD stack methods.

Parameters

ocContext	The context for which to test visibility.
obj	The external object.

Return Value

Returns **true** if the external object is visible, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDOCContextContentIsVisible](#)
[PDOCContextPopOCMD](#)
[PDOCContextPushOCMD](#)
[PDOCContextResetOCMDStack](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDOCG

PDOCGCreate

```
PDOCG PDOCGCreate (PDDoc pdDoc, AText name);
```

Description

Creates a new optional-content group (OCG) object in the document. The order of the groups (as returned by [PDDocGetOCGs](#)) is not guaranteed, and is not the same as the display order (see [PDOCConfigGetOCGOrder](#)).

Parameters

doc	The document in which the group is used.
name	The name of the optional-content group.

Return Value

The newly created group object.

Header File

PDCalls.h

Related Methods

[PDDocGetOCGs](#)
[PDOCGCreateFromCosObj](#)
[PDOCGDestroy](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCGCreateFromCosObj

```
PDOCG PDOCGCreateFromCosObj (CosObj ocgObj);
```

Description

Creates a new optional-content group (OCG) object from a Cos object.

Parameters

ocgObj	The Cos object.

Return Value

The newly created OCG object.

Header File

PDCalls.h

Related Methods

[PDOCGCreate](#)
[PDOCGGetCosObj](#)
[PDOCGDestroy](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDOCGDestroy

```
void PDOCGDestroy (PDOCG pdocg) ;
```

Description

Destroys an optional-content group (OCG) object. This does not delete any content, but deletes the **PDOCG** object, destroys the corresponding Cos object, and invalidates references from OCMDs.

Parameters

pdocg	The optional-content group object.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCGCreate](#)
[PDOCGCreateFromCosObj](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



PDOCGGetCosObj

```
CosObj PDOCGGetCosObj (PDOCG pdocg) ;
```

Description

Gets the Cos object associated with the optional-content group (OCG) object.

Parameters

pdocg	The optional-content group object.

Return Value

The Cos object.

Header File

PDCalls.h

Related Methods

[PDOCGCreateFromCosObj](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDOCGGetCurrentState

```
ASBool PDOCGGetCurrentState (PDOCG pdocg,
                             PDOCContext ocContext);
```

Description

Gets the current ON-OFF state of the optional-content group (OCG) object in a given context.

Parameters

pdocg	The optional-content group object.
ocContext	The context for which to get the group's state.

Return Value

Returns **true** if the state is ON, **false** if it is OFF.

Header File

PDCalls.h

Related Methods

[PDOCGGetInitialState](#)
[PDOCGGetUsageEntry](#)
[PDOCContextGetOCGStates](#)
[PDOCContextSetOCGStates](#)
[PDOCGSetCurrentState](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCGGetFromCosObj

```
PDOCG PDOCGGetFromCosObj (CosObj obj);
```

Description

Gets an optional-content group (OCG) object from the associated Cos object. If you call this multiple times for the same **PDOCG**, it returns the same object.

Parameters

obj	The Cos object.

Return Value

The OCG object.

Header File

PDCalls.h

Related Methods

[PDOCGCreate](#)
[PDOCGCreateFromCosObj](#)
[PDOCGGetCosObj](#)
[PDOCGDestroy](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCGGetInitialState

```
ASBool PDOCGGetInitialState (PDOCG pdocg, PDOCConfig pdOCCfg,
ASBool* initState);
```

Description

Gets a initial state (ON or OFF) of the optional-content group (OCG) object in a given configuration. If the configuration has a **BaseState** of **Unchanged**, and the OCG is not listed explicitly in its ON list or OFF list, then the initial state is taken from the OCG's current state in the document's default context, and the method returns **false**.

Parameters

pdocg	The optional-content group object.
pdOCCfg	The configuration for which to get the group's initial state.
initState	<i>(Filled by the method)</i> The initial state, true if the state is ON, false if it is OFF.

Return Value

Returns **true** if the initial state is unambiguously defined in the configuration, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDOCGGetCurrentState](#)
[PDOCGGetUsageEntry](#)
[PDOCGRemoveInitialState](#)
[PDOCGSetInitialState](#)
[PDOCContextGetOCGStates](#)
[PDOCContextSetOCGStates](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCGGetIntent

```
ASAtom* PDOCGGetIntent ( PDOC pdocg ) ;
```

Description

Gets the intent list for an optional-content group. An intent is an **ASAtom** value broadly describing the intended use, either **View** or **Design**.

A group's content is considered to be optional (that is, the group's state is considered in its visibility) if any intent in its list matches an intent of the context. The intent list of the context is usually set from the intent list of the document configuration.

Parameters

pdocg	The optional-content group object for which the intent is desired.

Return Value

An array containing intent entries (**ASAtoms**) terminated by **ASAtomNull**. The client is responsible for freeing it using **ASfree**.

Header File

PDCalls.h

Related Methods

[PDOCGSetIntent](#)
[PDOCContextGetIntent](#)
[PDOCConfigGetIntent](#)
[PDOCGUsedInOCConfig](#)
[PDOCGUsedInOCContext](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCGGetName

```
ASText PDocGGetName ( PDocG pdocg );
```

Description

Gets the name of an optional-content group. The returned **ASText** is a copy of the OCG's name. The client is free to modify it and responsible for destroying it.

Parameters

pdocg	The optional-content group object for which the name is desired.

Return Value

The name string.

Header File

PDCalls.h

Related Methods

[PDocGSetName](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDOCGGetPDDoc

```
PDDoc PDOCGGetPDDoc (PDOCG pdocg) ;
```

Description

Gets the document that contains an optional-content group.

Parameters

pdocg	The optional-content group object for which the document is desired.

Return Value

The document object.

Header File

PDCalls.h

Related Methods

[PDDocGetOCGs](#)
[PDOCMDGetPDDoc](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCGGetUsageEntry

[CosObj](#) PDOCGGetUsageEntry ([PDOCG](#) pdocg, [ASAtom](#) entry);

Description

Gets usage information from an optional-content group (OCG) object. A **Usage** dictionary entry provides more specific intended usage information than an intent entry. Possible key values are:

CreatorInfo
Language
Export
Zoom
Print
View
User
PageElement

Parameters

pdocg	The optional-content group object.
entry	The usage key in the usage dictionary entry.

Return Value

The usage information associated with the given key in the **Usage** dictionary for the group, or a **NULL** Cos object if the operation fails (because the OCG is malformed or has no dictionary, or because the dictionary has no entry corresponding to the given key).

Example

See example in [PDOCGSetUsageDictEntry](#).

Header File

PDCalls.h

Related Methods

[PDOCGHasUsageInfo](#)
[PDOCGSetUsageDictEntry](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCGGetUserOverride

ASBool PDOCGGetUserOverride (**PDOC** ocg, **PDOCContext** ctx)

Description

Tests whether the optional-content group is marked as having had its state set directly by client code in the specified context (as opposed to automatically by the optional-content AutoState mechanism).

When a group is so marked, automatic state changes caused by the **View** event are prevented. When a group's automatic state change is caused by the **Export** or **Print** event, the user-override setting for the group is ignored.

A configuration's **AS** array defines how usage entries are used to automatically manipulate the OCG states. It associates an *event* (**View**, **Print**, or **Export**) with a list of OCGs and a *category*, or list of usage keys identifying OCG usage dictionary entries. See the [PDF Reference](#), and the OCGTextAutoStateSnip example in the Snippet Runner.

Parameters

ocg	The optional-content group object.
ctx	The context for which the group is tested.

Return Value

true if the group is marked as being overridden in the context, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDOCSetUserOverride](#)
[PDOCContextClearAllUserOverrides](#)
[PDOCContextFindAutoStateChanges](#)
[PDOCContextApplyAutoStateChanges](#)
[PDOCContextMakeCopyWithAutoStateChanges](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDOCGHasUsageInfo

```
ASBool PDOCGHasUsageInfo (PDOCG pdocg) ;
```

Description

Tests whether an optional-content group (OCG) object is associated with a **Usage** dictionary.

Parameters

pdocg	The optional-content group object.

Return Value

Returns **true** if the group has a **Usage** dictionary, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDOCGGetUsageEntry](#)
[PDOCGSetUsageDictEntry](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCGRemoveInitialState

```
void PDOCGRemoveInitialState (PDOCG pdocg,
PDOCConfig pdOCCfg);
```

Description

Removes the initial ON-OFF state information for the optional-content group (OCG) object in a given configuration.

Parameters

pdocg	The optional-content group object.
pdOCCfg	The configuration for which to remove the group's initial state.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCGGetCurrentState](#)
[PDOCGGetInitialState](#)
[PDOCGGetUsageEntry](#)
[PDOCContextGetOCGStates](#)
[PDOCContextSetOCGStates](#)
[PDOCGSetCurrentState](#)
[PDOCGSetInitialState](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCGSetCurrentState

```
void PDOCGSetCurrentState (PDOCG pdocg, PDOCContext ocContext,
ASBool newState);
```

Description

Sets the current ON-OFF state of the optional-content group (OCG) object in a given context.

Parameters

pdocg	The optional-content group object.
ocContext	The context for which to set the group's state.
newState	The new state.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCContextGetOCGStates](#)
[PDOCContextSetOCGStates](#)
[PDOCGGetCurrentState](#)
[PDOCGGetInitialState](#)
[PDDocGetOCContext](#)
[PDOCGRemoveInitialState](#)
[PDOCGSetInitialState](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCGSetInitialState

```
void PDocGSetInitialState (PDocG pdocg, PDocConfig pdocCfg,
ASBool onOff);
```

Description

Sets the initial state (ON or OFF) of the optional-content group (OCG) object in a given configuration.

Parameters

pdocg	The optional-content group object.
pdocCfg	The configuration for which to set the group's initial state.
onOff	The new initial state, true if the state is ON, false if it is OFF.

Return Value

None

Header File

PDCalls.h

Related Methods

- PDocContextGetOCGStates
- PDocContextSetOCGStates
- PDocGGetCurrentState
- PDocGGetInitialState
- PDocGRemoveInitialState
- PDocGSetCurrentState

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCGSetIntent

```
void PDocGSetIntent (PDocG pdocg, ASAtom *intent);
```

Description

Sets the **Intent** entry in an optional-content group's Cos dictionary. An intent is an **ASAtom** value broadly describing the intended use, either **View** or **Design**.

A group's content is considered to be optional (that is, the group's state is considered in its visibility) if any intent in its list matches an intent of the context. The intent list of the context is usually set from the intent list of the document configuration.

Parameters

pdocg	The optional-content group object for which the intent is desired.
intent	The new Intent entry value, an array of atoms terminated with ASAtomNull .

Return Value

None

Header File

PDCalls.h

Related Methods

[PDocGGetIntent](#)
[PDocContextSetIntent](#)
[PDocConfigSetIntent](#)
[PDocGUsedInOCConfig](#)
[PDocGUsedInOCContext](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCGSetName

```
void PDOCGSetName (PDOCG pdocg, AText name);
```

Description

Sets the name of an optional-content group.

Parameters

<code>pdocg</code>	The optional-content group object.
<code>name</code>	The new name string.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCGGetName](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

PDOCGSetUsageDictEntry

```
void PDOCGSetUsageDictEntry (PDOCG pdocg, ASAtom usagekey,  
CosObj usageinfo);
```

Description

Sets a **Usage** dictionary entry in an optional-content group (OCG) object. The entry associates usage information with an entry key for retrieval. If a dictionary does not exist, the method creates one.

A **Usage** dictionary entry provides more specific intended usage information than an intent entry. Possible key values are:

```
CreatorInfo  
Language  
Export  
Zoom  
Print  
View  
User  
PageElement
```

The usage value can act as a kind of metadata, describing the sort of things that belong to the group: for example, text in French, fine detail on a map, or a watermark. The usage values can also be used by the AutoState mechanism to make decisions about what groups should be on and what groups should be off. The AutoState mechanism considers the usage information in the OCGs, the **AS** array of the configuration, and external factors; for example, the language the app is running in, the current zoom level on the page, or if the page is being printed.

Parameters

pdocg	The optional-content group object.
usagekey	The usage entry key.
usageinfo	The usage information to associate with the key.

Return Value

None

Example

The following **Usage** dictionary associates information with two keys for detailed content in a map image:

```
<</Type/OCG  
/Name(1000 foot view)  
/Usage<</Zoom<</min 2.0/max 20.0>>  
<</Export<</ExportState/OFF>> >> >>
```


The group's state applies when the zoom factor is between 2 and 20, and when the content is not being exported.

The following code fragment is from the OCTextAutoStateSnip example in the Snippet Runner:

```
// Change the ocg usage dictionary to indicate a specific zoom usage.
// get the cos document
CosDoc cosDoc = PDDocGetCosDoc(pdDoc);

// set up the OCG's usage dictionary to specify max/min zoom factors
CosObj zoomDict = CosNewDict(cosDoc, false, 2);
CosDictPut(zoomDict, ASAtomFromString("max"),
            CosNewFixed(cosDoc, true, FloatToFixed(1.5)));
CosDictPut(zoomDict, ASAtomFromString("min"),
            CosNewFixed(cosDoc, true, FloatToFixed(1)));
PDOCGSetUsageDictEntry(ocg, ASAtomFromString("Zoom"), zoomDict);

// set up the OCG with particular usage
// in the default configuration's autostate array.
CosObj cosDocConfig = PDOCConfigGetCosObj(docConfig);
```

Header File

PDCalls.h

Related Methods

[PDOCGGetUsageEntry](#)
[PDOCGHasUsageInfo](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCGSetUserOverride

```
void PDOCGSetUserOverride (PDOCG ocg, PDOCContext ctx,
ASBool overridden)
```

Description

Marks the optional-content group as having had its state set directly by client code in the specified context (as opposed to automatically by the optional-content AutoState mechanism).

When a group is so marked, automatic state changes caused by the **View** event are prevented. When a group's automatic state change is caused by the **Export** or **Print** event, the user-override setting for the group is ignored.

A configuration's **AS** array defines how usage entries are used to automatically manipulate the OCG states. It associates an *event* (**View**, **Print**, or **Export**) with a list of OCGs and a *category*, or list of usage keys identifying OCG usage dictionary entries. See the [PDF Reference](#), and the `OCTextAutoStateSnip` example in the Snippet Runner.

Parameters

ocg	The optional-content group object.
ctx	The context for which the group is marked.
overridden	true to mark the group as having had its state set manually, false to clear the mark.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDOCGGetUserOverride](#)
[PDOCContextClearAllUserOverrides](#)
[PDOCContextFindAutoStateChanges](#)
[PDOCContextApplyAutoStateChanges](#)
[PDOCContextMakeCopyWithAutoStateChanges](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

PDOCGUsedInOCConfig

```
ASBool PDOCGUsedInOCConfig (PDOCG pdocg, PDOCConfig pdoccfg);
```

Description

Tests whether an optional-content group (OCG) object is used in a context initialized using the given configuration.

A group's content is considered to be optional (that is, the group's state is considered in its visibility) if any intent in its list matches an intent of the context. The intent list of the context is usually set from the intent list of the document configuration.

Parameters

pdocg	The optional-content group object.
pdoccfg	The optional-content configuration.

Return Value

Returns **true** if the group is taken into consideration when determining the visibility of content, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDOCConfigGetIntent](#)
[PDOCContextGetIntent](#)
[PDOCGGetIntent](#)
[PDOCGUsedInOCContext](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCGUsedInOCContext

```
ASBool PDOCGUsedInOCContext (PDOCG pdocg,
                              PDOCContext pdocctx);
```

Description

Tests whether an optional-content group (OCG) object is used in a given context.

A group's content is considered to be optional (that is, the group's state is considered in its visibility) if any intent in its list matches an intent of the context. The intent list of the context is usually set from the intent list of the document configuration.

Parameters

pdocg	The optional-content group object.
pdocctx	The optional-content context.

Return Value

Returns **true** if the group is taken into consideration when determining the visibility of content, **false** otherwise.

Header File

PDCalls.h

Related Methods

```
PDOCConfigGetIntent
PDOCContextGetIntent
PDOCGetIntent
PDOCUsedInOCConfig
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCMD

PDOCMDCreate

```
PDOCMD PDOCMDCreate (PDDoc pdDoc, PDOCG* ocgs,
PDOCMDVisPolicy policy);
```

Description

Creates a new optional-content membership dictionary object in the given document for the given groups and visibility policy.

To add a group to an existing OCMD, get the current OCG list, modify it, then create a new OCMD with the new list of groups.

Parameters

pdDoc	The document in which the dictionary is used.
ocgs	A NULL -terminated array of optional-content groups (OCGs) to be members of the dictionary.
policy	The visibility policy that determines the visibility of content with respect to the ON-OFF state of OCGs listed in the dictionary.

Return Value

The newly created dictionary object, or **NULL** if no groups are supplied.

Header File

PDCalls.h

Related Methods

[PDOCMDFindOrCreate](#)
[PDOCMDGetFromCosObj](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCMDFindOrCreate

```
PDOCMD PDOCMDFindOrCreate (PDDoc pdDoc, PDOCG* ocgs,
PDOCMDVisPolicy policy);
```

Description

Locates an existing optional-content membership dictionary (OCMD) object that references the given groups, and that uses the same visibility policy. If no such dictionary is found, the method creates one.

If only one group is supplied, the policy is **kOCMDVisibility_AnyOn** or **kOCMDVisibility_AllOn**, and no matching dictionary is found, the method creates an OCMD that directly contains the group without the level of indirection normally introduced by an OCMD. If the indirection is needed to add more groups to the OCMD, use [PDOCMDCreate](#).

To add a group to an existing OCMD, get the current OCG list, modify it, then create a new OCMD with the new list of groups.

Parameters

pdDoc	The document in which the dictionary is used.
ocgs	A NULL -terminated array of optional-content groups (OCGs) to be members of the dictionary.
policy	The visibility policy that determines the visibility of content with respect to the ON-OFF state of OCGs listed in the dictionary.

Return Value

The newly created or existing dictionary object, or **NULL** if no groups are supplied.

Header File

PDCalls.h

Related Methods

[PDOCMDCreate](#)
[PDOCMDGetFromCosObj](#)
[PDAnnotGetOCMD](#)
[PDEElementGetOCMD](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCMDGetCosObj

```
CosObj PDOCMDGetCosObj ( PDOCMD pdocmd ) ;
```

Description

Gets the Cos object associated with the optional-content membership dictionary (OCMD) object.

Parameters

<code>pdocmd</code>	The dictionary object.

Return Value

The Cos object.

Header File

`PDCalls.h`

Related Methods

[PDOCMDGetFromCosObj](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.



PDOCMDGetFromCosObj

[PDOCMD](#) PDOCMDGetFromCosObj ([CosObj](#) obj) ;

Description

Gets an optional-content membership dictionary (OCMD) object from the associated Cos object.

Parameters

obj	The Cos object.

Return Value

The dictionary object.

Header File

PDCalls.h

Related Methods

[PDOCMDCreate](#)
[PDOCMDGetCosObj](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.



PDOCMDGetOCGs

```
PDOCG* PDOCMDGetOCGs (PDOCMD pdocmd);
```

Description

Gets the optional-content groups listed in a membership dictionary.

Parameters

pdocmd	The membership dictionary whose OCGs are obtained.

Return Value

A **NULL**-terminated array of the document's optional-content groups. The client is responsible for freeing the array using [ASfree](#).

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDOCMDGetPDDoc

[PDDoc](#) PDOCMDGetPDDoc ([PDOCMD](#) pdocmd) ;

Description

Gets the document that contains an optional-content membership dictionary.

Parameters

pdocmd	The dictionary for which the document is desired.

Return Value

The document object.

Header File

PDCalls.h

Related Methods

[PDDocGetOCGs](#)
[PDOCGGetPDDoc](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDOCMDGetVisPolicy

```
PDOCMDVisPolicy PDOCMDGetVisPolicy (PDOCMD pdocmd);
```

Description

Gets the optional-content membership dictionary's visibility policy, which determines the visibility of content with respect to the ON-OFF state of OCGs listed in the dictionary.

Parameters

pdocmd	The dictionary whose policy is obtained.

Return Value

The visibility policy.

Header File

PDCalls.h

Related Methods

[PDOCMDIsCurrentlyVisible](#)
[PDOCMDsAreCurrentlyVisible](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCMDIsCurrentlyVisible

```
ASBool PDOCMDIsCurrentlyVisible (PDOCMD pdocmd,
PDOCContext ocContext);
```

Description

Based on the optional-content groups listed in the dictionary, the current ON-OFF state of those groups within the specified context, and the dictionary's visibility policy, test whether the content tagged with this dictionary would be visible.

Ignores the context's current **PDOCDrawEnumType** and **NonOCDrawing** settings.

Parameters

pdocmd	The dictionary.
ocContext	The context in which the visibility of content is tested.

Return Value

Returns **true** if content tagged with this dictionary is visible in the given context, **false** if it is hidden.

Header File

PDCalls.h

Related Methods

[PDOCMDGetVisPolicy](#)
[PDOCMDsAreCurrentlyVisible](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDOCMDsAreCurrentlyVisible

```
ASBool PDOCMDsAreCurrentlyVisible (PDOCMD* pdocmds,
PDOCContext ocContext);
```

Description

Tests a set of optional-content membership dictionaries to determine whether contents tagged with any of them is visible in a given optional-content context. The method calls [PDOCMDIsCurrentlyVisible](#) on each of the dictionaries. If content is visible in the given context in any of the dictionaries, this method returns **true**.

Parameters

pdocmds	A NULL -terminated array of dictionaries to test.
ocContext	The context in which visibility is tested.

Return Value

true if content using any of the dictionaries is visible in the given context, **false** if it is hidden for all of the dictionaries.

Header File

PDCalls.h

Related Methods

[PDOCMDGetVisPolicy](#)
[PDOCMDIsCurrentlyVisible](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDOCMDsMakeContentVisible

```
ASBool PDOCMDsMakeContentVisible (PDOCMD* ocmds,
PDOCContext ocContext);
```

Description

Makes content that uses any of a set of optional-content membership dictionaries visible in a given optional-content context. The method manipulates the states of optional-content groups in the dictionaries so that any content controlled by any of the dictionaries will be visible in the given context. There can be more than one combination of states that satisfies the request. The particular combination of states is not guaranteed from one call to the next.

The method returns **false** if it is not possible to make the content visible—for instance, if there are nested dictionaries, where one specifies "show if the group state is ON", and the other specifies "show if the group state is OFF". In such a case, visibility is always off, so no state setting can make the content visible.

This method ignores the context's draw type.

Parameters

ocmds	A NULL -terminated array of dictionaries to act upon.
ocContext	The context in which the contents are made visible.

Return Value

true if successful or if the OCMD list is empty, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDOCMDGetVisPolicy](#)
[PDOCMDIsCurrentlyVisible](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



PDPage

PDPageAcquirePDEContent

```
PDEContent PDPageAcquirePDEContent (PDPage page,
ASExtension self);
```

Description

Creates a **PDEContent** from the **PDPage**'s contents and resources. The **PDEContent** is cached, so that subsequent calls on the same **PDPage** return the same **PDEContent**, even if the request is from another PDFEdit client. The **PDEContent** remains in the cache as long as someone has it acquired—until someone not using the PDFEdit API changes the **PDPage**'s contents, such as the viewer rotating a page 90 degrees.

Do not call **PDERelease** on **PDEContent** you have acquired with **PDPageAcquirePDEContent**; call **PDPageReleasePDEContent** to release it.

Parameters

page	The page whose content object is acquired.
self	Identifies the caller or client. For plug-ins, this should be the gExtensionID extension. For the Adobe PDF Library, if there is only one client of the PDFEdit subsystem, this should be zero. If there are multiple clients, each should specify a nonzero, non-negative value. (A negative value is reserved for the implementation.)

Return Value

A **PDEContent** representing the page's contents.

Header File

PagePDECntCalls.h

Related Methods

```
PDEContentCreateFromCosObj
PDPageReleasePDEContent
PDPageSetPDEContent
```

Product

base, pro, pdf

**Availability**

Available if `PI_PAGE_PDE_CONTENT_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.

PDPageAddAnnot

```
void PDPageAddAnnot (PDPage aPage, ASInt32 addAfter,
PDAnnot annot);
```

Description

Adds an annotation at the specified location in a page’s annotation array.

Parameters

aPage	The page to which the annotation is added.
addAfter	The index into the page’s annotation array where the annotation is added. See Section 8.4 in the PDF Reference for a description of the annotation array. The first annotation in the array has an index of zero. Passing a value of –2 adds the annotation to the end of the array. Passing other negative values produces undefined results.
annot	The annotation to add.

Known Exceptions

Raises [pdErrOpNotPermitted](#) if:

- 1) The annotation is of subtype Text and the document’s permissions do not include [pdPermEditNotes](#) (see [PDPerms](#)).
- or
- 2) The annotation is of any other subtype and the document’s permissions do not include [pdPermEdit](#).

Notifications

[PDPageWillAddAnnot](#)
[PDPageDidAddAnnot](#)

Header File

PDCalls.h

Related Methods

[PDPageCreateAnnot](#)
[PDPageAddNewAnnot](#)
[PDPageRemoveAnnot](#)

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDPPageAddCosContents

```
void PDPPageAddCosContents (PDPPage page, CosObj newContents);
```

Description

Completely *replaces* the contents of the specified page with **newContents**.

Parameters

page	The page whose Cos contents are replaced.
newContents	A stream Cos object or an array Cos object containing the new contents (stream Cos objects) for page .

Return Value

None

Notifications

[PDDocDidChangePages](#)

Header File

PDCalls.h

Related Methods

[PDPPageRemoveCosContents](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDPAGEADDCOSRESOURCE

```
void PDPAGEADDCOSRESOURCE (PDPAGE page, char* resType,
char* resName, COSOBJ resObj);
```

DESCRIPTION

Adds a Cos resource to a page object. See Section 3.7.2 in the [PDF Reference](#) for a description of page resources.

The necessary dictionaries are created automatically if the page does not already have any resources of the type specified by **resType**, or does not have a Resources dictionary. For example, if you specify a font resource, but the page does not already have a font resource dictionary, this method automatically creates one and puts the font you specify into it.

ProcSet resources cannot be added using this method; they must be added using Cos-level methods to:

- 1) Get the page’s Resources dictionary
- 2) Get the **ProcSet** array from the Resources dictionary
- 3) Add an element to the **ProcSet** array.

PARAMETERS

page	The page to which a resource is added.
resType	Resource type. The named resource types in PDF are: ExtGState ColorSpace Pattern Shading XObject Font Properties Although ProcSet is also a valid resource type, it cannot be added by this method.
resName	Resource name. For example, the name of a font might be F1 .
resObj	The Cos object being added as a resource to page .

Return Value

None

Notifications

[PDDocDidChangePages](#)

Header File

PDCalls.h



Related Methods

[PDPageRemoveCosResource](#)

[PDPageGetCosResources](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDPageAddNewAnnot

```
PDAnnot PDPageAddNewAnnot (PDPage apage, ASInt32 addAfter,
ASAtom subType, const ASFixedRect* initialRect);
```

Description

Adds an annotation to the page. To make the annotation visible after adding it, convert the coordinates of **initialRect** to device coordinates using **AVPageViewRectToDevice**, then call **AVPageViewInvalidateRect** using the converted rectangle. This method is equivalent to calling **PDPageCreateAnnot** followed by **PDPageAddAnnot**.

The **PDPageWillAddAnnot** and **PDPageDidAddAnnot** notifications are broadcast *before* the **PDPageAddNewAnnot** method returns. If you want to finish formatting the annotation before these notifications are called, for example by adding additional key-value pairs to the annotation dictionary, you should call **PDPageCreateAnnot** followed by **PDPageAddAnnot** instead of **PDPageAddNewAnnot**.

Parameters

apage	The page to which the annotation is added.
addAfter	Where to add the annotation in the page’s annotation array. See Section 8.4 in the <i>PDF Reference</i> for a description of the annotation array. Passing a value of –2 adds the annotation to the end of the array (this is generally what you should do unless you have a need to place the annotation at a special location in the array). Passing a value of –1 adds the annotation to the beginning of the array. Passing other negative values produces undefined results.
subType	The subtype of the annotation to add.
initialRect	Pointer to a rectangle specifying the annotation’s bounds, specified in user space coordinates.

Return Value

The newly created annotation.

Known Exceptions

- Raises **pdErrOpNotPermitted** if:
- The annotation is of subtype Text and the document’s permissions do not include **pdPermEditNotes** (see **PDPerms**).
- or—

- The annotation is of any other subtype and the document's permissions do not include `pdPermEdit`.

Notifications

[PDPageWillAddAnnot](#)
[PDPageDidAddAnnot](#)

Header File

`PDCalls.h`

Related Methods

[PDPageCreateAnnot](#)
[PDPageAddAnnot](#)
[PDPageRemoveAnnot](#)

Example

```
AnnotCnt = PDPageGetNumAnnots (TmpPDPage);
myannot= PDPageAddNewAnnot (TmpPDPage,
    AnnotCnt-1,
    ASAtomFromString (STR_EXTENSION_NAME),
    &rect);
/* Display the annotation on the page. */
AVPageViewGetAnnotRect (avPageView,
    ClipAnnotation, &avRect );
AVPageViewInvalidateRect (avPageView,
    &avRect);
```

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDPageCreateAnnot

[PDAnnot](#) PDPageCreateAnnot ([PDPage](#) aPage, [ASAtom](#) subtype, const [ASFixedRect](#)* initialLocation);

Description

Creates a new annotation, associated with the specified page's **CosDoc**, but not added to the page. Use [PDPageAddAnnot](#) to add the annotation to the page.

If you want to create an annotation that prints even if the annotation handler is not present, you have to provide an appearance for it. To do this, add an appearance key (**AP**) to the annotation dictionary, in which you place the Forms XObject for the Normal (**N**), Rollover (**R**), and Down (**D**) appearances; only the Normal appearance is required. Also, add a flags field (**F**) to the annotation dictionary and set the appropriate value for the bit field. A value of 4, which displays the annotation if the handler is not present, shows the annotation, and allows printing it is recommended.

Parameters

aPage	The page to whose PDDoc the annotation is added.
subtype	Subtype of annotation to create.
initialLocation	Pointer to a rectangle specifying the annotation's bounds, specified in user space coordinates.

Return Value

The newly created annotation.

Known Exceptions

Raises [pdErrOpNotPermitted](#) if:

- The annotation is of subtype Text and the document's permissions do not include **pdPermEditNotes** (see [PDPerms](#)).
- or—
- The annotation is of any other subtype and the document's permissions do not include **pdPermEdit**.

Notifications

[PDAnnotWasCreated](#)

Header File

PDCalls.h

Related Methods

[PDPageAddAnnot](#)



PDPageAddNewAnnot

Example

```
PDAnnot annot;  
annot = PDPageCreateAnnot(pdPage,  
ASAtomFromString("Text"), &rect);  
PDPageAddAnnot(pdPage, (ASInt32) 0, annot);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDPageDrawContentsPlacedWithParams

```
void PDPageDrawContentsPlacedWithParams (PDPage page,  
PDDrawParams params)
```

Description

Provides control over the rendering of contents on the page, including an optional-content context that determines which contents are visible.

Parameters

page	The page to draw.
params	The parameters with which to draw the page, including the optional-content context to use for content visibility.

Return Value

None

Known Exceptions

[pdPErrUnableToCreateRasterPort](#)

Header File

PDFLcalls.h

Related Methods

[PDPageDrawContentsToWindowEx](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDPageDrawContentsToWindow

```
void PDPageDrawContentsToWindow (PDPage page, void* window,
void* displayContext, ASBool isDPS, ASFixedMatrix* matrix,
ASFixedRect* updateRect, CancelProc cancelProc,
void* cancelProcClientData);
```

Description

Draws the contents of a page into the specified window.

This method just draws a bitmap to the window. If you want a live document, you need to open an [AVDoc](#) for the PDF file.

The page can also be derived from a [PDDoc](#).

In UNIX, this method cannot be used to draw into a window. UNIX developers should instead use [AVDocOpenFromASFileWithParamString](#) to draw PDF files into their own window from a client.

NOTE: This method cannot be reliably used to print to a device.

Parameters

page	The page to draw into window .
window	Pointer to a platform-dependent window object (WindowPtr or CWindowPtr in Mac OS, or HWND in Windows). In Mac OS, to draw into an offscreen GWorld, pass NULL in window and pass the GWorldPtr in displayContext . In Windows, to draw into an offscreen DC, pass NULL for window .
displayContext	A platform-dependent display context structure (GWorldPtr in Mac OS, HDC in Windows). In Mac OS, displayContext is ignored if window is non- NULL . NOTE: displayContext cannot be reliably used as the hDC for a printer device.
isDps	Currently unused. Always set to false .
matrix	Pointer to the matrix to concatenate onto the default page matrix. It is useful for converting from page to window coordinates and for scaling.

updateRect	Pointer to the rectangle to draw, defined in user space coordinates. Any objects outside of updateRect will not be drawn. All objects are drawn if updateRect is NULL .
cancelProc	Procedure called periodically to check for user cancel of the drawing operation. The default cancel proc can be obtained using AVAppGetCancelProc . May be NULL , in which case no cancel proc is used.
cancelProcClientData	Pointer to user-supplied data to pass to cancelProc each time it is called. Should be NULL if cancelProc is NULL .

Return Value

None

Header File

PDCalls.h

Related Methods

[AVAppGetCancelProc](#)
[PDDrawCosObjToWindow](#)
[PDPageDrawContentsToWindowEx](#)

Example

```

/* Example 1 for UNIX */
XWindowRec winrec;
memset (&winrec, 0, sizeof(winrec));

port = AVWindowGetPlatformThing(win);
winrec.display = XtDisplay(port);
winrec.window = XtWindow(port);
winrec.CMap = None;
winrec.colorStrategy = CS_Find;
PDPPageDrawContentsToWindow(page, winrec,
    displayContext, false, &matrix, 0, 0, 0);

/* Example 2 for Windows */
/* Write the page in the active AVDoc to the window */
CancelProc cancelProc;
void* cancelProcData;
HWND window;

AVDoc adoc;
PDDoc pdoc;
PDPage page;
HDC dc;
ASFixedMatrix matrix;
AVPageView pageview;
ASInt32 pagenum;

cancelProc = AVAppGetCancelProc(&cancelProcData);
adoc = AVAppGetActiveDoc();
pdoc = AVDocGetPDDoc(adoc);
pageview = AVDocGetPageView(adoc);
pagenum = AVPageViewGetPageNum(pageview);

page = PDDocAcquirePage(pdoc, pagenum);
AVPageViewGetPageToDevMatrix(pageview, &matrix);
dc = GetDC(window);

PDPPageDrawContentsToWindow(page, NULL, (void *)dc, false, &matrix, NULL,
cancelProc, cancelProcData);
PDPPageRelease(page);

```

Product

reader, base, pro, pdfi

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDPageDrawContentsToWindowEx

```
void PDPageDrawContentsToWindowEx(PDPage page, void* window,
void* displayContext, ASBool isDPS, ASFixedMatrix* matrix,
ASUns32 flags, ASFixedRect* updateRect, CancelProc cancelProc,
void* cancelProcClientData);
```

Description

Provides control over the rendering of annotations on the page to be drawn into **window**. Provides the ability to specify the flags passed in to the **PDPageDrawContentsToWindows** function.

NOTE: Acrobat 5.0 Implementation Note: This function can only be called with a **flags** value of 0. The function is not supported with any other values for **flags**. Adobe may expand that in the future. **flags = 0** means “do not render the annot faces.” If you want to draw to a window with annots, you should call the original **PDPageDrawContentsToWindow**. In general, Adobe recommends that you not use **PDPageDrawContentsToWindowsEx** in the Acrobat 5.0 release unless you have a specific need to prevent the drawing of annotations.

Parameters

page	The page to draw into window .
window	Pointer to a platform-dependent window object (WindowPtr or CWindowPtr in Mac OS, or HWND in Windows). In Mac OS, to draw into an offscreen GWorld , pass NULL in window and pass the GWorldPtr in displayContext . In Windows, to draw into an offscreen DC, pass NULL for window .
displayContext	A platform-dependent display context structure (GWorldPtr in Mac OS, HDC in Windows). In Mac OS, displayContext is ignored if window is non- NULL . NOTE: Note: displayContext cannot be reliably used as the hDC for a printer device.
isDPS	Currently unused. Always set to false .
matrix	Pointer to the matrix to concatenate onto the default page matrix. It is useful for converting from page to window coordinates and for scaling.
flags	See above. The only value you should use is 0.
updateRect	A rectangle represented by the coordinates of its four sides.

cancelProc	Procedure called periodically to check for user cancel of the drawing operation. The default cancel procedure can be obtained using AVAppGetCancelProc . May be NULL in which case no cancel procedure is used.
cancelProcClientData	Pointer to user-supplied data to pass to cancelProc each time it is called. Should be NULL if cancelProc is NULL .

Return Value

None

Known Exceptions

[pdPErrUnableToCreateRasterPort](#)

Header File

PDCalls.h

Related Methods

[AVAppGetCancelProc](#)
[PDDrawCosObjToWindow](#)
[PDPageDrawContentsToWindow](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDPageDrawContentsWithParams

```
void PDPageDrawContentsWithParams (PDPage page,
PDDrawParams params)
```

Description

Provides control over the rendering of contents on the page, including both those parameters you would pass to [PDPageDrawContentsToWindowEx](#), and an optional-content context that determines which contents are visible.

Parameters

page	The page to draw.
params	The parameters with which to draw the page, including the optional-content context to use for content visibility.

Return Value

None

Known Exceptions

[pdPErrUnableToCreateRasterPort](#)

Header File

PDCalls.h

Related Methods

[PDPageDrawContentsToWindow](#)
[PDPageDrawContentsToWindowEx](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDPageEnumContents

```
void PDPageEnumContents (PDPage page,  
    PDGraphicEnumMonitor mon, void* clientData);
```

Description

Enumerates the contents of a page, calling a procedure for each drawing object in the page description.

NOTE: This method is provided only for backwards compatibility. It has not been updated beyond PDF Version 1.1 and may not work correctly for newly created PDF 1.2 or later files. You should use the PDFEdit API to enumerate page contents.

Parameters

page	The page whose contents are enumerated.
mon	Pointer to a structure containing user-supplied callbacks that are called for each drawing operator on a page. Enumeration ends if any procedure returns false .
clientData	Pointer to user-supplied data to pass to mon each time it is called.

Return Value

None

Header File

PDCalls.h

Product

reader, base, pro

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDPageEnumInks

```
void PDPageEnumInks (PDPage page, PDPageEnumInksCallback proc,
void* clientData);
```

Description

Enumerates the inks for a page, calling the supplied procedure for each [PDPageInk](#) structure.

For the **DeviceCMYK_K** process color model, always finds the four inks Cyan, Magenta, Yellow, and Black, which are marked as process inks. The RGB values in the [PDPageInk](#) structure are the RGB equivalents (in system monitor space) of 100% of the ink, which can be used to show color swatches for a given ink.

If the inks are part of a **DeviceN** colorspace which has not been defined in a **Colorants** dictionary or elsewhere in a **Separation** colorspace, the color of the swatch is undefined.

This call finds all colorspace dictionaries that are in a colorspace dictionary within the page, even if they are not used by the page contents.

Parameters

page	The page whose contents are enumerated.
proc	The user-supplied callback procedure to be applied to each ink. Enumeration ends if any procedure returns false .
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDPageMakeSeparations](#)
[AVPageViewGetNumVisibleInks](#)
[AVPageViewGetVisibleInks](#)
[AVPageViewGetPixelInformationAtPoint](#)
[AVPageViewSetInkPreview](#)
[AVPageViewSetVisibleInks](#)

Product

pro, pdf

**Availability**

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDPageEnumOCGs

```
void PDPageEnumOCGs (PDPage pdPage, PDOCGEnumProc enumProc,  
void* clientData);
```

Description

Enumerates the optional-content groups for the page, calling the supplied procedure for each one. Enumeration continues until all groups have been enumerated, or until **enumProc** returns **false**. Each group is reported once, even if it is referenced multiple times in the page.

Parameters

pdPage	The page whose groups are enumerated.
enumProc	User-supplied callback to call for each group. Enumeration terminates if proc returns false .
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocEnumOCGs](#)

[PDPageGetOCGs](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDPageEnumResources

```
void PDPageEnumResources (PDPage page,
PDResourceEnumMonitor mon, void* clientData);
```

Description

(*Obsolete, provided only for backwards compatibility*) Enumerates the page's resources, calling an enumeration procedure for each resource.

Instead of this method, use [PDDocEnumOCGs](#).

NOTE: This method is provided only for backwards compatibility. It has not been updated beyond PDF Version 1.1 and may not work correctly for newly created PDF 1.2 or later files.

Parameters

page	The page whose Cos resources are enumerated.
mon	Pointer to a structure containing user-supplied callbacks that are called for each of the page's resources. Enumeration ends if any procedure returns false .
clientData	Pointer to user-supplied data to pass to each procedure in mon when it is called.

Return Value

None

Header File

PDCalls.h

Related Methods

- [PDDocEnumOCGs](#)
- [PDPageGetCosResources](#)

Product

reader, base, pro

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDPageFlattenOC

[ASBool](#) PDPageFlattenOC ([PDPage](#) pdPage, [PDOCContext](#) context)

Description

Replaces the page's contents with a version that has no optional content, containing only what was visible on the page when the call was made.

Parameters

pdPage	The page to be modified.
context	The optional-content context in which content is checked for visibility.

Return Value

true if the operation is successful, **false** otherwise.

Header File

PDCalls.h

Related Methods

- [PDDocFlattenOC](#)
- [PDEContentFlattenOC](#)

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDPageGetAnnot

```
PDAnnot PDPageGetAnnot (PDPage aPage, ASInt32 annotIndex);
```

Description

Gets the **annotIndex** annotation on the page.

Parameters

aPage	The page on which the annotation is located.
annotIndex	The index of the annotation to get on aPage . The first annotation on a page has an index of zero.

Return Value

Annotation object.

Header File

PDCalls.h

Related Methods

[PDPageGetNumAnnots](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDPAGEGETANNOTINDEX

ASInt32 PDPAGEGETANNOTINDEX (**PDPage** aPage, **PDAnnot** anAnnot);

Description

Gets the index of a given annotation object on a given page.

Parameters

aPage	The page to which the annotation is attached.
anAnnot	The annotation whose index is obtained.

Return Value

The annotation's index. Returns -1 if the annotation is not on the page.

Header File

PDcalls.h

Related Methods

[PDPAGEGETANNOT](#)

[PDPAGEGETANNOTSEQUENCE](#)

Example

```
ASInt32 ndx = PDPAGEGETANNOTINDEX(pdPage,
    annot);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDPageGetAnnotSequence

ASInt32 PDPageGetAnnotSequence (**PDPage** page, **PDAnnot** annot);

Description

Returns the sequence number of the specified annotation for the given page. It is applicable only to annotations that are listed in Acrobat's Comments pane and therefore are summarizable using **Tools -> Comments -> Summarize** command (not link and widget annotations, for example).

This method is similar to **PDPageGetAnnotIndex** but it checks the information flags from the annotation handler's **PDAnnotHandlerGetAnnotInfoFlagsProc** to determine whether the **PDAnnotOperationSummarize** flag is set, meaning that the annotation has a sequence number.

Parameters

page	The page on which the annotation exists.
annot	The annotation for which the sequence number is desired.

Return Value

The sequence number of the specified annotation; or -1 if the annotation is not in the page or if it is not a summarizable annotation.

NOTE: The sequence number is one-based, while the index returned by **PDPageGetAnnotIndex** is zero-based.

Header File

PDCalls.h

Related Methods

PDPageGetAnnot
PDPageGetAnnotIndex

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDPageGetBBox

```
void PDPageGetBBox (PDPage page, ASFixedRect* bboxP);
```

Description

Gets the bounding box for a page. The bounding box is the rectangle that encloses all text, graphics, and images on the page.

Parameters

page	The page whose bounding box is obtained.
bboxP	<i>(Filled by the method)</i> Pointer to a rectangle specifying the page's bounding box, specified in user space coordinates.

Return Value

None

Header File

PDCalls.h

Related Methods

- [PDPageGetMediaBox](#)
- [PDPageGetCropBox](#)
- [PDPageGetVisibleBBox](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDPageGetBox

```
ASBool PDPageGetBox (PDPage page, ASAtom boxName,
ASFixedRect* box) ;
```

Description

Returns the box specified for the page object intersected with the media box. If the value for **boxName** is **CropBox**, this call is equivalent to [PDPageGetCropBox](#); if the value is **MediaBox**, this call is equivalent to [PDPageGetMediaBox](#).

Parameters

page	The page whose box is obtained.
boxName	An ASAtom representing the type of box. Examples: ArtBox , BleedBox , CropBox , TrimBox , or MediaBox .
box	<i>(Filled by the method)</i> An ASFixedRect specifying the page's box.

Return Value

Returns **true** if the requested box was specified for the page; **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDPageSetBox](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDPAGEGETCOSOBJ

```
CosObj PDPAGEGETCOSOBJ (PDPAGE page) ;
```

Description

Gets the dictionary Cos object associated with a page. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

page	The page whose Cos object is obtained.

Return Value

The dictionary Cos object associated with **page**.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDPageGetCosResources

[CosObj](#) PDPageGetCosResources ([PDPage](#) page) ;

Description

Gets the Cos object corresponding to a page's resource dictionary. A page's resource Cos object may either be directly in the Page Cos object and apply only to the page. Or, it may be in the Pages tree, be shared by multiple pages, and applies to all Page nodes below the point in the Pages tree where it is located.

Parameters

page	The page whose Cos resources are obtained.

Return Value

The dictionary Cos object associated with the page's resource.

Header File

PDCalls.h

Related Methods

[PDPageAddCosResource](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDPageGetCropBox

```
void PDPageGetCropBox (PDPage page, ASFixedRect* cropBoxP);
```

Description

Gets the crop box for a page. The crop box is the region of the page to display and print.

Parameters

page	The page whose crop box is obtained.
cropBoxP	(Filled by the method) Pointer to a rectangle specifying the page's crop box, specified in user space coordinates.

Return Value

None

Header File

PDCalls.h

Related Methods

- [PDPageSetCropBox](#)
- [PDPageGetMediaBox](#)
- [PDPageGetBBox](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDPAGEGETDEFAULTMATRIX

```
void PDPAGEGETDEFAULTMATRIX (PDPAGE pdpage,
    ASFIXEDMATRIX* defaultMatrix);
```

DESCRIPTION

Gets the matrix that transforms user space coordinates to rotated and cropped coordinates. The origin of this space is the bottom-left of the rotated, cropped page. Y is increasing.

PARAMETERS

pdpage	The page whose default transformation matrix is obtained.
defaultMatrix	(Filled by the method) Pointer to the default transformation matrix.

RETURN VALUE

None

HEADER FILE

PDCalls.h

RELATED METHODS

[PDPAGEGETFLIPPEDMATRIX](#)

PRODUCT

reader, base, pro, pdf

AVAILABILITY

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.



PDPageGetDoc

```
PDDoc PDPageGetDoc (PDPage page);
```

Description

Gets the document that contains the specified page.

Parameters

page	The page whose document is obtained.

Return Value

The document that contains the page.

Header File

PDCalls.h

Related Methods

[PDDocAcquirePage](#)

Example

```
PDDoc doc = PDPageGetDoc(pdPage);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDPageGetDuration

ASFixed PDPageGetDuration (PDPage page);

Description

Gets the page's automatic-advance timing value—the maximum amount of time the page is displayed before the viewer automatically advances to the next page.

Parameters

page	The page whose timing value is obtained.

Return Value

The automatic-advance timing for the page, in seconds. If the page does not have an advance timing value, **fxDefaultPageDuration** (representing positive infinity, that is, never advance) is returned.

Header File

PDCalls.h

Related Methods

[PDPageSetDuration](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.



PDPageGetFlippedMatrix

```
void PDPageGetFlippedMatrix (PDPage pdpage,  
    ASFixedMatrix* flipped);
```

Description

Gets the matrix that transforms user space coordinates to rotated and cropped coordinates. The origin of this space is the top-left of the rotated, cropped page. Y is decreasing.

Parameters

pdpage	The page whose flipped transformation matrix is obtained.
flipped	<i>(Filled by the method)</i> Pointer to the flipped transformation matrix.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDPageGetDefaultMatrix](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDPageGetMediaBox

```
void PDPageGetMediaBox (PDPage page, ASFixedRect* mediaBoxP);
```

Description

Gets the media box for a page. The media box is the “natural size” of the page, for example, the dimensions of an A4 sheet of paper.

Parameters

page	The page whose media box is obtained.
mediaBoxP	<i>(Filled by the method)</i> Pointer to a rectangle specifying the page’s media box, specified in user space coordinates.

Return Value

None

Header File

PDCalls.h

Related Methods

- [PDPageSetMediaBox](#)
- [PDPageGetCropBox](#)
- [PDPageGetBBox](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDPageGetNumAnnots

```
ASInt32 PDPageGetNumAnnots (PDPage aPage);
```

Description

Gets the number of annotations on a page. Annotations associated with pop-up windows (such as strikeouts) are counted as two annotations. Widget annotations (form fields) are included in the count.

Parameters

aPage	The page for which the number of annotations is obtained.

Return Value

The number of annotations on **aPage**.

Header File

PDCalls.h

Related Methods

[PDPageGetAnnot](#)

Example

```
AnnotCnt = PDPageGetNumAnnots (TmpPDPage);
myannot= PDPageAddNewAnnot (TmpPDPage,
    AnnotCnt-1,
    ASAtomFromString (STR_EXTENSION_NAME),
    &rect);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in PIRquir.h) is set to **0x00020000** or higher.



PDPageGetNumber

ASInt32 PDPageGetNumber (**PDPage** page) ;

Description

Gets the page number for the specified page.

Parameters

page	The page whose page number is obtained.

Return Value

The page within the document. The first page is 0.

Header File

PDCalls.h

Related Methods

[PDPageNumFromCosObj](#)

Example

```
if (!PDPageGetNumber (pdPage) )
    AVAlertNote ("First page!") ;
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDPAGEGetOCGs

```
PDOCG* PDPAGEGetOCGs (PDPAGE pdPage) ;
```

Description

Gets the optional-content groups for the document.

Parameters

pdPage	The page whose OCGs are obtained.

Return Value

A **NULL**-terminated array of **PDOCG** objects. The client is responsible for freeing the array with **ASfree**.

Header File

PDCalls.h

Related Methods

[PDDocGetOCGs](#)
[PDPAGEEnumOCGs](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDPageGetPalette

```
ASBool PDPageGetPalette (PDPage page, void* displayContext,
char* table);
```

Description

Useful for obtaining the static, platform-specific palette; the bitmap must be already selected into the **displayContext** to get the palette. This API was exposed for the purpose of the ImageConversion plug-in. When that code uses [PDPageDrawContentsToWindow](#) to get a bitmap from AGM, it needs the palette that AGM used in order to get good/correct results.

Parameters

page	The page whose palette is obtained.
displayContext	The bitmap.
table	<i>(Filled by the method)</i> The palette.

Return Value

Returns **true** if the palette was returned; **false** otherwise.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDPAGEGetPDEContentFilters

```
ASBool PDPAGEGetPDEContentFilters (PDPAGE page,
ASInt32* numFilters, ASAtom** filters);
```

Description

Gets filters used by [PDPAGESetPDEContent](#).

The caller is responsible for allocating the filter array **filters** that receives the filters. **filters** can be **NULL** to just obtain the number of filters.

Parameters

page	The page whose content filters are obtained.
numFilters	(Filled by the method) Number of filters used by PDPAGESetPDEContent .
filters	(Filled by the method) Filters used by PDPAGESetPDEContent . If NULL , numFilters contains the number of filters.

Return Value

Returns **true** if filters are obtained, **false** if page's contents are not cached.

Header File

PagePDECntCalls.h

Related Methods

[PDPAGESetPDEContent](#)
[PDPAGESetPDEContentFilters](#)

Product

base, pro, pdf

Availability

Available if **PI_PAGE_PDE_CONTENT_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDPAGEGETPDECONTENTFLAGS

`ASBool PDPAGEGETPDECONTENTFLAGS (PDPAGE page, ASUns32* flags);`

Description

Gets flags used by `PDPAGESETPDECONTENT`.

Parameters

<code>page</code>	The page whose content flags are obtained.
<code>flags</code>	(Filled by the method) <code>PDECONTENTTOCOSOBJFLAGS</code> flags.

Return Value

Returns `true` if flags obtained, `false` if page's contents are not cached.

Header File

`PagePDECntCalls.h`

Related Methods

`PDPAGESETPDECONTENT`
`PDPAGESETPDECONTENTFLAGS`

Product

base, pro, pdf

Availability

Available if `PI_PAGE_PDE_CONTENT_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.



PDPageGetRotate

```
PDRotate PDPageGetRotate (PDPage page) ;
```

Description

Gets the rotation value for a page.

Parameters

page	The page whose rotation is obtained.

Return Value

Rotation value for the given page. Must be one of the [PDRotate](#) values.

Header File

PDCalls.h

Related Methods

[PDPageSetRotate](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDPageGetTransition

```
PDTrans PDPageGetTransition (PDPage page);
```

Description

Gets the transition for a given page.

Parameters

page	The page whose transition is obtained.

Return Value

The page's transition. If the page has no transition, returns a **NULL** transition.

Header File

PDCalls.h

Related Methods

[PDPageSetTransition](#)
[PDTransNull](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

PDPageGetVisibleBBox

```
void PDPageGetVisibleBBox (PDPage page, PDOCContext ocContext,
ASBool includeAnnots, ASFixedRect *fr);
```

Description

Gets the bounding box for a given page for those contents that are visible in the given optional-content context. The bounding box is the rectangle that encloses the visible text, graphics, and images on the page.

Parameters

page	The page whose visible-content bounding box is obtained.
ocContext	The context within which the contents are visible.
includeAnnots	When true , include annotations as content that must be visible to affect the bounding box. When false , annotations are not considered at all.
fr	<i>(Filled by the method)</i> Pointer to a rectangle specifying the page's visible content bounding box, specified in user space coordinates. The client must not pass NULL .

Return Value

None

Header File

PDCalls.h

Related Methods

[PDPageGetBBox](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060002** or higher.



PDPageHasTransition

```
ASBool PDPageHasTransition (PDPage page);
```

Description

Tests whether a page has a transition.

Parameters

page	The page to test.

Return Value

Returns **true** if the page has a transition, **false** otherwise.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

PDPageHasTransparency

```
ASBool PDPageHasTransparency (PDPage pdPage,  
ASBool includeAnnotAppearances);
```

Description

Checks whether a page uses any transparency features.

NOTE: To determine whether the page uses transparency, the resources of the page must be enumerated (though the page contents do not need to be parsed). The page resources may not be optimized for slow (browser-based) connections, so calling **PDPageHasTransparency** before the page has been downloaded may cause unpleasant read behavior and performance problems.

Parameters

pdPage	The page to check.
includeAnnotAppearances	If true , annotation appearances are included in the check; if false annotation appearances will be ignored.

Return Value

Returns **true** if and only if the page uses any transparency features.

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDPageMakeSeparations

```
void PDPageMakeSeparations(PDPage pdPage,
    PDHostSepsSpec spec);
```

Description

Generates print color separations for a page.

This is the entry point for creating separations for a single page. The **spec** structure contains an array of **PDHostSepsPlate** pointers, (typically based on the page inks reported by **PDPageEnumInks**), with settings such as what to do on each plate and the output stream for plates that are being produced. The client owns the memory for the array and all of the records in it, and is responsible for disposing of all allocated memory.

On completion, the marked flags in the **wasColorSet** field of the plates indicate whether each plate was marked—that is, any marking operation happened, even if it was clipped away or knocked out later. The special **All** colorant in a **Separation** color space does not affect the marked flags.

For Adobe Reader and Adobe Acrobat Standard, this method does nothing.

Parameters

pdPage	The page.
spec	The separation specification structure containing parameters for the generation.

Return Value

None

Header File

PDCalls.h

Related Methods

[AVDocPrintSeparations](#)
[PDPageEnumInks](#)
[PDPageMakeSeparations](#)

Product

pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDPageNotifyContentsDidChange

```
void PDPageNotifyContentsDidChange (PDPage page);
```

Description

Broadcasts a [PDPageContentsDidChange](#) notification. If the Acrobat viewer is version 2.1 or later, also broadcasts a [PDPageContentsDidChangeEx](#) notification with `invalidateViews` set to `true`.

You must use this method after using Cos methods to change a page's contents. Do not use this method if you use [PDPageAddCosContents](#) or [PDPageRemoveCosContents](#) to change a page's contents, because those methods automatically generate the appropriate notifications.

Use [PDPageNotifyContentsDidChangeEx](#) instead of this method if you wish to suppress the Acrobat viewer's immediate redraw of the page.

Parameters

<code>page</code>	The page that changed.

Return Value

None

Notifications

[PDPageContentsDidChange](#)
[PDPageContentsDidChangeEx](#) (in version 2.1 and later)

Header File

PDCalls.h

Related Methods

[PDPageAddCosContents](#)
[PDPageRemoveCosContents](#)
[PDPageNotifyContentsDidChangeEx](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDPPageNotifyContentsDidChangeEx

```
void PDPPageNotifyContentsDidChangeEx (PDPPage page,
ASBool invalidateViews);
```

Description

Broadcasts a [PDPPageContentsDidChange](#) notification and a [PDPPageContentsDidChangeEx](#) notification. These notify the Acrobat viewer that a page's contents have been modified, and tells the Acrobat viewer whether to redraw the page immediately.

You must use this method after using Cos methods to change a page's contents. Do not use this method if you use [PDPPageAddCosContents](#) or [PDPPageRemoveCosContents](#) to change a page's contents, because those methods automatically generate the appropriate notifications.

If your plug-in must be compatible with version 2.0 of the Acrobat viewer, you must use [PDPPageNotifyContentsDidChange](#) instead.

Parameters

page	The page that changed.
invalidateViews	true if the Acrobat viewer redraws the page view, false otherwise. This allows plug-ins to make a sequence of modifications to a page's contents, without having the entire page flash after each modification. Passing true for invalidateViews is equivalent to calling PDPPageNotifyContentsDidChange .

Notifications

[PDPPageContentsDidChange](#)
[PDPPageContentsDidChangeEx](#)

Header File

PDCalls.h

Related Methods

[PDPPageAddCosContents](#)
[PDPPageRemoveCosContents](#)
[PDPPageNotifyContentsDidChange](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020001` or higher.



PDPageNumFromCosObj

```
ASInt32 PDPageNumFromCosObj (CosObj pageObj);
```

Description

Gets the page number of the page specified by a Cos object.

Parameters

pageObj	Dictionary Cos object for the page whose number is obtained.

Return Value

The page within the document (the first page in a document is page number zero). Returns -1 if **pageObj** is a **NULL** Cos object.

Header File

PDCalls.h

Related Methods

[PDPageGetNumber](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDPAGEPDECONTENTWASCHANGED

```
void PDPAGEPDECONTENTWASCHANGED (PDPAGE page,
    ASEXTENSION self);
```

Description

Indicates a page's PDECONTENT has changed.

Call this after you alter a PDPAGE's PDECONTENT but do not call PDPAGESETPDECONTENT, so others who have acquired the PDECONTENT know it has changed.

Parameters

page	The page whose content was changed.
self	Identifies the caller or client. For plug-ins, this should be the gExtensionID extension. For the Adobe PDF Library, if there is only one client of the PDFedit subsystem, this should be zero. If there are multiple clients, each should specify a nonzero, non-negative value. (A negative value is reserved for the implementation.)

Return Value

None

Notifications

PagePDECONTENTDIDCHANGE

Header File

PagePDECNTCALLS.h

Related Methods

PDPAGESETPDECONTENT

Product

base, pro, pdf

Availability

Available if PI_PAGE_PDE_CONTENT_VERSION (in PIRequir.h) is set to 0x00040000 or higher.

PDPageRegisterForPDEContentChanged

```
void PDPageRegisterForPDEContentChanged
(PagePDEContentDidChangeNPROTO proc, ASExtension self);
```

Description

Registers for the [PagePDEContentDidChange](#) notification.

Parameters

proc	Callback for function to call when an acquired PDPage 's PDEContent has changed.
self	Identifies the caller or client. For plug-ins, this should be the gExtensionID extension. For the Adobe PDF Library, if there is only one client of the PDFedit subsystem, this should be zero. If there are multiple clients, each should specify a nonzero, non-negative value. (A negative value is reserved for the implementation.)

Return Value

None

Notifications

[PagePDEContentDidChange](#)

Header File

PagePDECntCalls.h

Related Methods

[PDPageRegisterForPDEContentNotCached](#)
[PDPageUnRegisterForPDEContentChanged](#)

Product

base, pro, pdf

Availability

Available if **PI_PAGE_PDE_CONTENT_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDPageRegisterForPDEContentNotCached

```
void PDPageRegisterForPDEContentNotCached
(PagePDEContentNotCachedNPROTO proc, ASExtension self);
```

Description

Register for the [PagePDEContentNotCached](#) notification.

This notification is also sent when others change (or delete) a **PDPage**'s contents without using PDFEdit methods. For instance, rotating or deleting a page in the viewer results in this notification being sent.

PDFEdit registers for almost a half dozen different notifications for the different ways Acrobat can alter page contents—you may need only this notification.

Parameters

proc	Callback for function to call when an acquired PDPage 's PDEContent is no longer valid.
self	Identifies the caller or client. For plug-ins, this should be the gExtensionID extension. For the Adobe PDF Library, if there is only one client of the PDFEdit subsystem, this should be zero. If there are multiple clients, each should specify a nonzero, non-negative value. (A negative value is reserved for the implementation.)

Return Value

None

Notifications

[PagePDEContentNotCached](#)

Header File

PagePDECntCalls.h

Related Methods

[PDPageRegisterForPDEContentChanged](#)
[PDPageUnRegisterForPDEContentNotCached](#)

Product

base, pro, pdf



Availability

Available if **PI_PAGE_PDE_CONTENT_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.



PDPageRelease

```
void PDPageRelease (PDPage page);
```

Description

Decrements the specified page's reference count.

Parameters

page	The page whose reference count is decremented.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDBeadAcquirePage](#)

[PDDocAcquirePage](#)

Example

```
dest = PDDocAcquirePage(pdDoc, ival-1);
newview = PDViewDestCreate(pdDoc, dest,
    k_XYZ, &destRect, fixedZero, ival-1);
newAction = PDActionNewFromDest(pdDoc,
    newview, pdDoc);
PDPageRelease(dest);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDPageReleasePDEContent

```
ASInt32 PDPageReleasePDEContent (PDPage page,  
ASExtension self);
```

Description

Decrements a **PDPage**'s **PDEContent** internal reference count.

The **PDEContent** is *not* automatically deleted when the reference count becomes zero—it remains in the cache until the cache slot is needed for another **PDPage**. Thus, you do not need to keep a **PDEContent** acquired for performance reasons. There is a notification you can register for, that is sent when a **PDEContent** is actually removed from the cache, thus enabling the use of PDFEdit's tagging methods **PDEAddTag**, **PDEGetTag**, **PDERemoveTag** on the **PDEContent** object.

Parameters

page	The page whose content object's use count is decremented.
self	Identifies the caller or client. For plug-ins, this should be the gExtensionID extension. For the Adobe PDF Library, if there is only one client of the PDFEdit subsystem, this should be zero. If there are multiple clients, each should specify a nonzero, non-negative value. (A negative value is reserved for the implementation.)

Return Value

Updated reference count.

Header File

PagePDECntCalls.h

Related Methods

PDPageAcquirePDEContent
PDPageSetPDEContent

Product

base, pro, pdf

Availability

Available if **PI_PAGE_PDE_CONTENT_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDPageRemoveAnnot

```
void PDPageRemoveAnnot (PDPage aPage, ASInt32 annotIndex);
```

Description

Removes an annotation from the specified page. Annotations are stored in Cos arrays, which are automatically compressed when an annotation is removed (see [CosArrayRemove](#)). For this reason, if you use a loop in which you remove annotations, structure the code so the loop processes from the highest to the lowest index. If you loop the other direction, you will skip over annotations immediately following ones you remove.

Parameters

aPage	The page from which the annotation is removed.
annotIndex	The index (into the page's annotation array) of the annotation to remove.

Return Value

None

Known Exceptions

Raises [pdErrOpNotPermitted](#) if:

1) The annotation is of subtype **Text** and the document's permissions do not include [pdPermEditNotes](#) (see [PDPerms](#)).

or

2) The annotation is of any other subtype and the document's permissions do not include [pdPermEdit](#).

Notifications

[PDPageWillRemoveAnnot](#)

[PDPageDidRemoveAnnot](#)

Header File

PDCalls.h

Related Methods

[PDPageGetAnnotIndex](#)

[PDPageAddNewAnnot](#)

[PDPageAddAnnot](#)

**Example**

```
for (i=PDPageGetNumAnnots (pdPage) -1; i>=0;
    i--)
{
    annot = PDPageGetAnnot (pdPage, i);
    if (PDAnnotIsValid (annot) &&
        PDAnnotGetSubtype (annot) ==
        ASAtomFromString ("Link"))
    {
        PDPageRemoveAnnot (annot);
    }
}
```

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDPageRemoveCosContents

```
void PDPageRemoveCosContents (PDPage page);
```

Description

Removes the contents of the specified page.

Parameters

page	The page whose Cos contents are removed.

Return Value

None

Notifications

[PDDocDidChangePages](#)

Header File

PDCalls.h

Related Methods

[PDPageAddCosContents](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDPageRemoveCosResource

```
void PDPageRemoveCosResource (PDPage page, char* resType,
char* resName);
```

Description

Removes a Cos resource from a page object. See Section 3.7.2 in the [PDF Reference](#) for a description of page resources.

Parameters

page	The page whose Cos resources are removed.
resType	Resource type. Resource type. The named resource types in PDF are: ExtGState ColorSpace Pattern Shading XObject Font ProcSet Properties
resName	Resource name. For example, the name of a font might be F1 .

Return Value

None

Notifications

[PDDocDidChangePages](#)

Header File

PDCalls.h

Related Methods

[PDPageAddCosResource](#)
[PDPageGetCosResources](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDPageResumePDEContentChanged

```
void PDPageResumePDEContentChanged (PDPage page);
```

Description

Resumes destruction of [PDEContent](#) objects when a [PDPageContentsDidChange](#) notification occurs. Only use this API if you called [PDPageSuspendPDEContentChanged](#).

Parameters

page	The page whose content is changed.

Return Value

None

Header File

`PgCntProcs.h`

Related Methods

[PDPageSuspendPDEContentChanged](#)

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

PDPageSetBox

```
void PDPageSetBox (PDPage page, ASAtom boxName, ASFixedRect
box);
```

Description

Sets the box specified by **boxName** for the page.

Parameters

page	The page for which the box is to be set.
boxName	An ASAtom representing the type of box. Box names are: ArtBox , BleedBox , CropBox , TrimBox , or MediaBox .
box	An ASFixedRect specifying the coordinates to set for the box.

Return Value

None

Known Exceptions

Numerous

Notifications

[PDDocWillChangePages](#)
[PDDocDidChangePages](#)

Header File

PDCalls.h

Related Methods

[PDPageGetBox](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDPageSetCropBox

```
void PDPageSetCropBox (PDPage page, ASFixedRect cropBox);
```

Description

Sets the crop box for a page. The crop box is the region of the page to display and print. This method ignores the request if either the width or height of cropBox is less than 72 points (one inch).

Parameters

page	The page whose crop box is set.
cropBox	Rectangle specifying the page's crop box, specified in user space coordinates.

Return Value

None

Notifications

- [PDDocWillChangePages](#)
- [PDDocDidChangePages](#)

Header File

PDCalls.h

Related Methods

- [PDPageGetCropBox](#)
- [PDPageSetMediaBox](#)

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.



PDPageSetDuration

```
void PDPageSetDuration (PDPage page, ASFixed fxDuration);
```

Description

Sets the page’s automatic-advance timing value—the maximum amount of time the page is displayed before the viewer automatically advances to the next page.

Parameters

page	The page whose timing is set.
fxDuration	The auto-advance timing, in seconds. If no advance timing is desired, fxDuration should be set to fxDefaultPageDuration .

Return Value

None

Header File

PDCalls.h

Related Methods

[PDPageGetDuration](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.



PDPageSetMediaBox

```
void PDPageSetMediaBox (PDPage page, ASFixedRect mediaBox);
```

Description

Sets the media box for a page. The media box is the “natural size” of the page, for example, the dimensions of an A4 sheet of paper.

Parameters

page	The page whose media box is set.
mediaBox	Rectangle specifying the page’s media box, specified in user space coordinates.

Return Value

None

Notifications

- [PDDocWillChangePages](#)
- [PDDocDidChangePages](#)

Header File

PDCalls.h

Related Methods

- [PDPageGetMediaBox](#)
- [PDPageSetCropBox](#)
- [PDPageGetBBox](#)

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDPageSetPDEContent

```
ASBool PDPageSetPDEContent (PDPage page, AExtension self);
```

Description

Sets the page's **PDEContent** back into the **PDPage**'s Cos object, using the same compression filters with which the content was previously encoded.

Parameters

page	The page whose PDEContent is set.
self	Identifies the caller or client. For plug-ins, this should be the gExtensionID extension. For the Adobe PDF Library, if there is only one client of the PDFEdit subsystem, this should be zero. If there are multiple clients, each should specify a nonzero, non-negative value. (A negative value is reserved for the implementation.)

Return Value

Returns **true** if **PDEContent** successfully set, **false** otherwise.

Notifications

PDPageContentsDidChangeEx

Header File

PagePDECntCalls.h

Related Methods

PDEContentToCosObj
PDPageAcquirePDEContent
PDPageGetPDEContentFlags

Product

base, pro, pdf

Availability

Available if **PI_PAGE_PDE_CONTENT_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDPageSetPDEContentCanRaise

```
void PDPageSetPDEContentCanRaise (PDPage page,
    ASExtension self);
```

Description

Sets the page's [PDEContent](#) back into the [PDPage](#)'s Cos object, using the same compression filters with which the content was previously encoded. This method calls [PDPageNotifyContentsDidChangeEx](#).

This method differs from [PDPageSetPDEContent](#) in that it returns no value, but does raise an exception if it is unable to set the content.

Parameters

page	The page whose PDEContent is set.
self	Identifies the caller or client. For plug-ins, this should be the gExtensionID extension. For the Adobe PDF Library, if there is only one client of the PDFedit subsystem, this should be zero. If there are multiple clients, each should specify a nonzero, non-negative value. (A negative value is reserved for the implementation.)

Return Value

None

Notifications

[PDPageContentsDidChangeEx](#)

Header File

PagePDECntCalls.h

Related Methods

[PDEContentToCosObj](#)
[PDPageAcquirePDEContent](#)
[PDPageGetPDEContentFlags](#)
[PDPageSetPDEContent](#)

Product

base, pro, pdf

Availability

Available if `PI_PAGE_PDE_CONTENT_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

PDPageSetPDEContentFilters

```
ASBool PDPageSetPDEContentFilters (PDPage page,
ASInt32 numFilters, ASAtom* filters);
```

Description

Sets filters used by [PDPageSetPDEContent](#). The filters are not instantiated until [PDPageSetPDEContent](#) is called.

Parameters

page	The page whose content filters are set.
numFilters	Number of filters used by PDPageSetPDEContent .
filters	Array of filters to use by PDPageSetPDEContent .

Return Value

Returns **true** if filters were set, **false** if page’s contents are not cached (and so nothing was done).

Header File

PagePDECntCalls.h

Related Methods

```
PDPageGetPDEContentFilters
PDPageSetPDEContent
```

Product

base, pro, pdf

Availability

Available if **PI_PAGE_PDE_CONTENT_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDPageSetPDEContentFlags

`ASBool PDPageSetPDEContentFlags (PDPage page, ASUns32 flags);`

Description

Sets flags used by `PDPageSetPDEContent`. The flags are not instantiated until `PDPageSetPDEContent` is called.

Parameters

<code>page</code>	The page whose content flags are set.
<code>flags</code>	<code>PDEContentToCosObjFlags</code> flags. The following flags are ignored, since content is always a page: • <code>kPDEContentToForm</code> • <code>kPDEContentToCharProc</code>

Return Value

Returns `true` if flags were set, `false` if page's contents are not cached (and so nothing was done).

Header File

`PagePDECntCalls.h`

Related Methods

`PDPageGetPDEContentFlags`
`PDPageSetPDEContent`

Product

base, pro, pdf

Availability

Available if `PI_PAGE_PDE_CONTENT_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.

PDPageSetRotate

```
void PDPageSetRotate (PDPage page, PDRotate angle);
```

Description

Sets the rotation value for a page.

Parameters

page	The page whose rotation is set.

Return Value

None

Notifications

[PDDocWillChangePages](#)
[PDDocDidChangePages](#)

Header File

PDCalls.h

Related Methods

[PDPageGetRotate](#)

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.



PDPageSetTransition

```
void PDPageSetTransition (PDPage page, PDTrans pdt);
```

Description

Sets the transition for a given page.

Parameters

page	The page whose transition is set.
pdt	The transition for the page.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDPageGetTransition](#)
[PDTransNull](#)

Example

```
/* To remove the transition from a page, pass a NULL transition: */  
  
PDPageSetTransition(page, PDTransNull());
```

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020002` or higher.

PDPageStmGetInlinelImage

```
ASUns32 PDPageStmGetInlineImage (ASStm stm, ASUns32 flags,
CosDoc cosDoc, CosObj resDict, PDPageStmImageDataProc proc,
void* procClientData, ASUns32* imageRawDataStmOffsetP,
ASUns32* imageRawDataLenP, CosObj* imageDict);
```

Description

Reads a PDF page content inline image from a stream. The stream is typically obtained by getting the Cos stream for a page contents or a Form contents, and calling [CosStreamOpenStm](#) to open the stream using the “filtered” mode. This method is called after a **BI** token has been read from the stream. **BI** indicates that the following tokens comprise an inline image dictionary and data.

Begins reading at the current **stm** position. Returns the number of bytes read. This is the number of bytes read from the stream and indicates the amount by which the stream position has advanced.

The image attributes dictionary is returned in **imageDict**. The image data is passed to the [PDPageStmImageDataProc](#); if **proc** is not provided, the image data is discarded.

imageRawDataStmOffsetP and **imageRawDataLenP** may be **NULL**, in which case they are ignored.

The caller should call [CosObjDestroy](#) on **imageDict** when done.

Parameters

stm	The stream from which data is read.
cosDoc	The CosDoc with the PDPage that contains the inline image.
resDict	The Resources dict in which to look up ColorSpace resources for inline images.
flags	Currently unused by this method. (Used by PDPageStmGetToken .)
proc	Callback method to handle inline image data.
procClientData	Client data passed to proc .
imageRawDataStmOffsetP	<i>(Filled by the method)</i> Offset of the data stream, after BI , relative to the beginning of stm .
imageRawDataLenP	<i>(Filled by the method)</i> Offset of the last byte of the data stream between the BI and EI PDF operators.
imageDict	<i>(Filled by the method)</i> The returned image dictionary.



Return Value

Number of bytes read from `stm`.

Known Exceptions

Can raise memory, I/O, and parsing exceptions.

Header File

`PDCalls.h`

Related Methods

[CosStreamOpenStm](#)
[PDPageGetBox](#)
[PDPageStmGetToken](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020002` or higher.

PDPageStmGetToken

```
ASUns32 PDPageStmGetToken (ASStm stm, ASUns32 flags,
PDPageStmStringOverflowProc proc, void* procClientData,
PDPageStmToken pageStmToken) ;
```

Description

Reads a PDF page content token from a stream. The stream is typically obtained by getting the Cos stream for a page contents or a Form contents, and calling [CosStreamOpenStm](#) to open the stream using the “filtered” mode.

It begins reading at the current stream position, and reads exactly one token. It returns the number of bytes read. This is the number of bytes read from the stream and indicates the amount by which the stream position has advanced. The end-of-stream criteria (loop terminating condition) is the following:

- **NULL** object returned (of type **CosNull**)
- Number of bytes read (return value) is **1**

If the token is an integer, real (**ASFixed**) or **ASBool**, then the value is returned in **pageStmToken.ival**. If the token is a string or a name, the value is returned in **pageStmToken.sVal**, and the length of the token is in **pageStmToken.sValLen**. Strings are not null-terminated, but names are null-terminated. If a string length is greater than **kPDPageStmStringMax**, the [PDPageStmStringOverflowProc](#) is called repeatedly with portions of the string. On return from [PDPageStmGetToken](#), the value of **pageStmToken.sValLen** is zero, and **pageStmToken.sVal** is empty (**ival**, **sVal** and **sValLen** are components of the [PDPageStmToken](#)). If there is no overflow proc, then the first **kPDPageStmStringMax** bytes of the string will be returned in **pageStmToken.sVal**, and the remaining bytes are lost. The value of **pageStmToken.sValLen** is **kPDPageStmStringMax** in this case.

If the token is **BI** (begin inline image), [PDPageStmGetInlineImage](#) should be called to parse the inline image.

Parameters

stm	The stm from which data is read.
flags	Bit field of options such as “skip comments” token the returned token value (set the size field before calling). kPDPageStmSkipComments — Skip comments during token generation.
proc	Callback method to handle long strings.
procClientData	Client data passed to the callback method.
pageStmToken	The returned token.

**Return Value**

Number of bytes read from `stm`.

Known Exceptions

Can raise memory, I/O, and parsing exceptions.

Header File

`PDCalls.h`

Related Methods

[CosStreamOpenStm](#)

[PDPageGetCosObj](#)

[PDPageStmGetInlineImage](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020002` or higher.

PDPagesuspendPDEContentChanged

```
void PDPagesuspendPDEContentChanged (PDPages page);
```

Description

Suspends destruction of [PDEContent](#) objects when a [PagePDEContentDidChange](#) notification occurs. Only use this API if you are about to call [PDPagesnotifyContentsDidChange](#) and you do not want PDFEdit to destroy all [PDEContent](#) objects associated with that [PDPages](#). This is used, for example, when [AVAppSetPreference](#) is called. Make sure to call [PDPagesuspendPDEContentChanged](#) on the [PDPages](#) object after you call [PDPagesnotifyContentsDidChange](#).

Parameters

page	The page whose content is changed.

Return Value

None

Header File

[PgCntProcs.h](#)

Related Methods

[PDPagesResumePDEContentChanged](#)

Product

base, pro, pdf

Availability

Available if [PI_PDMODEL_VERSION](#) (in [PIRequir.h](#)) is set to **0x00050000** or higher.

PDPageUnRegisterForPDEContentChanged

```
void PDPageUnRegisterForPDEContentChanged
(PagePDEContentDidChangeNPROTO proc, ASExtension self);
```

Description

Un-registers for the [PagePDEContentDidChange](#) notification.

Parameters

proc	Callback for function to call when an acquired PDPage's PDEContent has changed.
self	Identifies the caller or client. For plug-ins, this should be the gExtensionID extension. For the Adobe PDF Library, if there is only one client of the PDFedit subsystem, this should be zero. If there are multiple clients, each should specify a nonzero, non-negative value. (A negative value is reserved for the implementation.)

Return Value

None

Notifications

[PagePDEContentDidChange](#)

Header File

PagePDECntCalls.h

Related Methods

[PDPageRegisterForPDEContentChanged](#)

Product

base, pro, pdf

Availability

Available if **PI_PAGE_PDE_CONTENT_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDPAGEUnRegisterForPDEContentNotCached

```
void PDPAGEUnRegisterForPDEContentNotCached
(PagePDEContentNotCachedNPROTO proc, ExtensionID clientID);
```

Description

Un-registers for the [PagePDEContentNotCached](#) notification.

Parameters

proc	Callback for function to call when an acquired PDPAGE 's PDEContent is no longer valid.
clientID	Identifies the caller/client. For plug-ins, this should be the gExtensionID extension. For the Adobe PDF Library, if there is only one client of the PDFEdit subsystem, clientID should be zero. If there are multiple clients, each should specify a nonzero, non-negative clientID . (A negative clientID is reserved for the implementation.)

Return Value

None

Notifications

[PagePDEContentNotCached](#)

Header File

PagePDECntCalls.h

Related Methods

[PDPAGERegisterForPDEContentNotCached](#)

Product

base, pro, pdf

Availability

Available if **PI_PAGE_PDE_CONTENT_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDPageLabel

PDPageLabelEqual

```
ASBool PDPageLabelEqual (PDPageLabel pdlOne,  
PDPageLabel pdlTwo) ;
```

Description

Compares two page labels to see if they are equivalent. Two labels are equivalent if they have the same style, starting number (numeric value of the first page associated with the label), and prefix strings which are the same byte-for-byte.

Parameters

<code>pdlOne</code>	A page label.
<code>pdlTwo</code>	Another page label.

Return Value

Returns **true** if the two labels are valid and equivalent, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDPageLabelIsValid](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.

PDPageLabelFromCosObj

```
PDPageLabel PDPageLabelFromCosObj (CosObj cosLabel);
```

Description

Creates a type cast of the **cosObj** to a **PDPageLabel** object.

Parameters

cosLabel	Cos object level representation of a page label.

Return Value

Page label representation of **cosLabel**.

Known Exceptions

Raises **pdErrrBadBaseObj** if **cosLabel** is not a valid page label.

Header File

PDCalls.h

Related Methods

[PDPageLabelGetCosObj](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDPageLabelGetCosObj

```
CosObj PDPageLabelGetCosObj (PDPageLabel pdl);
```

Description

Creates a type cast of the page label object to a Cos object.

Parameters

pd1	A PDPageLabel representation of a page label.

Return Value

A **CosObj** representation of **pd1** if the page label is valid, **NULL CosObj** otherwise.

Header File

PDCalls.h

Related Methods

[PDPageLabelFromCosObj](#)

Product

reader, base, pro, pdf1

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDPageLabelGetPrefix

```
const char* PDPageLabelGetPrefix (PDPageLabel pgLabel,
ASInt32* prefixLen);
```

Description

Returns the prefix string for the label. The prefix string is transitory and should be copied immediately.

Parameters

pgLabel	Label for page whose prefix is desired.
prefixLen	(Filled by the method) The length, in bytes, of the prefix string. Zero if page label is not valid.

Return Value

The prefix string for the label, or **NULL** if none is specified.

Header File

PDCalls.h

Related Methods

- PDPageLabelGetStart
- PDPageLabelGetStyle

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDPageLabelGetStart

```
ASInt32 PDPageLabelGetStart (PDPageLabel pgLabel);
```

Description

Gets the starting number of a given page label.

Parameters

pgLabel	Page label for page whose starting number is desired.

Return Value

The starting number of the page label; that is, the numeric value of the first page associated with the label. Returns 1 if the page label is not valid, or unknown.

Header File

PDCalls.h

Related Methods

[PDPageLabelGetPrefix](#)
[PDPageLabelGetStyle](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDPageLabelGetStyle

```
ASAtom PDPageLabelGetStyle (PDPageLabel pgLabel);
```

Description

Returns an **ASAtom** for the style of the label.

Parameters

pgLabel	Page label whose style is desired.

Return Value

An **ASAtom** for the label style. If no style is specified, returns **ASAtomFromString("None")**.

Known Exceptions

Raises an exception if storage is exhausted or file access fails.

Header File

PDCalls.h

Related Methods

[PDPageLabelGetPrefix](#)
[PDPageLabelGetStart](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDPageLabelIsValid

```
ASBool PDPageLabelIsValid (PDPageLabel pgLabel);
```

Description

Determines whether a page label is valid.

A page label is valid if its values correspond to the specification for page label dictionaries in Table 8.6 in the [PDF Reference](#).

Parameters

pgLabel	Page label whose validity is determined.

Return Value

Returns **true** if the label is valid, **false** otherwise.

Known Exceptions

Raises an exception if storage is exhausted or file access fails.

Header File

PDCalls.h

Related Methods

[PDPageLabelEqual](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDPageLabelNew

```
PDPageLabel PDPageLabelNew (PDDoc pdDoc, ASAtom style,
const char* prefix, ASInt32 prefixLen, ASInt32 startAt);
```

Description

Constructs a new label object in the document with the specified style, prefix, and starting page number.

Parameters

pdDoc	The document that contains the new page label.
style	The numbering system to use for the numeric portion of each label in this range of pages. Possible values are D for decimal numbers, R for upper-case Roman numbers, r for lower-case Roman numbers, A for upper-case alphabetic numbers, or a for lower-case alphabetic numbers. If NULL , the labels for this range will not have a numeric portion.
prefix	A string to prefix to the numeric portion of the page label. May be a NULL string.
prefixLen	Length in bytes of the prefix string.
startAt	The value to use when generating the numeric portion of the first label in this range; must be greater than or equal to 1.

Return Value

The newly created page label.

Known Exceptions

Raises [pdErrrBadBaseObj](#) if the base pages object is missing or invalid.

Header File

PDCalls.h

Related Methods

[PDDocRemovePageLabel](#)
[PDDocGetPageLabel](#)
[PDDocFindPageNumForLabel](#)
[PDDocGetLabelForPageNum](#)
[PDDocSetPageLabel](#)

Product

base, pro, pdf

**Availability**

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.



PDPath

PDPathEnum

```
void PDPathEnum (PDPath obj, PDPathEnumMonitor mon,
void* clientData);
```

Description

Enumerates the specified path’s operators, calling one of several user-supplied callbacks for each operator. The callback that is called depends on which operator is encountered.

Parameters

obj	The path whose operators are enumerated.
mon	Pointer to a structure that contains callbacks. One of the callbacks will be called for each path segment operator in the path. Enumeration ends if any of the monitor’s callbacks returns false .
clientData	Pointer to user-supplied data to pass to the monitor’s callbacks each time one is called.

Return Value

None

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDPathGetPaintOp

```
ASInt32 PDPathGetPaintOp (PDPath obj);
```

Description

Gets flags that indicate which paint/close/clip operators are used for the specified path. For a description of the path painting operators, see Section 4.4.2 in the [PDF Reference](#).

Parameters

<code>obj</code>	The path whose painting operators are obtained.

Return Value

A bit-wise OR of the [PDPathPaintOp](#) flags.

Header File

`PDCalls.h`

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.



PDStyle

PDStyleGetColor

```
void PDStyleGetColor (PDStyle obj, PDColorValue color);
```

Description

Gets a style's color.

Parameters

obj	The style whose color is obtained.
color	<i>(Filled by the method)</i> Pointer to a structure that contains the style's color.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDStyleGetFont](#)

[PDStyleGetFontSize](#)

Example

```
PDColorValue theColor;

PDStyle style = PDWordGetNthCharStyle(pdWF,
    pdWord, 0);
PDStyleGetColor(style, &theColor);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDStyleGetFont

```
PDFont PDStyleGetFont (PDStyle obj);
```

Description

Gets the specified style's font.

Parameters

obj	The style whose font is obtained.

Return Value

The font for the specified style.

Header File

PDCalls.h

Related Methods

[PDStyleGetColor](#)

[PDStyleGetFontSize](#)

Example

```
PDStyle style = PDWordGetNthCharStyle(pdWF,
    pdWord, 0);
font = PDStyleGetFont(style);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDStyleGetFontSize

```
ASFixed PDStyleGetFontSize (PDStyle obj);
```

Description

Get a style's font size.

Parameters

obj	The style whose font size is obtained.

Return Value

The size of the font, in points.

Header File

PDCalls.h

Related Methods

[PDStyleGetColor](#)

[PDStyleGetFont](#)

Example

```
PDStyle style = PDWordGetNthCharStyle(pdWF,
    pdWord, 0);
size = PDStyleGetFontSize(style);
PDFontGetName(font, mbuf, sizeof(mbuf));
if(size == fixedTen && !strstr(mbuf,
    "Italic"))
{
    /* process the 10 pt. Italic */
    ...
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDText

PDTextEnum

```
void PDTextEnum (PDText text, PDStringEnumProc enumProc,  
void* clientData);
```

Description

Enumerates the strings of a text object, calling a procedure for each string. The [PDText](#) object may be obtained from the [PDGraphicEnumTextProc](#) callback of [PDGraphicEnumMonitor](#).

Parameters

text	The text object whose strings are enumerated.
enumProc	User-supplied callback to call for each text string in the text object. Enumeration ends if enumProc returns false .
clientData	Pointer to user-supplied data to pass to enumProc each time it is called.

Return Value

None

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDTextGetState

```
void PDTextGetState (PDText obj, PDTextStateP stateP,
ASInt32 stateLen);
```

Description

Gets the text state for a text object. See Section 5.2 in the [PDF Reference](#) for a discussion of the text state parameters.

Parameters

obj	The text object whose text state is obtained.
stateP	(Filled by the method) Pointer to a PDTextState structure containing the text state information.
stateLen	Must be sizeof (PDTextState).

Return Value

None

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDTextAnnot

PDTextAnnotGetContents

```
ASInt32 PDTextAnnotGetContents (PDTextAnnot aTextAnnot,
char* buffer, ASInt32 bufSize);
```

Description

Gets the text of a text annotation.

NOTE: The contents of text annotations are encoded in **PDFDocEncoding** if created in a Roman environment, Unicode if created in a non-Roman environment.

Parameters

aTextAnnot	The text annotation whose text is obtained.
buffer	<i>(Filled by the method)</i> A buffer into which the text is placed. If the text is encoded using PDFDocEncoding , it can be converted to a platform's native encoding using PDXlateToHost or PDXlateToHostEx .
bufSize	The maximum number of characters that buffer can hold.

Return Value

The number of characters written into **buffer**.

Header File

PDCalls.h

Related Methods

[PDTextAnnotSetContents](#)
[PDXlateToPDFDocEnc](#)
[PDXlateToPDFDocEncEx](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDTextAnnotIsOpen

```
ASBool PDTextAnnotIsOpen (PDTextAnnot aTextAnnot);
```

Description

Tests whether a text annotation is open.

NOTE: In versions later than Acrobat 4.0, this method cannot always correctly determine a text annotation's open state. Beginning with Acrobat 4 (PDF 1.3), text annotations (and other annotations) have an associated Popup annotation which maintains the open state of the popup window; the method works correctly when you pass it the popup annotation itself. You can use Cos-level routines to find the **Popup** entry in the annotation dictionary, which is itself a dictionary containing an **Open** entry.

Parameters

aTextAnnot	The text annotation whose open/closed state is obtained.

Return Value

Returns **true** if the annotation is open, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDTextAnnotSetOpen](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDTextAnnotSetContents

```
void PDTextAnnotSetContents (PDTextAnnot aTextAnnot,  
const char* str, ASInt32 nBytes);
```

Description

Sets the text of a text annotation.

This method also sets the modification date of the annotation to the current date and time.

NOTE: The text must be encoded using **PDFDocEncoding** or Unicode.

Parameters

aTextAnnot	The text annotation whose text is set.
str	A string containing the new text. The string must be encoded using PDFDocEncoding or Unicode. Strings in a platform's native encoding can be converted to PDFDocEncoding using PDXlateToPDFDocEnc or PDXlateToPDFDocEncEx .
nBytes	The number of characters in str .

Return Value

None

Notifications

[PDAnnotWillChange](#)
[PDAnnotDidChange](#)

Header File

PDCalls.h

Related Methods

[PDTextAnnotGetContents](#)
[PDXlateToPDFDocEnc](#)
[PDXlateToPDFDocEncEx](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDTextAnnotSetOpen

```
void PDTextAnnotSetOpen (PDTextAnnot aTextAnnot,
ASBool isOpen);
```

Description

Opens or closes a text annotation.

NOTE: In versions later than Acrobat 4.0, this method cannot always correctly set a text annotation's open state. Beginning with Acrobat 4 (PDF 1.3), text annotations (and other annotations) have an associated Popup annotation which maintains the open state of the popup window; the method works correctly when you pass it the popup annotation itself. You can use Cos-level routines to find the **Popup** entry in the annotation dictionary, which is itself a dictionary containing an **Open** entry.

Parameters

aTextAnnot	The annotation to open or close.
isOpen	true if the annotation is opened, false if the annotation is closed.

Return Value

None

Notifications

[PDAnnotWillChange](#)
[PDAnnotDidChange](#)

Header File

PDCalls.h

Related Methods

[PDTextAnnotIsOpen](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDTextSelect

PDTextSelectCreatePageHilite

```
PDTextSelect PDTextSelectCreatePageHilite (PDPage page,
HiliteEntry* list, ASInt32 listLen);
```

Description

Creates a text selection from a page and a list of highlights specified as character offsets from the start of the page. Character offsets are a well-defined quantity in the PDF file, and are therefore stable against revisions of the word-finding algorithm, which makes them a good way to isolate yourself from changes in the algorithm.

This method does not highlight the text selection. That occurs when you pass the **PDTextSelect** returned by this method to [AVDocSetSelection](#).

NOTE: As in the Acrobat viewer, the text selection is always of whole words, not part of words.

Parameters

page	The page on which the highlights appear.
list	Pointer to an array of highlight entries. If the length field of a HiliteEntry is 0, the entire word is highlighted. list should not contain multiple instances of the same highlight; the display appearance is undefined when it does.
listLen	The number of highlight entries in list .

Return Value

The newly created text selection.

Header File

PDCalls.h

Related Methods

[AVDocSetSelection](#)
[PDTextSelectCreateWordHilite](#)
[PDTextSelectCreateWordHiliteEx](#)
[PDTextSelectCreateRanges](#)
[PDTextSelectCreateRangesEx](#)
[PDTextSelectCreatePageHiliteEx](#)
[PDDocCreateTextSelect](#)
[PDTextSelectDestroy](#)

**Product**

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDTextSelectCreatePageHiliteEx

```
PDTextSelect PDTextSelectCreatePageHiliteEx (PDPage page,
HiliteEntry* list, ASInt32 listLen, ASInt16 WFVersion);
```

Description

Adds **WFVersion** parameter to **PDTextSelectCreatePageHilite**. Same as **PDTextSelectCreatePageHilite**, but it creates WordFinder using the specified version number. It is intended to be used by plug-ins that want to do text highlight with previous versions of the word finder algorithm.

Parameters

page	The page on which the highlights appear.
list	Pointer to an array of highlight entries. If the length field of a HiliteEntry is 0, the entire word is highlighted. list should not contain multiple instances of the same highlight; the display appearance is undefined when it does.
listLen	The number of highlight entries in list .
WFVersion	WordFinder version: WF_LATEST_VERSION —To obtain latest available version. WF_VERSION_2 —Version used for Acrobat 3.x, 4.x. WF_VERSION_3 —Available in Acrobat 5.0 without Accessibility enabled. Includes some improved word piecing algorithms. WF_VERSION_4 —For Acrobat 5.0 with Accessibility enabled. Includes advanced word ordering algorithms in addition to improved word piecing algorithms.

Return Value

The newly created text selection.

Header File

PDCalls.h

Related Methods

```
AVDocSetSelection
PDTextSelectCreateWordHilite
PDTextSelectCreateWordHiliteEx
PDTextSelectCreatePageHilite
PDTextSelectCreateRanges
PDTextSelectCreateRangesEx
PDDocCreateTextSelect
```



PDTextSelectDestroy

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDTextSelectCreateRanges

```
PDTextSelect PDTextSelectCreateRanges (PDPage page,
PDTextSelectRange range, ASInt32 rangeCount);
```

Description

Creates a text selection from one or more ranges.

This method does not highlight the text selection. That occurs when you pass the **PDTextSelect** returned by this method to [AVDocSetSelection](#).

Parameters

page	The page on which the text appears.
range	Pointer to an array of ranges that are used to create a text selection. Each array element is a PDTextSelectRange structure.
rangeCount	The number of ranges in range .

Return Value

A text selection created from the specified ranges.

Header File

PDCalls.h

Related Methods

[AVDocSetSelection](#)
[PDTextSelectCreateRangesEx](#)
[PDTextSelectCreatePageHilite](#)
[PDTextSelectCreatePageHiliteEx](#)
[PDTextSelectCreateWordHilite](#)
[PDTextSelectCreateWordHiliteEx](#)
[PDTextSelectDestroy](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDTextSelectCreateRangesEx

```
PDTextSelect PDTextSelectCreateRangesEx (PDPage page,
PDTextSelectRange range, ASInt32 rangeCount,
ASInt16 WFVersion);
```

Description

Adds the **WFVersion** parameter to **PDTextSelectCreateRanges**. Same as **PDTextSelectCreateRanges** but it creates WordFinder using specified version number. It is intended to be used by plug-ins that want to do text highlight with previous versions of the word finder algorithm.

Parameters

page	The page on which the text appears.
range	Pointer to an array of ranges that are used to create a text selection. Each array element is a PDTextSelectRange structure.
rangeCount	The number of ranges in range .
WFVersion	WordFinder version: WF_LATEST_VERSION —To obtain latest available version. WF_VERSION_2 —Version used for Acrobat 3.x, 4.x. WF_VERSION_3 —Available in Acrobat 5.0 without Accessibility enabled. Includes some improved word piecing algorithms. WF_VERSION_4 —For Acrobat 5.0 with Accessibility enabled. Includes advanced word ordering algorithms in addition to improved word piecing algorithms.

Return Value

A text selection created from the specified ranges.

Header File

PDCalls.h

Related Methods

```
AVDocSetSelection
PDTextSelectCreatePageHilite
PDTextSelectCreatePageHiliteEx
PDTextSelectCreateRanges
PDTextSelectCreateWordHilite
PDTextSelectCreateWordHiliteEx
PDTextSelectDestroy
```



Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDTextSelectCreateWordHilite

```
PDTextSelect PDTextSelectCreateWordHilite (PDPage page,
HiliteEntry* list, ASInt32 listLen);
```

Description

Creates a text selection from a list of highlights specified as word offsets from the start of the page. Word offsets are not well-defined in PDF files, but are calculated by the word-finding algorithm. As a result, word offsets will in general differ in different versions of the word finding algorithm. If you choose to store word offsets, you must also store the version of the word-finding algorithm from which they are obtained using [PDWordFinderGetLatestAlgVersion](#).

This method does not highlight the text selection. That occurs when you pass the [PDTextSelect](#) returned by this method to [AVDocSetSelection](#).

Parameters

page	The page on which the highlights appear.
list	Pointer to an array of highlight entries. list should not contain multiple instances of the same highlight; the display appearance is undefined when it does.
listLen	The number of highlight entries in list .

Return Value

The newly created text selection.

Header File

PDCalls.h

Related Methods

[AVDocSetSelection](#)
[PDTextSelectCreateRanges](#)
[PDTextSelectCreateRangesEx](#)
[PDTextSelectCreatePageHilite](#)
[PDTextSelectCreatePageHiliteEx](#)
[PDTextSelectCreateWordHiliteEx](#)
[PDWordFinderGetLatestAlgVersion](#)
[PDTextSelectDestroy](#)

Product

reader, base, pro, pdf



Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDTextSelectCreateWordHiliteEx

```
PDTextSelect PDTextSelectCreateWordHiliteEx (PDPage page,
HiliteEntry* list, ASInt32 listLen, ASInt16 WFVersion);
```

Description

Adds the **WFVersion** parameter to **PDTextSelectCreateWordHilite**.

Parameters

page	The page on which the highlights appear.
list	Pointer to an array of highlight entries. list should not contain multiple instances of the same highlight; the display appearance is undefined when it does.
listLen	The number of highlight entries in list .
WFVersion	WordFinder version: WF_LATEST_VERSION —To obtain latest available version. WF_VERSION_2 —Version used for Acrobat 3.x, 4.x. WF_VERSION_3 —Available in Acrobat 5.0 without Accessibility enabled. Includes some improved word piecing algorithms. WF_VERSION_4 —For Acrobat 5.0 with Accessibility enabled. Includes advanced word ordering algorithms in addition to improved word piecing algorithms.

Return Value

The newly created text selection.

Header File

PDCalls.h

Related Methods

```
AVDocSetSelection
PDTextSelectCreateRanges
PDTextSelectCreateRangesEx
PDTextSelectCreatePageHilite
PDTextSelectCreatePageHiliteEx
PDTextSelectCreateWordHilite
PDWordFinderGetLatestAlgVersion
PDTextSelectDestroy
```

Product

reader, base, pro, pdf



Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDTextSelectDestroy

```
void PDTextSelectDestroy (PDTextSelect text);
```

Description

Deletes a text selection object (the text on the page remains unchanged). Do not use this method to destroy a text selection that was passed to [AVDocSetSelection](#); such text selections are automatically destroyed when a new selection is made or the selection is cleared.

Parameters

text	The text selection to destroy.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDDocCreateTextSelect](#)
[PDTextSelectCreatePageHilite](#)
[PDTextSelectCreateRanges](#)
[PDTextSelectCreateWordHilite](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDTextSelectEnumQuads

```
void PDTextSelectEnumQuads (PDTextSelect text,
    PDTextSelectEnumQuadProc proc, void* procObj);
```

Description

Enumerates the bounding quads in a text selection. **proc** is called for each quad. If a word is on a curve it may have a quad for each character, but it may also have two characters per quad. An upright word will have only one quad for all the characters. An upright hyphenated word will have two quads.

Parameters

text	The text selection whose bounding quads are enumerated.
proc	User-supplied callback to call for each quad. Enumeration halts if proc returns false .
procObj	User-supplied data to pass to proc each time it is called.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

PDCalls.h

Related Methods

[PDDocCreateTextSelect](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDTextSelectEnumText

```
void PDTextSelectEnumText (PDTextSelect text,
    PDTextSelectEnumTextProc proc, void* procObj);
```

Description

Enumerates the strings of the specified text select object, calling a procedure for each string. A string, in this context, is the set of like-styled characters within a word. It is never larger than a single word. A word containing three styles is enumerated as three strings. There is no guaranteed correspondence between these strings and the actual show strings in the PDF file. Acrobat enumerates text in the order it appears in the PDF file, which is often not the same as the order in which a person would read the text.

Parameters

text	The text selection whose strings are enumerated.
proc	User-supplied callback to call for each string in the text object. Enumeration ends if proc returns false .
procObj	User-supplied data to pass to proc each time it is called.

Return Value

None

Known Exceptions

```
genErrBadParm
pdErrOpNotPermitted
```

Header File

PDCalls.h

Related Methods

```
PDDocCreateTextSelect
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDTextSelectEnumTextUCS

```
void PDTextSelectEnumTextUCS (PDTextSelect textP,
PDTextSelectEnumTextProc proc, void* procData);
```

Description

Same as [PDTextSelectEnumText](#) except output is forced to UCS.

Parameters

textP	The text selection whose strings are enumerated.
proc	User-supplied callback to call for each string in the text object. Enumeration ends if proc returns false .
procData	User-supplied data to pass to proc each time it is called.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

PDCalls.h

Related Methods

[PDDocCreateTextSelect](#)

[PDTextSelectEnumText](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDTextSelectGetBoundingRect

```
void PDTextSelectGetBoundingRect (PDTextSelect text,
ASFixedRect* boundRectP) ;
```

Description

Gets a text selection’s bounding rectangle. This is the smallest rectangle that completely encloses all characters in the selection.

Parameters

text	The text selection whose bounding rectangle is determined.
boundRectP	<i>(Filled by the method)</i> Pointer to the text selection’s bounding rectangle, specified in user space coordinates.

Return Value

None

Header File

PDCalls.h

Related Methods

- PDTextSelectGetPage
- PDTextSelectGetRange
- PDTextSelectGetRangeCount

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDTextSelectGetPage

```
ASInt32 PDTextSelectGetPage (PDTextSelect text);
```

Description

Gets the page number of a text selection's page.

Parameters

text	The text selection whose page number is obtained.

Return Value

The page number of the text selection's page.

Header File

PDCalls.h

Related Methods

[PDTextSelectGetBoundingRect](#)

[PDTextSelectGetRange](#)

[PDTextSelectGetRangeCount](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDTextSelectGetRange

```
void PDTextSelectGetRange (PDTextSelect textP, ASInt32 index,
    PDTextSelectRange range);
```

Description

Extracts the range specified by **index** from a text selection. Use [PDTextSelectGetRangeCount](#) to determine the number of ranges in a text selection.

Parameters

textP	The text selection from which a range is extracted.
index	The index of the range to extract from textP .
range	<i>(Filled by the method)</i> Pointer to a structure that contains the specified range.

Return Value

None

Header File

PDCalls.h

Related Methods

- [PDTextSelectGetBoundingRect](#)
- [PDTextSelectGetPage](#)
- [PDTextSelectGetRangeCount](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDTextSelectGetRangeCount

```
ASInt32 PDTextSelectGetRangeCount (PDTextSelect textP);
```

Description

Gets the number of ranges in a text selection. Use [PDTextSelectGetRange](#) to extract a single range from a text selection.

Parameters

textP	The text selection whose range count is obtained.

Return Value

The number of ranges in the text selection.

Header File

PDCalls.h

Related Methods

[PDTextSelectGetBoundingRect](#)
[PDTextSelectGetPage](#)
[PDTextSelectGetRange](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDThread

PDThreadDestroy

```
void PDThreadDestroy (PDThread thread);
```

Description

Deletes an article thread from its document. You must call [PDDocRemoveThread](#) to remove the thread from the document (if it was added to one using [PDDocAddThread](#)) before calling **PDThreadDestroy**.

Parameters

thread	The thread to destroy.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDThreadNew](#)
[PDThreadFromCosObj](#)
[PDDocRemoveThread](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDThreadFromCosObj

```
PDThread PDThreadFromCosObj (CosObj obj);
```

Description

Gets the thread object corresponding to the specified Cos object. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

obj	Dictionary Cos object for the thread.

Return Value

The **PDThread** object for the thread.

Known Exceptions

Raises **pdErrBadThread** if the thread is not valid as defined by **PDThreadIsValid**.

Header File

PDCalls.h

Related Methods

[PDThreadGetCosObj](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDThreadGetCosObj

```
CosObj PDThreadGetCosObj (PDThread thread) ;
```

Description

Gets the Cos object corresponding to a thread. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

thread	The thread whose Cos object is to obtained.

Return Value

Dictionary Cos object for the thread. The contents of the dictionary can be enumerated using **CosObjEnum**. Returns a **NULL** Cos object if **PDThreadIsValid(thread)** returns **false**.

Header File

PDCalls.h

Related Methods

CosObjEnum
PDThreadFromCosObj

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDThreadGetFirstBead

```
PDBead PDThreadGetFirstBead (PDThread thread);
```

Description

Gets an article thread's first bead.

Parameters

thread	The thread whose first bead is obtained.

Return Value

The thread's first bead, or a **NULL** Cos object if the thread has no beads.

Header File

PDCalls.h

Related Methods

[PDThreadSetFirstBead](#)
[PDBeadGetNext](#)
[PDBeadGetPrev](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDThreadGetInfo

```
ASInt32 PDThreadGetInfo (PDThread thread, char* infoKey,
char* buffer, ASInt32 bufSize);
```

Description

Gets the specified article thread's info.

NOTE: For Roman viewers, this text is always stored in the **PDFDocEncoding**. For non-Roman character set viewers, this text is stored as **PDFDocEncoding** or Unicode, depending on the file's creator. Files created in a non-Roman environment contain Unicode versions of these strings; in a Roman environment, files contain **PDFDocEncoding** versions of these strings.

Parameters

thread	The thread whose thread info is obtained.
infoKey	The key whose value is obtained.
buffer	<i>(Filled by the method)</i> The value associated with infoKey .
bufSize	The maximum number of characters that buffer can hold.

Return Value

The number of characters written into **buffer**.

Header File

PDCalls.h

Related Methods

[PDThreadSetInfo](#)

Example

```

size = PDDocGetNumThreads (pdDoc) ;
if (size)
{
    for (i = 0; i < size; i++)
    {
        /* get the right thread */
        thread = PDDocGetThread (pdDoc, i);
        len = PDThreadGetInfo (thread, "Title",
                               msg, sizeof (msg));
        if (!strcmp (msg, "Contents"))
            break;
    }
}

size = PDDocGetNumThreads (pdDoc) ;
if (size)
{
    for (i = 0; i < size; i++)
    {
        /* get the right thread */
        thread = PDDocGetThread (pdDoc, i);
        len = PDThreadGetInfo (thread, "Title",
                               msg, sizeof (msg));
        if (!strcmp (msg, "Contents"))
            PDThreadSetInfo (thread, "Title",
                             "In Progress", sizeof (
                             "In Progress"));
    }
}

```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDThreadIsValid

```
ASBool PDThreadIsValid (PDThread thread);
```

Description

Tests whether a thread is valid. This is intended only to ensure that the thread has not been deleted, not to ensure that all necessary information is present and valid.

Parameters

thread	The thread whose validity is tested.

Return Value

Returns **true** if the thread is valid, **false** otherwise.

Header File

PDCalls.h

Example

```
size = PDDocGetNumThreads(pdDoc);
if(size){
    for(i = 0;i<size;i++){/* get the right thread */
        thread = PDDocGetThread(pdDoc, i);
        if(!PDThreadIsValid(thread))
            AVAlertNote("invalid thread");
    }
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDThreadNew

```
PDThread PDThreadNew (PDDoc aDocP);
```

Description

Creates a new article thread. Use [PDDocAddThread](#) to add the thread to a document.

Parameters

doc	The document in which the thread is created.

Return Value

The newly created thread.

Header File

PDCalls.h

Related Methods

[PDThreadDestroy](#)
[PDThreadFromCosObj](#)
[PDDocAddThread](#)

Example

```
PDThread thread = PDThreadNew(doc);
```

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDThreadSetFirstBead

```
void PDThreadSetFirstBead (PDThread thread,
    PDBead newFirstBead);
```

Description

Sets an article thread's first bead.

Parameters

thread	The thread whose first bead is set.
newFirstBead	The bead to use as the first in thread .

Return Value

None

Header File

PDCalls.h

Related Methods

- [PDThreadGetFirstBead](#)
- [PDBeadInsert](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDThreadSetInfo

```
void PDThreadSetInfo (PDThread thread, char* infoKey,  
char* buffer, ASInt32 bufSize);
```

Description

Sets the specified article thread's info.

NOTE: For Roman viewers, this text is always stored in the **PDFDocEncoding**. For non-Roman character set viewers, this text is stored as **PDFDocEncoding** or Unicode, depending on the file's creator. Files created in a non-Roman environment contain Unicode versions of these strings; in a Roman environment, files contain **PDFDocEncoding** versions of these strings.

Parameters

thread	The thread whose thread info is set.
infoKey	The key whose value is set.
buffer	The value to set.
bufSize	The number of characters in buffer .

Return Value

None

Notifications

[PDThreadDidChange](#)

Header File

PDCalls.h

Related Methods

[PDThreadGetInfo](#)

Example

```

size = PDDocGetNumThreads (pdDoc) ;
if (size)
{
    for (i = 0; i < size; i++)
    {
        /* get the right thread */
        thread = PDDocGetThread (pdDoc, i);
        len = PDThreadGetInfo (thread, "Title",
                               msg, sizeof (msg));
        if (!strcmp (msg, "Contents"))
            break;
    }
}

size = PDDocGetNumThreads (pdDoc) ;
if (size)
{
    for (i = 0; i < size; i++)
    {
        /* get the right thread */
        thread = PDDocGetThread (pdDoc, i);
        len = PDThreadGetInfo (thread, "Title",
                               msg, sizeof (msg));
        if (!strcmp (msg, "Contents"))
            PDThreadSetInfo (thread, "Title",
                             "In Progress", sizeof (
                             "In Progress"));
    }
}

```

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDTrans

PDTransEqual

```
ASBool PDTransEqual (PDTrans pdtOne, PDTrans pdtTwo);
```

Description

Tests two transitions for equality. Two transitions are equal if and only if their Cos objects are equal (see [CosObjEqual](#)).

Parameters

pdtOne, pdtTwo	The transitions to test for equality.

Return Value

Returns **true** if the transitions are equal, **false** otherwise.

Header File

PDCalls.h

Related Methods

[CosObjEqual](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

PDTransFromCosObj

```
PDTrans PDTransFromCosObj (CosObj obj);
```

Description

Converts the specified dictionary Cos object to a transition and verifies that the transition is valid. This method does not copy the object but is instead the logical equivalent of a type cast.

Parameters

<code>obj</code>	Dictionary Cos object to convert to a transition.

Return Value

The transition corresponding to the given dictionary object, `obj`.

Known Exceptions

Raises `pdErrBadBaseObj` if the transition is not valid as determined by `PDTransIsValid`.

Header File

`PDCalls.h`

Related Methods

[PDTransGetCosObj](#)
[PDTransIsValid](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020002` or higher.



PDTransGetCosObj

```
CosObj PDTransGetCosObj (PDTrans pdt);
```

Description

Gets the dictionary Cos object corresponding to the transition and verifies that the transition is valid. This method does not copy the object but is the logical equivalent of a type cast.

Parameters

pdt	The transition whose Cos object is obtained.

Return Value

The dictionary Cos object corresponding to the transition. Returns the **NULL** Cos object if the transition is invalid as determined by [PDTransIsValid](#).

Header File

PDCalls.h

Related Methods

[PDTransFromCosObj](#)
[PDTransIsValid](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

PDTransGetDuration

```
ASFixed PDTransGetDuration (PDTrans pdt);
```

Description

Gets the duration for the given transition.

Parameters

pdt	The transition for which the duration is obtained.

Return Value

The transition duration, specified in seconds. If no duration is specified in the transition, the return value is **fxDefaultTransDuration** (1 second).

NOTE: Standard durations are

- 0 seconds - fast
- 1 second - medium
- 2 seconds - slow

Header File

PDCalls.h

Related Methods

[PDTransGetSubtype](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.



PDTransGetSubtype

```
ASAtom PDTransGetSubtype (PDTrans pdt);
```

Description

Gets a transition's subtype.

Parameters

pdt	The transition whose subtype is obtained.

Return Value

The **ASAtom** for the transition's subtype. Can be converted to a string using [ASAtomGetString](#).

Header File

PDCalls.h

Related Methods

[PDTransGetDuration](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

PDTransIsValid

```
ASBool PDTransIsValid (PDTrans pdt);
```

Description

Tests whether a transition is valid, that is, the transition has not been deleted.

Parameters

pdt	The transition dictionary whose validity is tested.

Return Value

Returns **true** if the transition is valid, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDTransEqual](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

PDTransNew

```
PDTrans PDTransNew (PDDoc pdd, ASAtom asaSubtype,
ASFixed fxDuration);
```

Description

Creates a new transition of the specified type and duration associated with the **CosDoc** of the given **PDDoc**.

Parameters

pdd	The PDDoc to whose CosDoc the transition is added.
asaSubtype	The transition subtype to create. Must be one of the transition effects described in Section 8.3.3 in the PDF Reference . All implementations that support transitions are required to support these transitions at a minimum. Plug-ins can register new types or provide a new handler for an existing type.
fxDuration	The transition duration, in seconds.

Return Value

The newly created transition.

Known Exceptions

Raises [pdErrBadBaseObj](#) if the transition is not valid as determined by [PDTransIsValid](#).

Header File

`PDCalls.h`

Related Methods

[PDTransNewFromCosDoc](#)

Product

base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020002` or higher.

PDTransNewFromCosDoc

```
PDTrans PDTransNewFromCosDoc (CosDoc cd, ASAtom asaSubtype,
ASFixed fxDuration);
```

Description

Creates a new transition of the specified type and duration associated with the given CosDoc.

Parameters

cd	The CosDoc to which the transition is added.
asaSubtype	The transition subtype to create. This subtype is returned by the AVTransHandlerGetTypeProc callback in AVTransHandler.
fxDuration	The transition duration, in seconds.

Return Value

The newly created transition.

Known Exceptions

Raises pdErrBadBaseObj if the transition is not valid as determined by PDTransIsValid.

Header File

PDCalls.h

Related Methods

PDTransNew

Product

reader, base, pro, pdf

Availability

Available if PI_PDMODEL_VERSION (in PIRequir.h) is set to 0x00020002 or higher.



PDTransNull

```
PDTrans PDTransNull (void);
```

Description

Gets a **NULL** transition. This can be used in conjunction with [PDTransEqual](#) to determine whether a transition is **NULL**.

Parameters

None

Return Value

A **NULL** transition.

Header File

PDCalls.h

Related Methods

[PDTransNew](#)

Example

```
PDTrans trans = PDPageGetTransition(page);
if (!PDTransEqual(PDTransNull(), trans)) {
    /* Page has a transition */
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020002** or higher.

PDViewDest

PDViewDestCreate

```
PDViewDestination PDViewDestCreate (PDDoc doc,
PDPage initialPage, ASAtom initialFitType,
const ASFixedRectP initialRect, const ASFixed initialZoom,
ASInt32 pageNumber);
```

Description

Creates a new view destination object.

Parameters

doc	The document in which the destination is used.
initialPage	The destination page.
initialFitType	Destination fit type. Must be one of the View Destination Fit Types , which must be converted into an ASAtom with ASAtomFromString .
initialRect	Pointer to a ASFixedRect specifying the destination rectangle, specified in user space coordinates. The appropriate information will be extracted from initialRect , depending on initialFitType , to create the destination. All four of initialRect 's components should be set; using PDViewDestNULL for any components can result in incorrect results for rotated pages.
initialZoom	The zoom factor to set for the destination. Used only if initialFitType is XYZ . Use the predefined value PDViewDestNULL (see <code>PDExpT.h</code>) to indicate a NULL zoom factor, described in Table 8.2 in the PDF Reference .
pageNum	Currently unused.

Return Value

The newly created view destination.

Header File

`PDCalls.h`

Related Methods

[PDViewDestFromCosObj](#)
[PDViewDestDestroy](#)



Example

```
dstPage = PDDocAcquirePage(pdDoc, (offset +
    Pages[i].pg)-1);
dest = PDViewDestCreate(pdDoc, dstPage,
    fit, &initRect, zoom, Pages[i].pg-1);
if(PDViewDestIsValid(dest))
{
    pdAction = PDActionNewFromDest(pdDoc,
        dest, pdDoc);
    PDBookmarkSetAction(newBm, pdAction);
}
```

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDViewDestDestroy

```
void PDViewDestDestroy (PDViewDestination dest);
```

Description

Deletes a view destination object. Before deleting a view destination, ensure that no link or bookmark refers to it.

Parameters

dest	The view destination to destroy.

Return Value

None

Header File

PDCalls.h

Related Methods

[PDViewDestFromCosObj](#)
[PDViewDestCreate](#)

Product

base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDViewDestFromCosObj

```
PDViewDestination PDViewDestFromCosObj (CosObj obj);
```

Description

Converts the specified Cos object to a view destination and verifies that the view destination is valid. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

obj	Dictionary Cos object to convert to a view destination.

Return Value

Array Cos object for the view destination.

Known Exceptions

Raises [pdErrBadAction](#) if the destination is invalid, as determined by [PDViewDestIsValid](#).

Header File

`PDCalls.h`

Related Methods

[PDViewDestGetCosObj](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDViewDestGetAttr

```
void PDViewDestGetAttr (PDViewDestination dest,
ASInt32* pageNum, ASAtom* fitType, ASFixedRectP destRect,
ASFixed* zoom);
```

Description

Gets a view destination’s fit type, destination rectangle, and zoom factor. The destination must be represented by an array, which is the case for a **GoToR** action.

Parameters

dest	The view destination whose attributes are obtained.
pageNum	(Filled by the method) The page number of the destination’s page.
fitType	(Filled by the method) Destination fit type. One of the values listed in View Destination Fit Types .
destRect	(Filled by the method) Pointer to a ASFixedRect containing the destination’s rectangle, specified in user space coordinates.
zoom	(Filled by the method) The destination’s zoom factor.

Return Value

None

Known Exceptions

Raises [genErrBadParm](#) if the destination is not represented by an array.

Header File

PDCalls.h

Related Methods

[PDViewDestCreate](#)



Example

```
PDLinkAnnot pdAnnot;  
PDAction oldAction;  
PDViewDestination viewDest;  
ASInt32 page;  
ASAtom fit;  
ASFixedRect destRect;  
ASFixed zoom;  
  
oldAction = PDLinkAnnotGetAction(pdAnnot);  
viewDest = PDActionGetDest(oldAction);  
PDViewDestGetAttr(viewDest, &page, &fit,  
    &destRect, &zoom);  
PDActionDestroy(oldAction);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDViewDestGetCosObj

```
CosObj PDViewDestGetCosObj (PDViewDestination dest);
```

Description

Gets the Cos object corresponding to a view destination and verifies that the view destination is valid. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

dest	The view destination whose Cos object is obtained.

Return Value

Array Cos object for the view destination. The contents of the array can be enumerated using **CosObjEnum**. Returns a **NULL** Cos object if the view destination is invalid, as determined by **PDViewDestIsValid**.

Header File

PDCalls.h

Related Methods

PDViewDestFromCosObj
PDViewDestIsValid

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDViewDestIsValid

```
ASBool PDViewDestIsValid (PDViewDestination dest);
```

Description

Tests whether a view destination is valid. This is intended only to ensure that the view destination has not been deleted, not to ensure that all necessary information is present and valid.

Parameters

dest	The view destination whose validity is determined.

Return Value

Returns **true** if the view destination is valid, **false** otherwise.

Header File

PDCalls.h

Example

```
dstPage = PDDocAcquirePage(pdDoc, (offset +
    Pages[i].pg) - 1);
dest = PDViewDestCreate(pdDoc, dstPage,
    fit, &initRect, zoom, Pages[i].pg - 1);
if (PDViewDestIsValid(dest))
{
    pdAction = PDActionNewFromDest(pdDoc,
        dest, pdDoc);
    PDBookmarkSetAction(newBm, pdAction);
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDViewDestResolve

```
PDViewDestination PDViewDestResolve(PDViewDestination dest,
PDDoc doc);
```

Description

Resolves a destination. **dest** is the value of the **D** key in an action. It can be a real destination (an array) or a name. If it is a name, look it up in **doc**'s Dests dictionary. The value found there can be a real dest (an array) or a dictionary. If it's a dictionary, look up the **D** key in that dictionary.

This method is useful for getting a **PDViewDestination** from an action, as provided by **PDActionGetDest**—since this method may not return a **PDViewDestination**.

Parameters

dest	The destination to resolve.
doc	The PDDoc that contains the destination.

Return Value

The resolved view destination.

Known Exceptions

Can raise memory, I/O, and parsing exceptions.

Header File

PDCalls.h

Related Methods

[PDActionGetDest](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

PDWord

PDWordCreateTextSelect

```
PDTextSelect PDWordCreateTextSelect (PDPage page,
PDWord* wList, ASUns32 wListLen);
```

Description

Creates a text selection object for a given page that includes all words in a word list, as returned from a [PDWordFinder](#) method. The text selection can then be set as the current selection using [AVDocSetSelection](#).

NOTE: For consistent text selection behavior, avoid using other [PDTextSelect](#) creation methods which depend on the word finder versions and word offsets. These include: [PDTextSelectCreatePageHiliteEx](#), [PDTextSelectCreateRanges](#), [PDTextSelectCreateRangesEx](#), [PDTextSelectCreateWordHilite](#), and [PDTextSelectCreateWordHiliteEx](#).

Parameters

page	The page on which to select the words.
wList	The word list to be selected.
wListLen	The number of words in the word list.

Return Value

The newly created text selection.

Header File

PDCalls.h

Related Methods

[PDDocCreateTextSelect](#)
[PDTextSelectDestroy](#)
[AVDocSetSelection](#)
[PDTextSelectEnumQuads](#)
[PDTextSelectEnumText](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

PDWordFilterString

```
ASBool PDWordFilterString (ASUns16* infoArray, char* cNewWord,
char* cOldWord);
```

Description

NOTE: In Acrobat 6.0, the method [PDWordFinderEnumWordsStr](#) is preferred to this method, which remains for backward compatability.

Removes leading and trailing spaces and leading and trailing punctuation (including soft hyphens) from the specified word. Does not remove punctuation characters surrounded by alphanumeric characters within a word. Does not remove leading punctuation characters (character type [W_COMMA](#), [W_PERIOD](#), and [W_HYPHEN](#)) if they are followed by numeric characters. Does not remove wildcard characters (that is, characters of type [W_WILD_CARD](#)). The characters * and ? are not wildcard characters by default. Wildcard, alphanumeric, and punctuation characters must be specified by the custom character type array, **infoArray**.

Although this method seems very similar to [PDWordFilterWord](#), the two methods treat letters and digits slightly differently. [PDWordFilterWord](#) uses the encoding info array but also does a straight character code test for any characters that have not been mapped to anything. It does this to catch letters and digits from non-standard character sets, and is necessary to avoid removing words with non-standard character sets.

PDWordFilterString, on the other hand, was designed for known character sets such as WinAnsi and Mac Roman.

For non-Roman character set viewers, this method currently supports only SHIFT-JIS encoding on a Japanese system.

Parameters

infoArray	An array specifying the type of each character in the font. Each entry in this table must be one of the Character Type Codes . If infoArray is set to NULL , a default table is used. For non-UNIX Roman systems, it is MacRomanEncoding in Mac OS and WinAnsiEncoding in Windows. In UNIX (except HP-UX) Roman systems, it is ISO8859-1 (ISO Latin-1); for HP-UX, it is HP-ROMAN8 . See Appendix D in the PDF Reference for descriptions of MacRomanEncoding and WinAnsiEncoding .
cNewWord	<i>(Filled by the method)</i> The filtered word.
cOldWord	The unfiltered word. This value must be passed to the method.

Return Value

Returns **true** if the string required filtering, **false** if the filtered string is the same as the unfiltered string.

**Header File**

PDCalls.h

Related Methods

[PDWordFilterWord](#)

Example

```
extern ASUns16 myInfo[];  
char newWord[128], oldWord[128];  
PDWordFilterString(&myInfo, newWord,  
oldWord);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDWordFilterWord

```
ASBool PDWordFilterWord (PDWord word, char* buffer,
ASInt16 bufferLen, ASInt16* newLen);
```

Description

NOTE: In Acrobat 6.0, the method [PDWordFinderEnumWords](#) is preferred to this method, which remains for backward compatability.

Removes leading and trailing spaces and leading and trailing punctuation (including soft hyphens) from the specified word. Does not remove punctuation characters surrounded by alphanumeric characters within a word. Does not remove leading punctuation characters (character type [W_COMMA](#), [W_PERIOD](#), and [W_HYPHEN](#)) if they are followed by numeric characters. Does not remove wildcard characters (that is, characters of type [W_WILD_CARD](#)). The characters * and ? are not wildcard characters by default. Wildcard, alphanumeric, and punctuation characters must be specified by the custom character type array, [infoArray](#).

The determination of which characters to remove is made by examining the flags in the [outEncInfo](#) array passed to [PDDocCreateWordFinder](#). As a result, this method is most useful after you have been called with words obtained by calling [PDWordFinderGetNthWord](#), in the callback for [PDWordFinderEnumWords](#), and words in the [pXYSortTable](#) returned by [PDWordFinderAcquireWordList](#). See the description of [PDWordFilterString](#) for further information, and for a description of how the two methods differ.

The Acrobat Catalog program uses this method to filter words before indexing them.

This method works with non-Roman systems.

Parameters

word	The PDWord to filter.
buffer	<i>(Filled by the method)</i> The filtered string.
bufferLen	The maximum number of characters that buffer can hold.
newLen	<i>(Filled by the method)</i> Number of characters actually written into buffer .

Return Value

Returns **true** if the word required filtering, **false** if the filtered string is the same as the unfiltered string.

Header File

PDCalls.h



Related Methods

[PDWordFilterString](#)

Example

```
ACCB1 ASBool ACCB2
WordCounterProc(PDWordFinder WordFinder,
PDWord word, ASInt32 PageNum,
void* fileNum)
{
    PDWordGetNthQuad(word, 0, &quad);
    PDWordFilterWord(word, gbuf,
        sizeof(gbuf), &len);
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDWordGetASText

```
void PDWordGetASText (PDWord word, ASUns32 filter,
ASText str);
```

Description

Copies the text from a word into an **ASText** object. Automatically performs necessary encoding conversions from the specified word (either in Unicode or Host Encoding) to the **ASText** object.

Parameters

word	The word whose text becomes the new ASText .
filter	Character types to be dropped from the output string. For example, the following returns text without soft hyphens and accent marks: <code>PDWordGetASText (word, W_SOFT_HYPHEN + W_ACCENT, mystr);</code>
str	An existing ASText object, whose content will be replaced by the new text.

Return Value

None.

Header File

PDCalls.h

Example

```
PDWordGetASText (myWord, W_SOFT_HYPHEN + W_ACCENT, myStr);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDWordGetAttr

```
ASUns16 PDWordGetAttr (PDWord word);
```

Description

Gets a bit field containing information on the types of characters in a word. Use [PDWordGetCharacterTypes](#) if you wish to check each character's type individually.

Parameters

word	The word whose character types are obtained.

Return Value

A bit field containing information on the types of characters in **word**. The value is a logical OR of the [Word Attributes](#).

NOTE: **PDWordGetAttr** may return an attribute value greater than the maximum of all of the public attributes since there can be private attributes added on. It is recommended to AND the result with the attribute you are interested in.

Header File

PDCalls.h

Related Methods

[PDWordGetCharacterTypes](#)
[PDWordGetStyleTransition](#)
[PDWordGetNthCharStyle](#)

Example

```
attr = PDWordGetAttr(myWord) & WXE_HAS_TRAILING_PUNC;
if(attr)
{
    /* process words with trailing punc. */
    ...
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDWordGetAttrEx

`ASUns16 PDWordGetAttrEx (PDWord word, ASUns32 groupID);`

Description

This is a version 6.0 extension of [PDWordGetAttr](#) that can be used only with a word finder created with algorithm version `WF_VERSION_3` or higher. It can get an additional 16-bit flag group defined in Acrobat 6.

Gets a bit field containing information on the types of characters in a word. Use [PDWordGetCharacterTypes](#) if you wish to check each character's type individually.

Parameters

word	The word whose character types are obtained.
groupID	The group number of the Word Attributes flags. 0, the default, is the first 16-bit group, and is the same as PDWordGetAttr. 1 gets the second group defined in Acrobat 6.

Return Value

A bit field containing information on the types of characters in **word**. The value is a logical OR of the [Word Attributes](#).

NOTE: `PDWordGetAttr` may return an attribute value greater than the maximum of all of the public attributes since there can be private attributes added on. It is recommended that you AND the result with the attribute you are interested in.

Header File

`PDCalls.h`

Related Methods

[PDWordGetCharacterTypes](#)
[PDWordGetStyleTransition](#)
[PDWordGetNthCharStyle](#)

Example

```
attr = PDWordGetAttrEx(myWord, 1) & WXE_HAS_TRAILING_PUNC;
if(attr)
{
    /* process words with trailing punc. */
    ...
}
```

Product

reader, base, pro, pdf

**Availability**

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDWordGetByteIdxFromHiliteChar

```
ASUns32 PDWordGetByteIdxFromHiliteChar (PDWord word,
ASUns32 charIdx);
```

Description

This method can be used only with a word finder created with algorithm version **WF_VERSION_3** or higher.

Returns the byte offset within the specified word of the highlightable character at the specified character offset. The first character of a word is at byte offset 0.

The returned byte offset can be passed to [PDWordGetCharOffsetEx](#) and [PDWordGetCharQuad](#) to get additional information. Use [PDWordGetNumHiliteChar](#) to get the number of highlightable characters in a word.

Parameters

word	The word containing the character.
charIdx	The character index within the word.

Return Value

The byte offset of the specified character within the word, or 0 if the character index is out of range.

Header File

PDCalls.h

Related Methods

- [PDWordGetCharOffsetEx](#)
- [PDWordGetCharQuad](#)
- [PDWordGetNumHiliteChar](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDWordGetCharacterTypes

```
void PDWordGetCharacterTypes (PDWord word, ASUns16* cArr,
ASInt16 size);
```

Description

Gets the character type for each character in a word.

Parameters

word	The word whose character types are obtained.
cArr	<i>(Filled by the method)</i> An array of character types. This array contains one element for each character in the word. Use PDWordGetLength to determine the number of elements that must be in the array. Each element is the logical OR of one or more of the Character Type Codes . For non-Roman character set viewers, meaningful values are returned only for Roman characters. For non-Roman characters, it returns 0, which is the same as W_CNTL . If the character is 2 bytes, both bytes indicate the same character type.
size	Number of elements in cArr .

Return Value

None

Header File

PDCalls.h

Related Methods

- [PDWordGetAttr](#)
- [PDWordGetLength](#)

Example

```
ASUns16 ctTab[255];

attr = PDWordGetAttr(myWord);
wlen = PDWordGetLength(myWord);
if(attr & WXE_HAS_DIGIT){
    PDWordGetCharacterTypes(myWord, ctTab,
        (ASUns16)wlen);
    IsNumber = false;
    for(i=0;i<wlen;i++){
        if(!(ctTab[i] & W_DIGIT))
            break;
    }
    if(i == wlen)
        IsNumber = true;
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDWordGetCharDelta

```
ASUns8 PDWordGetCharDelta (PDWord word);
```

Description

Gets the difference between the word length (the number of printed characters in the word) and the PDF word length (the number of character codes in the word). For instance, if the PDF word is “*fi* (ligature) *sh*” the mapped word will be “fish”. The ligature occupies only one character code, so in this case the character delta will be $3-4 = -1$.

Parameters

word	The word whose character delta is obtained.

Return Value

The character delta for **word**. Cast the return value to an [ASInt8](#) before using.

If the [PDWord](#)'s character set has no ligatures, such as on a non-Roman viewer supporting Japanese, returns **0**.

Header File

PDCalls.h

Related Methods

[PDWordGetLength](#)
[PDWordGetCharOffset](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDWordGetCharEncFlags

```
void PDWordGetCharEncFlags (PDWord word, ASUns32 *fList,
ASUns32 size);
```

Description

Gets the [WordFinder Character Encoding Flags](#) for each character in a word, which specify how reliably the word finder identified the character encoding.

This method can be used only with a word finder created with algorithm version **WF_VERSION_3** or higher.

Parameters

word	The word whose character encoding flags are obtained.
fList	<i>(Filled by the method)</i> An array of character encoding flags types. This array contains one element for each byte of text in the word. The byte length of the text can be determined with PDWordGetLength . Each element is the logical OR of one or more of the character encoding flags.
size	The maximum number of elements in the array fList .

Return Value

None

Header File

PDCalls.h

Related Methods

[PDWordGetAttrEx](#)
[PDWordGetLength](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDWordGetCharOffset

[ASUns16](#) PDWordGetCharOffset ([PDWord](#) word) ;

Description

Returns a word's character offset from the beginning of its page. This information, together with the character delta obtained from [PDWordGetCharDelta](#), can be used to highlight a range of words on a page, using [PDTextSelectCreatePageHilite](#).

Parameters

word	The word whose character offset is obtained.

Return Value

The word's character offset. On multi-byte systems, it points to the first byte.

Header File

PDCalls.h

Related Methods

[PDWordGetCharDelta](#)
[PDWordGetLength](#)
[PDTextSelectCreatePageHilite](#)

Example

```
ACCB1 ASBool ACCB2 WordCounterProc(
    PDWordFinder WordFinder, PDWord word,
    ASInt32 pageNum, void* fileNum)
{
    ASUns32 cur;

    cur = (ASUns32) PDWordGetCharOffset(word);
    WordCnt += (cur - gLast);
    gLast = cur;
}

WordFinder = PDDocCreateWordFinder(
    CurrentPDDoc, NULL, NULL, NULL, 0,
    WXE_PDF_ORDER, NULL);
PDWordFinderEnumWords(WordFinder, pageNum,
    ASCallbackCreateProto(PDWordProc,
        &WordCounterProc), NULL);
PDWordFinderDestroy(WordFinder);
```

Product

reader, base, pro, pdf

**Availability**

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDWordGetCharOffsetEx

```
ASUns32 PDWordGetCharOffsetEx (PDWord word, ASUns32 byteIdx,
ASUns32* bytesConsumed, ASUns32* offsetLen);
```

Description

This is a version 6.0 extension of [PDWordGetCharOffset](#) that can be used only with a word finder created with algorithm version **WF_VERSION_3** or higher.

Returns the character offset for a character identified by its index number, and the number of bytes (length) used for that character. The length is usually 1 for single-byte characters and 2 for double-byte characters. If multiple bytes are used to construct one character, only the first byte has valid character offset information and the other bytes have zero offset length with the same character offset of the first byte. If the returned offset length is zero, it means the specified byte in the word is a part (other than the first byte) of a multi-byte character.

The character offset is the character position calculated in bytes from the beginning of a page. Because of the encoding conversions and character replacements applied by the word finder, some characters may have different byte lengths from the original PDF content. The character offset itself can locate a character in the PDF content. However, without the offset length (that is the number of bytes in the PDF content), clients cannot tell whether two characters are next to each other in the PDF content or not. For example, suppose you want to create a Text Select object of two characters at character offset 1 and 3. You can create an object with two disconnected ranges of [Offset 1, Length 1] and [Offset 3, Length 1]. However, if you know that the offset length of both characters is 2, you can create a simpler object with a single range of [Offset 1, Length 4].

Parameters

word	The word whose character offset is obtained.
byteIdx	The byte index within the word of the character whose offset is obtained. Valid values are 0 to PDWordGetLength(word) - 1 .
bytesConsumed	(Filled by method) Returns the number of bytes in the word that are occupied by the specified character. Can be NULL if not needed. Use (byteIdx + *bytesConsumed) to get the byte index of the next character in the word.
offsetLen	(Filled by the method) Returns the number of bytes occupied by the specified character in the original PDF content. This is 0 if the specified byte is not the starting byte of a character in the PDF content. Can be NULL if not needed.

**Return Value**

The word's character offset and number of bytes occupied by the character.

Header File

PDCalls.h

Related Methods

[PDWordGetCharOffset](#)

[PDWordGetCharDelta](#)

[PDWordGetLength](#)

[PDTextSelectCreatePageHilite](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDWordGetCharQuad

```
ASBool PDWordGetCharQuad (PDWord word, ASUns32 byteIdx,
ASFixedQuad* quad) ;
```

Description

Gets the quadrilateral bounding of the character at a given index position in the word. If the specified character is constructed with multiple bytes, only the first byte returns a valid quad. Otherwise, this method returns **false**.

This method can be used only with a word finder created with algorithm version **WF_VERSION_3** or higher.

Parameters

word	The word whose character offset is obtained.
byteIdx	The byte index within the word of the character whose quad is obtained. Valid values are 0 to PDWordGetLength(word) - 1 .
quad	(Filled by method) A pointer to an existing quad structure in which to return the character's quad specified in user-space coordinates.

Return Value

Returns **true** if the provided byte index is the beginning byte of a character and a valid quad is returned, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDWordGetCharOffsetEx](#)
[PDWordGetNthQuad](#)
[PDWordGetNumQuads](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



PDWordGetLength

ASUns8 PDWordGetLength (**PDWord** word);

Description

Gets the number of bytes in a word. This method also works on non-Roman systems.

Parameters

word	The word object whose character count is obtained.

Return Value

The number of characters in **word**.

Header File

PDCalls.h

Related Methods

[PDWordGetCharDelta](#)
[PDWordGetCharOffset](#)

Example

```
ASUns16 ctTab[255];
attr = PDWordGetAttr(myWord);
wlen = PDWordGetLength(myWord);
if(attr & WXE_HAS_DIGIT)
{
    PDWordGetCharacterTypes(myWord, ctTab,
        (ASUns16)wlen);
    IsNumber = false;
    for(i=0;i<wlen;i++)
    {
        if(!(ctTab[i] & W_DIGIT))
            break;
    }
    if(i == wlen)
        IsNumber = true;
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDWordGetNthCharStyle

```
PDStyle PDWordGetNthCharStyle (PDWordFinder wObj, PDWord word,
ASInt32 dex);
```

Description

Returns a **PDStyle** object for the n^{th} style in a word.

Parameters

wObj	A word finder object.
word	The word whose n^{th} style is obtained.
dex	The index of the style to obtain. The first style in a word has an index of zero.

Return Value

The n^{th} style in the word. Returns **NULL** if **dex** is greater than the number of styles in the word.

Known Exceptions

Raises **genErrBadParm** if **dex** < 0.

Header File

PDCalls.h

Related Methods

[PDWordGetStyleTransition](#)

Example

```
for(pdwPtr = pXYSortTable;size;size--)
{
    myWord = *pdwPtr++;
    PDWordGetNthQuad(myWord, 0, &quad);
    style = PDWordGetNthCharStyle(pdWF,
        myWord, 0);
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDWordGetNthQuad

```
ASBool PDWordGetNthQuad (PDWord word, ASInt16 nTh,
ASFixedQuad* quad) ;
```

Description

Gets the specified word's n^{th} quad, specified in user space coordinates. See [PDWordGetNumQuads](#) for a description of a quad.

The quad's height is the height of the font's bounding box, not the height of the tallest character used in the word. The font's bounding box is determined by the glyphs in the font that extend farthest above and below the baseline; it often extends somewhat above the top of "A" and below the bottom of "y."

The quad's width is determined from the characters actually present in the word.

As an example, the quads for the words "AWAY" and "away" have the same height, but generally do not have the same width unless the font is a mono-spaced font (a font in which all characters have the same width).

Despite the names of the fields in an [ASFixedQuad](#) (**tl** for top left, **bl** for bottom left, and so forth) the corners of **quad** do not necessarily have these positions.

Parameters

word	The word whose n^{th} quad is obtained.
nTh	The quad to obtain. A word's first quad has an index of zero.
quad	<i>(Filled by the method)</i> Pointer to the word's n^{th} quad, specified in user-space coordinates.

Return Value

Returns **true** if the word has an n^{th} quad, **false** otherwise.

Header File

```
PDCalls.h
```

Related Methods

```
PDWordGetNumQuads
```

Example

```
ACCB1 ASBool ACCB2 WordCounterProc(
    PDWordFinder WordFinder, PDWord word,
    ASInt32 PageNum, void* fileNum)
{
    PDWordGetNthQuad(word, 0, &quad);
    PDWordFilterWord(word, gbuf,
        sizeof(gbuf), &len);
}
WordFinder = PDDocCreateWordFinder(
    CurrentPDDoc, NULL, NULL, NULL, 0,
    WXE_PDF_ORDER, NULL);
PDWordFinderEnumWords(WordFinder, PageNum,
    ASCallbackCreateProto(PDWordProc,
        &WordCounterProc), NULL);
PDWordFinderDestroy(WordFinder);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDWordGetNumHiliteChar

[ASUns32](#) PDWordGetNumHiliteChar ([PDWord](#) word);

Description

This method can be used only with a word finder created with algorithm version **WF_VERSION_3** or higher.

Gets the number of highlightable characters in a word. A highlightable character is the minimum text unit that Acrobat can select and highlight.

Because of the encoding conversion, the characters in a word finder word list do not have a 1-to-1 correspondence to the characters displayed by Acrobat. For example, if the word is "fish" and the text operation in PDF content is 'fi'(ligature) + 's' + 'h', this method returns the number of highlightable characters as 3, counting 'fi' as one character. For the same word, the **PDWordGetLength** method returns the byte-length as 4.

Parameters

word	The word whose highlightable character count is obtained.

Return Value

The number of highlightable characters in **word**.

Header File

PDCalls.h

Related Methods

[PDWordGetLength](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDWordGetNumQuads

ASInt16 PDWordGetNumQuads (**PDWord** word);

Description

Gets the number of quads in a word. A quad is a quadrilateral bounding a contiguous piece of a word. Every word has at least one quad. A word has more than one quad, for example, if it is hyphenated and split across multiple lines or if the word is set on a curve rather than on a straight line.

Parameters

word	The word whose quad count is obtained.

Return Value

The number of quads in **word**.

Header File

PDCalls.h

Related Methods

[PDWordGetNthQuad](#)

Example

```
for(pdwPtr = pXYSortTable;size;size--){
    myWord = *pdwPtr++;
    if(PDWordGetNumQuads(myWord) > 1)
        continue;
    PDWordGetNthQuad(myWord, 0, &quad);
    style = PDWordGetNthCharStyle(pdWF,
        myWord, 0);
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDWordGetString

```
void PDWordGetString (PDWord word, char* str, ASInt32 len);
```

Description

Gets a word's text and also converts ligatures to their constituent characters. The string to return includes any word break characters (such as space characters) that follow the word, but not any that precede the word. The characters that are treated as word breaks are defined in the `outEncInfo` parameter of `PDDocCreateWordFinder` method. Use `PDWordFilterString` to subsequently remove the word break characters. `PDWordFilterString` does *not* convert ligatures to their constituent characters, so the character codes for the ligature remain in the string.

This method produces a string in whatever encoding the `PDWord` uses, for both Roman and non-Roman systems.

Parameters

word	The word whose string is obtained.
str	<i>(Filled by the method)</i> The string. The encoding of the string is the encoding used by the <code>PDWordFinder</code> that supplied the <code>PDWord</code> . For instance, if <code>PDDocCreateWordFinderUCS</code> is used to create the word finder, <code>PDWordGetString</code> returns only Unicode. There is no way to detect Unicode strings returned by <code>PDWordGetString</code> , since there is no UCS header (FEFF) added to each string returned.
len	Length of str , in bytes. Up to len characters of word will be copied into str . If str is long enough, it will be null-terminated.

Return Value

None

Known Exceptions

Raises `genErrBadParm` if either `word` or `str` is `NULL`.

Header File

`PDCalls.h`

Related Methods

[PDWordGetLength](#)
[PDWordGetCharDelta](#)
[PDWordSplitString](#)

**Example**

```
char buf[128];

PDWordGetString(myWord, buf, sizeof(buf));
attr = PDWordGetAttr(myWord);
if(attr & WXE_HAS_TRAILING_PUNC){
    buf[PDWordGetLength(myWord)-1] = 0x00;
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDWordGetStyleTransition

```
ASInt16 PDWordGetStyleTransition (PDWord word,  
ASInt16* transTbl, ASInt16 size);
```

Description

Gets the locations of style transitions in a word. Every word has at least one style transition, at character position zero in the word.

Parameters

word	The word whose style transition list is obtained.
transTbl	<i>(Filled by the method)</i> Array of style transitions. Each element is the character offset in word where the style changes. The offset specifies the first character in the word that has the new style. The first character in a word has an offset of zero.
size	Number of entries that transTbl can hold. The word is searched only until this number of style transitions have been found.

Return Value

Number of style transition offsets copied to **transTbl**.

Header File

PDCalls.h

Related Methods

[PDWordGetNthCharStyle](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDWordIsCurrentlyVisible

```
ASBool PDWordIsCurrentlyVisible (PDWord word, ASInt32 pageNum,
PDObjectContext ctx);
```

Description

Tests whether a word is visible in a given optional-content context on a given page.

Parameters

word	The word to test.
pageNum	The page number for which the word is tested.
ctx	The context in which the word is tested, as returned by PDDocGetOCContext (pdDoc) .

Return Value

Returns **true** if the word is visible in the given context, **false** if it is hidden.

Header File

PDCalls.h

Related Methods

```
PDWordMakeVisible
PDWordFinderAcquireVisibleWordList
PDWordFinderEnumVisibleWords
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in PIRequir.h) is set to **0x00060000** or higher.

PDWordIsRotated

```
ASBool PDWordIsRotated (PDWord word);
```

Description

Tests whether a word is rotated.

Parameters

word	The word to test.

Return Value

Returns **true** if the word is rotated, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDWordGetNthQuad](#)

Example

```
for (pdwPtr = pXYSortTable; numWords;
     numWords--)
{
    myWord = *pdwPtr++;
    if (PDWordIsRotated(myWord))
        break;
}
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDWordMakeVisible

```
ASBool PDWordMakeVisible (PDWord word, ASInt32 pageNum,
PDObjectContext ctx);
```

Description

Makes a word is visible in a given optional-content context on a given page.

Parameters

word	The word to test.
pageNum	The page number for which the word is to be made visible.
ctx	The context in which the word is to be made visible, as returned by <code>PDDocGetOCContext (pdDoc)</code> .

Return Value

Returns **true** if the word can be made visible in the given context, **false** otherwise.

Header File

PDCalls.h

Related Methods

[PDWordIsCurrentlyVisible](#)
[PDWordFinderAcquireVisibleWordList](#)
[PDWordFinderEnumVisibleWords](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.



PDWordSplitString

```
ASUns16 PDWordSplitString (ASUns16* infoArray, char* cNewWord,
char* cOldWord, ASUns16 nMaxLen);
```

Description

Splits the specified string into words by substituting spaces for word separator characters. The list of characters considered to be word separators can be specified, or a default list can be used.

The characters “,” and “.” are context-sensitive word separators. If surrounded by digits (for example, 654,096.345), they are not considered word separators.

For non-Roman character set viewers, this method currently supports only SHIFT-JIS encoding on a Japanese system.

Parameters

infoArray	A character information table. It specifies each character's type; word separator characters must be marked as W_WORD_BREAK (see Character Type Codes). This table can be identical to the table to pass to PDDocCreateWordFinder . If infoArray is NULL , a default table is used (see Glyph Names of Word Separators).
cNewWord	<i>(Filled by the method)</i> Word that has been split. Word separator characters have been replaced with spaces.
cOldWord	Word to split.
nMaxLen	Number of characters that cNewWord can hold. Word splitting stops when cOldWord is completely processed or nMaxLen characters have been placed in cNewWord , whichever occurs first.

Return Value

The number of splits that occurred.

Known Exceptions

Raises [genErrGeneral](#) if **infoArray** is **NULL**, but host encoding cannot be obtained.

Header File

PDCalls.h

Related Methods

[PDWordGetString](#)

Product

reader, base, pro, pdf



Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDWordFinder

PDWordFinderAcquireVisibleWordList

```
void PDWordFinderAcquireVisibleWordList (PDWordFinder wObj,
ASInt32 pgNum, PDOCContext ocContext, PDWord* wInfoP,
PDWord** pXYSortTable, PDWord** rdOrderTable,
ASInt32* numWords) ;
```

Description

Finds all words on the specified page that are visible in the given optional-content context and returns one or more tables containing the words. One table contains the words sorted in the order in which they appear in the PDF file, while the other contains the words sorted by their x- and y-coordinates on the page.

The list contains only words that are visible in the given context. If the word states change in the given context, the word list will have to be released and re-acquired to reflect the changed set of visible words.

There can be only one word list in existence at a time; clients must release the previous word list, using [PDWordFinderReleaseWordList](#), before creating a new one.

Use [PDWordFinderEnumWords](#) instead of this method if you wish to find one word at a time instead of obtaining a table containing all visible words on a page.

This procedure is intended to replace the call to [PDWordFinderAcquireWordList](#) in most cases where you want to work only with the content that is visible on screen (such as a text selection). Change this call to update an application to work with the Optional Content feature.

Parameters

wObj	The word finder (created using PDDocCreateWordFinder or PDDocCreateWordFinderUCS) used to acquire the word list.
pgNum	The page number for which words are found. First page is 0, not 1 as designated in Acrobat.
ocContext	The context within which the words are in a visible state. NULL is equivalent to passing PDDocGetOCContext (pdDoc) .

wInfoP	(Filled by the method) User-supplied PDWord variable. Acrobat will fill this in to point to an Acrobat-allocated array of PDWords, which should NEVER be accessed directly. Access the acquired list through PDWordFinderGetNthWord. The words are ordered in PDF order; that is, the order in which they appear in the PDF file's data. This is often, but not always, the order in which a person would read the words. Use PDWordFinderGetNthWord to traverse this array—you cannot access this array directly. This array is always filled, regardless of the flags used in the call to PDDocCreateWordFinder or PDDocCreateWordFinderUCS.
pXYSortTable	(Filled by the method) Acrobat fills in this user-supplied pointer to a pointer with the location of an Acrobat-allocated array of PDWords, sorted in x-y order—that is, all words on the first “line,” from left to right, followed by all words on the next “line.” This array is only filled if the WXE_XY_SORT flag was set in the call to PDDocCreateWordFinder or PDDocCreateWordFinderUCS. PDWordFinderReleaseWordList MUST be called to release allocated memory for this return or there will be a memory leak. As long as this parameter is non-NULL, the array is always filled regardless of the value of the rdFlags parameter in PDDocCreateWordFinder.
rdOrderTable	Currently unused. Pass NULL .
numWords	(Filled by the method) The number of visible words found on the page.

Return Value

None

Known Exceptions[pdErrOpNotPermitted](#)**Header File**

PDCalls.h

Related Methods

[PDDocCreateWordFinder](#)
[PDDocCreateWordFinderUCS](#)
[PDWordFinderAcquireWordList](#)
[PDWordFinderReleaseWordList](#)
[PDWordFinderEnumWords](#)
[PDWordFinderGetNthWord](#)

Example

```

#define MAX_LENGTH_WORD 128
#define BUFFER_SIZE 80

/* Create a PDWordFinder object */
PDWordFinder pdWF =
    PDDocCreateWordFinder(pdDoc, NULL,
        NULL, NULL, 0, WXE_RD_ORDER_SORT, NULL);
PDPage thePage;
ASInt32 pgNum = 2; /* page 3 */
thePage = PDDocAcquirePage(pdDoc, pgNum);

/* Acquire a word list */
ASInt32 numWords;
PDWord wordInfo;
PDWord *pXYSortTable;
PDWordFinderAcquireVisibleWordList(pdWF, pgNum, NULL,
    &wordInfo, &pXYSortTable, NULL, &numWords);
if(numWords == 0){
    PDPageRelease(thePage);
    PDWordFinderReleaseWordList(pdWF, pgNum);
    PDWordFinderDestroy(pdWF);
    return;
}
/* Search the acquired list for a text string */
char mbuf[BUFFER_SIZE];
char *title = "Adobe";
PDStyle style;
PDFont font;
ASBool IsBold;
PDWord myWord;
ASFixedQuad quad;
ASFixedRect destRect;
PDWord *pdwPtr;
char subtitle[MAX_LENGTH_WORD];
if(pXYSortTable){
    for(pdwPtr = pXYSortTable; numWords; numWords--){
        myWord = *pdwPtr++;
        PDWordGetNthQuad(myWord, 0, &quad);
        style = PDWordGetNthCharStyle(pdWF,
            myWord, 0);
        font = PDStyleGetFont(style);
        PDFontGetName(font, mbuf,
            sizeof(mbuf));
        if(strstr(mbuf, "Bold"))
            IsBold = TRUE;
        else
            IsBold = FALSE;
        if(IsBold){
            PDWordGetString(myWord, subtitle,

```



```
        sizeof(subtitle));
    if (strstr(title, subtitle)){
        QuadToRect(&quad, &destRect);
        break;
    }
}
}
PDWordFinderReleaseWordList(pdWF, pgNum);
}
newview = PDViewDestCreate(pdDoc, thePage,
    k_XYZ, &destRect, fixedZero, pgNum);
PDWordFinderDestroy(pdWF);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDWordFinderAcquireWordList

```
void PDWordFinderAcquireWordList (PDWordFinder wObj,
ASInt32 pgNum, PDWord* pPDWordArray, PDWord** pXYSortTable,
PDWord** rdOrderTable, ASInt32* numWords);
```

Description

Finds all words on the specified page and returns one or more tables containing the words. One table contains the words sorted in the order in which they appear in the PDF file, while the other contains the words sorted by their x- and y-coordinates on the page.

Only words within or partially within the page's crop box (see [PDPageGetCropBox](#)) are enumerated. Words outside the crop box are skipped.

There can be only one word list in existence at a time; clients must release the previous word list, using [PDWordFinderReleaseWordList](#), before creating a new one.

Use [PDWordFinderEnumWords](#) instead of this method if you wish to find one word at a time instead of obtaining a table containing all words on a page.

Parameters

wObj	The word finder (created using PDDocCreateWordFinder or PDDocCreateWordFinderUCS) used to acquire the word list.
pgNum	The page number for which words are found. First page is 0, not 1 as designated in Acrobat.
pPDWordArray	(Filled by the method) User-supplied PDWord variable. Acrobat will fill this in to point to an Acrobat-allocated array of PDWords, which should NEVER be accessed directly. Access the acquired list through PDWordFinderGetNthWord. The words are ordered in PDF order; that is, the order in which they appear in the PDF file's data. This is often, but not always, the order in which a person would read the words. Use PDWordFinderGetNthWord to traverse this array—you cannot access this array directly. This array is always filled, regardless of the flags used in the call to PDDocCreateWordFinder or PDDocCreateWordFinderUCS.

pXYSortTable	<i>(Filled by the method)</i> Acrobat fills in this user-supplied pointer to a pointer with the location of an Acrobat-allocated array of PDWords , sorted in x-y order—that is, all words on the first “line,” from left to right, followed by all words on the next “line.” This array is only filled if the WXE_XY_SORT flag was set in the call to PDDocCreateWordFinder or PDDocCreateWordFinderUCS . PDWordFinderReleaseWordList MUST be called to release allocated memory for this return or there will be a memory leak. As long as this parameter is non- NULL , the array is always filled regardless of the value of the rdFlags parameter in PDDocCreateWordFinder .
rdOrderTable	Currently unused. Pass NULL .
numWords	<i>(Filled by the method)</i> The number of words found on the page.

Return Value

None

Known Exceptions[pdErrOpNotPermitted](#)**Header File**

PDCalls.h

Related Methods[PDDocCreateWordFinder](#)[PDDocCreateWordFinderUCS](#)[PDWordFinderReleaseWordList](#)[PDWordFinderEnumWords](#)[PDWordFinderGetNthWord](#)

Example

```

#define MAX_LENGTH_WORD 128
#define BUFFER_SIZE 80

/* Create a PDWordFinder object */
PDWordFinder pdWF =
    PDDocCreateWordFinder(pdDoc, NULL,
        NULL, NULL, 0, WXE_RD_ORDER_SORT, NULL);
PDPage thePage;
ASInt32 pgNum = 2; /* page 3 */
thePage = PDDocAcquirePage(pdDoc, pgNum);

/* Acquire a word list */
ASInt32 numWords;
PDWord wordInfo;
PDWord *pXYSortTable;
PDWordFinderAcquireWordList(pdWF, pgNum,
    &wordInfo, &pXYSortTable, NULL, &numWords);
if(numWords == 0){
    PDPageRelease(thePage);
    PDWordFinderReleaseWordList(pdWF, pgNum);
    PDWordFinderDestroy(pdWF);
    return;
}
/* Search the acquired list for a text string */
char mbuf[BUFFER_SIZE];
char *title = "Adobe";
PDStyle style;
PDFont font;
ASBool IsBold;
PDWord myWord;
ASFixedQuad quad;
ASFixedRect destRect;
PDWord *pdwPtr;
char subtitle[MAX_LENGTH_WORD];
if(pXYSortTable){
    for(pdwPtr = pXYSortTable; numWords; numWords--){
        myWord = *pdwPtr++;
        PDWordGetNthQuad(myWord, 0, &quad);
        style = PDWordGetNthCharStyle(pdWF,
            myWord, 0);
        font = PDStyleGetFont(style);
        PDFontGetName(font, mbuf,
            sizeof(mbuf));
        if(strstr(mbuf, "Bold"))
            IsBold = TRUE;
        else
            IsBold = FALSE;
        if(IsBold){
            PDWordGetString(myWord, subtitle,

```

```

        sizeof(subtitle));
    if (strstr(title, subtitle)){
        QuadToRect(&quad, &destRect);
        break;
    }
}
}
PDWordFinderReleaseWordList(pdWF, pgNum);
}
newview = PDViewDestCreate(pdDoc, thePage,
    k_XYZ, &destRect, fixedZero, pgNum);
PDWordFinderDestroy(pdWF);

```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDWordFinderDestroy

```
void PDWordFinderDestroy (PDWordFinder wObj);
```

Description

Destroys a word finder. Use this when you are done extracting text in a file.

Parameters

wObj	The word finder to destroy.

Return Value

None

Header File

PDCalls.h

Related Methods

- PDDocCreateWordFinder
- PDDocCreateWordFinderUCS
- PDDocGetWordFinder

Example

```
ACCB1 ASBool ACCB2
WordCounterProc(
    PDWordFinder WordFinder, PDWord word,
    ASInt32 PageNum, void *fileNum)
{
    ASUns32 cur;
    cur = (ASUns32) PDWordGetCharOffset(word);
    WordCnt += (cur - gLast);
    gLast = cur;
}
WordFinder = PDDocCreateWordFinder(
    CurrentPDDoc, NULL, NULL, NULL, 0,
    WXE_PDF_ORDER, NULL);
PDWordFinderEnumWords(WordFinder, PageNum,
    ASCallbackCreateProto(PDWordProc,
        &WordCounterProc), NULL);
PDWordFinderDestroy(WordFinder);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDWordFinderEnumVisibleWords

```
ASBool PDWordFinderEnumVisibleWords (PDWordFinder wObj,
ASInt32 pageNum, PDOCContext ocContext, PDWordProc wordProc,
void* clientData);
```

Description

Extracts visible words, one at a time, from the specified page or the entire document. Calls a user-supplied procedure once for each word found. If you wish to extract all text from a page at once, use [PDWordFinderAcquireWordList](#) instead of this method.

Only words that are visible in the given optional-content context are enumerated.

Parameters

wObj	A word finder object.
pageNum	Page number from which to extract words. Pass PDAllPages (see <code>PDExpT.h</code>) to sequentially process all pages in the document.
ocContext	The context within which the words are in a visible state. NULL is equivalent to passing <code>PDDocGetOCContext (pdDoc)</code> .
wordProc	User-supplied callback to call once for each word found. Enumeration halts if wordProc returns false .
clientData	Pointer to user-supplied data to pass to wordProc each time it is called.

Return Value

Returns **true** if enumeration was successfully completed, **false** if enumeration was terminated because **wordProc** returned **false**.

Known Exceptions

Raises [genErrBadParm](#) if **wordProc** is **NULL**, or **pageNum** is less than zero or greater than the total number of pages in the document.

[pdErrOpNotPermitted](#)

Header File

```
PDCalls.h
```

Related Methods

```
PDDocCreateWordFinder
PDDocCreateWordFinderUCS
PDDocGetWordFinder
PDWordFinderAcquireWordList
PDWordFinderEnumWords
```



PDWordFinderEnumWordsStr

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDWordFinderEnumWords

```
ASBool PDWordFinderEnumWords (PDWordFinder wObj,
ASInt32 pageNum, PDWordProc wordProc, void* clientData);
```

Description

Extracts words, one at a time, from the specified page or the entire document. Calls a user-supplied procedure once for each word found. If you wish to extract all text from a page at once, use [PDWordFinderAcquireWordList](#) instead of this method.

Only words within or partially within the page's crop box (see [PDPageGetCropBox](#)) are enumerated. Words outside the crop box are skipped.

Parameters

wObj	A word finder object.
pageNum	Page number from which to extract words. Pass PDAllPages (see PDExpT.h) to sequentially process all pages in the document.
wordProc	User-supplied callback to call once for each word found. Enumeration halts if wordProc returns false .
clientData	Pointer to user-supplied data to pass to wordProc each time it is called.

Return Value

Returns **true** if enumeration was successfully completed, **false** if enumeration was terminated because **wordProc** returned **false**.

Known Exceptions

Raises [genErrBadParm](#) if **wordProc** is **NULL**, or **pageNum** is less than zero or greater than the total number of pages in the document.

[pdErrOpNotPermitted](#)

Header File

[PDCalls.h](#)

Related Methods

[PDDocCreateWordFinder](#)
[PDDocCreateWordFinderUCS](#)
[PDDocGetWordFinder](#)
[PDWordFinderAcquireWordList](#)
[PDWordFinderEnumVisibleWords](#)
[PDWordFinderEnumWordsStr](#)

Example

```

ACCB1 ASBool ACCB2 WordCounterProc(
    PDWordFinder WordFinder, PDWord word,
    ASInt32 PageNum, void* fileNum)
{
    ASUns32 cur;
    cur = (ASUns32) PDWordGetCharOffset(word);
    WordCnt += (cur - gLast);
    gLast = cur;
}
WordFinder = PDDocCreateWordFinder(CurrentPDDoc, NULL, NULL, NULL, 0,
    WXE_PDF_ORDER, NULL);
PDWordFinderEnumWords(WordFinder, PageNum,
    ASCallbackCreateProto(PDWordProc, &WordCounterProc), NULL);
PDWordFinderDestroy(WordFinder);

```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

PDWordFinderEnumWordsStr

```
ASBool PDWordFinderEnumWordsStr (PDWordFinder wObj,
ASUns16 *ucsStr, ASUns32 strLen, ASUns32 charOffsetAdj,
PDWordProc wordProc, void* clientData);
```

Description

Constructs a **PDWord** list from a Unicode string, and calls a user-supplied procedure once for each word found.

The words extracted by this method do not have quads, text style, or text selection information. The character offset is calculated from the beginning of the input string, and is increased by 2 on every 16-bit data—that is, the character offset of a character in a **PDWord** is the byte offset of the character in the source Unicode string.)

Parameters

wObj	A word finder object.
ucsStr	A pointer to the Unicode string.
strLen	The length of the string in bytes.
charOffsetAdj	The character offset value of the first character in the input Unicode string. This value is added to the word character offsets, and is used to maintain contiguous word character offsets when multiple strings (and multiple calls to this method) are combined into one word list. For example: <pre>PDWordFinderEnumWordsStr(wf, str1, stelen(str1), 0, wp, d); PDWordFinderEnumWordsStr(wf, str2, stelen(str2), stelen(str1), wp, d);</pre>
wordProc	User-supplied callback to call once for each word found. Enumeration halts if wordProc returns false .
clientData	Pointer to user-supplied data to pass to wordProc each time it is called.

Return Value

Returns **true** if enumeration was successfully completed, **false** if enumeration was terminated because **wordProc** returned **false**.

Known Exceptions

Raises **genErrBadParm** if **wordProc** is **NULL**.
pdErrOpNotPermitted

Header File

PDCalls.h

Related Methods

[PDDocCreateWordFinder](#)
[PDDocCreateWordFinderUCS](#)
[PDDocGetWordFinder](#)
[PDWordFinderAcquireWordList](#)
[PDWordFinderEnumVisibleWords](#)
[PDWordFinderEnumWords](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDWordFinderGetLatestAlgVersion

```
ASInt16 PDWordFinderGetLatestAlgVersion (PDWordFinder wObj);
```

Description

Gets the version number of the specified word finder, or the version number of the latest word finder algorithm.

Parameters

wObj	The word finder whose algorithm's version is obtained. Pass NULL to obtain the latest word finding algorithm version number.

Return Value

The algorithm version associated with **wObj**, or the version of the latest word finder algorithm if **wObj** is **NULL**.

Header File

PDCalls.h

Related Methods

[PDDocCreateWordFinder](#)
[PDDocCreateWordFinderUCS](#)
[PDDocGetWordFinder](#)

Example

```
/* store this in our index file so we can
   subsequently highlight text using the
   correct algorithm version */
ASInt16 version =
    PDWordFinderGetLatestAlgVersion(
        WordFinder);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDWordFinderGetNthWord

```
PDWord PDWordFinderGetNthWord (PDWordFinder wObj,
ASInt32 nTh);
```

Description

Gets the n^{th} word in the word list obtained using [PDWordFinderAcquireWordList](#).

Parameters

wObj	The word finder whose n^{th} word is obtained.
nTh	The index of the word to obtain. The first word on a page has an index of zero. Words are counted in PDF order. See the description of the wInfoP parameter in PDWordFinderAcquireWordList .

Return Value

The n^{th} word. Returns **NULL** when the end of the list is reached.

Header File

PDCalls.h

Related Methods

[PDWordFinderAcquireWordList](#)
[PDWordFinderEnumWords](#)

Example

```
PDWord word =
    PDWordFinderGetNthWord (WordFinder, 0);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDWordFinderReleaseWordList

```
void PDWordFinderReleaseWordList (PDWordFinder wObj,  
ASInt32 pageNum);
```

Description

Releases the word list for a given page. Use this to release a list created by [PDWordFinderAcquireWordList](#) when you are done using this list.

Parameters

wObj	A word finder object.
pageNum	Number of pages for which a word list is released.

Return Value

None

Known Exceptions

Raises [genErrBadUnlock](#) if the list has already been released.

Header File

PDCalls.h

Related Methods

[PDWordFinderAcquireWordList](#)
[PDDocCreateWordFinder](#)
[PDDocCreateWordFinderUCS](#)
[PDDocGetWordFinder](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



PDXObject

PDXObjectEnumFilters

```
void PDXObjectEnumFilters (PDXObject obj,
    PDXObjectFilterEnumProc proc, void* clientData);
```

Description

(Obsolete, provided only for backwards compatibility) Enumerates the filters attached to an **XObject**, calling a user-supplied procedure for each filter.

Parameters

obj	The XObject whose filters are enumerated.
proc	User-supplied callback to call for each filter attached to the XObject . Enumeration ends if proc returns false . proc will not be called if there are no filters attached to the XObject .
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Header File

PDCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDXObjectGetCosObj

`CosObj` PDXObjectGetCosObj (`PDXObject` xObj) ;

Description

(Obsolete, provided only for backwards compatibility) Gets the Cos object associated with an **XObject**. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

<code>xObj</code>	The XObject whose Cos object is obtained.

Return Value

The dictionary Cos object for the **XObject**.

Header File

`PDCalls.h`

Product

reader, base, pro, pdf

Availability

Available if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.

PDXObjectGetData

```
void PDXObjectGetData (PDXObject obj,  
    PDGetDataProc getDataProc, void* clientData);
```

Description

(Obsolete, provided only for backwards compatibility) Passes the data from an **XObject** to a user-supplied procedure.

Parameters

obj	The XObject whose data is read.
getDataProc	User-supplied callback to call with the XObject 's data. Enumeration ends if getDataProc returns false .
clientData	Pointer to user-supplied data to pass to getDataProc .

Return Value

None

Header File

PDCalls.h

Related Methods

[PDXObjectGetDataLength](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDXObjectGetDataLength

```
ASInt32 PDXObjectGetDataLength (PDXObject xObj);
```

Description

(Obsolete, provided only for backwards compatibility) Gets the value of the **XObject** stream's **Length** key, which specifies the amount of data in the PDF file (that is, after all compression/encoding filters have been applied).

Parameters

xObj	The XObject whose data length is obtained.

Return Value

The **XObject**'s data length.

Header File

PDCalls.h

Related Methods

[PDXObjectGetData](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDXObjectGetSubtype

```
ASAtom PDXObjectGetSubtype (PDXObject xObj);
```

Description

(Obsolete, provided only for backwards compatibility) Gets the subtype of an **XObject**. Examples of subtype are Image and Form.

Parameters

xObj	The XObject whose subtype is obtained.

Return Value

The **ASAtom** corresponding to the **XObject**'s subtype. Can be converted into a string using [ASAtomGetString](#).

Header File

PDCalls.h

Related Methods

[PDXObjectGetData](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDMODEL_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.





PDFEdit Methods

NOTE: Methods in this layer are not available in Adobe Reader.

Dump

General

PDEBeginContainer

PDEBeginGroup

PDEBeginContainer

PDEClip

PDEColorSpace

PDEContainer

PDEContent

PDEDeviceNColors

PDEElement

PDEEndContainer

PDEEndGroup

PDEExtGState

PDEFont

PDEForm

PDEFormHasXGroup

PDEImage

PDEObject

PDEPath

PDEPattern

PDEPlace

PDEPS

PDESoftMask

PDEText

PDEUnknown

PDEXGroup

PDEXObject

PDSysEncoding



PDFSysFont



Dump

PDELogDump

```
void PDELogDump (PDEObjectDumpProc proc, void* clientData);
```

Description

Enumerates the [PDEObjects](#). This is useful when looking for orphaned objects.

Parameters

proc	Callback to call once for each PDEObject .
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

None

Known Exceptions

- [peErrWrongPDEObjectType](#)
- [peErrUnknownPDEColorSpace](#)
- [genErrBadParm](#)

Header File

PERCalls.h

Related Methods

[PDEObjectDump](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEObjectDump

```
void PDEObjectDump (PDEObject obj, ASInt32 levels,
PDEObjectDumpProc proc, void* clientData);
```

Description

The object, its children and attributes are dumped. The dump contains information about each individual object. The output for child elements is indented with respect to their parents.

- The information for each object is char* — The string describing Object Type. (See [PDEObjectGetType](#).)
- The number representing Object Type. (See [PEExpT.h](#) [PDEType](#) enum.)
- The object reference count.
- The memory location for the object.

Parameters

obj	The PDEObject to dump.
levels	Depth of children to dump.
proc	Callback with the dump information; it may be called more than once per object.
clientData	Provided by the caller as the parameter of the same name for proc .

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

[PERCalls.h](#)

Related Methods

[PDELogDump](#)

Product

base, pro, pdf

Availability

Available if [PI_PDFEDIT_READ_VERSION](#) (in [PIRequir.h](#)) is set to [0x00040000](#) or higher.

PDEAttrEnumTable

```
void PDEAttrEnumTable (PDEAttrEnumProc enumProc,  
void* clientData);
```

Description

Enumerates the table of attributes. This method enumerates the shared resource objects. It is useful when looking for orphaned attributes.

Parameters

enumProc	Callback to call for each attribute.
clientData	Pointer to user-supplied data to pass to enumProc each time it is called.

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

General

PDEDefaultGState

```
void PDEDefaultGState (PDEGraphicStateP *stateP,  
ASInt32 stateSize);
```

Description

Fills out a [PDEGraphicStateP](#) structure with the default graphic state.

NOTE: Non-**NULL** objects in the graphic state, such as the fill and stroke color spaces, have their reference counts incremented by this method. Be sure to release these non-**NULL** objects when disposing of **stateP**.

Parameters

stateP	(Filled by the method) Pointer to a PDEGraphicStateP structure with the default graphic state.
stateSize	Size of the stateP structure, in bytes.

Return Value

None

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEEnumElements

```
void PDEEnumElements (const CosObj* contents,
const CosObj* resources, ASInt32 flags,
PDEElementEnumProc enumProc, void* enumProcClientData);
```

Description

Enumerates all the **PDEElements** in a given stream. Similar to **PDEContentCreateFromCosObj**, but provides enumeration instead of a list of elements.

If marked content is not ignored, each **PDEContainer** contains a **PDEContent** list within itself.

Parameters

contents	Cos object that is the source for the content stream. May be page contents, a Form XObject , a Type 3 font CharProc, or an appearance object from an annotation.
resources	The object's Resources dictionary. If the Form or Type 3 font or appearance dictionary contains a Resources dictionary, this dictionary must be passed in resources . Otherwise, it must be the page resources object of the page containing the Form or Type 3 font contents object.
flags	Flags from PDEEnumElementsFlags .
enumProc	User-supplied callback to call once for each top-level element. NOTE: The element in the enumProc may only be valid for this method. Use PDEAcquire if you need to hold on to the element longer than the scope of enumProc.
enumProcClientData	Pointer to user-supplied data to pass to enumProc each time it is called.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm
peErrPStackUnderflow
peErrCantGetImageDict

**Header File**

PERCalls.h

Related Methods

[PDEContentCreateFromCosObj](#)

[PDEContentGetNumElems](#)

[PDEContentGetElem](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEMergeResourcesDict

```
void PDEMergeResourcesDict (CosObj* resDictP, CosDoc cosDoc,
                             const CosObj* newResP);
```

Description

Merges two Resources dictionaries in the *same* **CosDoc**—you *cannot* merge two resource dictionaries from different **CosDocs**.

Both dictionaries and what they reference must be in **cosDoc**. The objects referenced by **newResP** are appended to **resDictP**.

This method only operates on the Cos dictionaries. It assumes there are no resource name conflicts.

NOTE: Since PDFEdit resolves resource names across **PDEContent** objects, this routine is safe for using with PDFEdit methods. This method may be unsafe if you modify streams and dictionaries outside of the PDFEdit API.

This method is useful for adding form resources to page resource dictionaries. This is only necessary if creating PDF 1.1 or earlier files for use with Acrobat 2.1 or earlier. This is unnecessary if creating PDF 1.2 or later documents.

Parameters

resDictP	(Filled by the method) Dictionary to which newResP is merged. When the method completes, resDictP is the merged dictionary result.
cosDoc	CosDoc containing both dictionaries.
newResP	Dictionary to merge with resDictP .

Return Value

None

Known Exceptions

[genErrBadParm](#)

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEPurgeCache

```
void PDEPurgeCache (PDDoc doc);
```

Description

Clears the PDE Cache of this **PDDoc**. This method is only of interest to clients.

NOTE: We do not recommend calling this method directly; it is provided only for backwards compatibility.

Parameters

doc	A PDDoc whose cache is purged.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEBeginContainer

PDEBeginContainerCreate

```
PDEBeginContainer PDEBeginContainerCreate (ASAtom atom,
CosObj* cosObjP, ASBool isInline);
```

Description

Creates a new **PDEBeginContainer** object.

Parameters

atom	The tag name for the marked-content sequence.
cosObjP	(May be NULL) A CosDict object containing the property list for the sequence.
isInline	If true the object will be emitted into the page content stream in-line.

Return Value

The newly created object.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEBeginContainerGetDict

```
ASBool PDEBeginContainerGetDict
(PDEBeginContainer pdeBeginContainer, CosObj* dictP,
ASBool* isInlineP);
```

Description

Gets the property list dictionary associated with a **PDEBeginContainer** object. The property list is stored in a Cos dictionary.

NOTE: Either **dictP** or **isInlineP** may be **NULL** if that information is not required.

Parameters

pdeBeginContainer	A PDEBeginContainer object.
dictP	(Filled by the method) The property list associated with the PDEBeginContainer .
isInlineP	(Filled by the method) If true the dictionary is emitted into the page content stream inline.

Return Value

true if **dictP** points to a Cos dictionary; **false** otherwise.

Known Exceptions

peErrWrongPDEObjectType if **pdeBeginContainer** is null or not the right type.

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEBeginContainerGetMCTag

```
ASAtom PDEBeginContainerGetMCTag
(PDEBeginContainer pdeBeginContainer);
```

Description

Gets the marked content tag associated with a **PDEBeginContainer** object.

Parameters

pdeBeginContainer A **PDEBeginContainer** object.

Return Value

The mark content tag.

Known Exceptions

peErrWrongPDEObjectType if **pdeBeginContainer** is null or not the right type.

Header File

PERCalls.h

Related Methods

PDEBeginContainerSetMCTag

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEBeginContainerSetDict

```
void PDEBeginContainerSetDict
(PDEBeginContainer pdeBeginContainer, CosObj* cosObjP,
ASBool isInline);
```

Description

Sets the property list for a **PDEBeginContainer**. The property list is passed as a Cos dictionary that can be emitted inline or referenced from the **\Properties** key in the **\Resources** dictionary of the containing page.

NOTE: If **cosObjP** is **NULL**, the property list is cleared.

Parameters

pdeBeginContainer	The PDEBeginContainer object.
cosObjP	(May be NULL) The Cos dictionary containing the property list.
isInline	If true the dictionary will be emitted into the page content stream inline.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#) if **pdeBeginContainer** is null or not the right type.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEBeginContainerSetMCTag

```
void PDEBeginContainerSetMCTag
(PDEBeginContainer pdeBeginContainer, ASAtom atom);
```

Description

Sets the marked content tag for a **PDEBeginContainer**.

Parameters

pdeBeginContainer	The PDEBeginContainer object.
atom	The tag name.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#) if **pdeBeginContainer** is **NULL** or not the right type.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEBeginGroup

PDEBeginGroupCreate

```
PDEBeginGroup PDEBeginGroupCreate ();
```

Description

Creates a new begin group object.

Parameters

None

Return Value

The newly created object.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



PDEClip

PDEClipAddElem

```
void PDEClipAddElem (PDEClip clip, ASInt32 addAfterIndex,
PDEElement pdeElement);
```

Description

Adds an element to a clip path.

Parameters

clip	The clip path to which an element is added.
addAfterIndex	The index after which to add pdeElement . Use kPDEBeforeFirst to insert an element at the beginning of the clip object.
pdeElement	The element added, which may be a PDEPath , a PDEText , a PDEContainer , a PDEGroup , or a PDEPlace object. NOTE: This method increments the reference count of pdeElement .

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PEWCalls.h

Related Methods

[PDEClipRemoveElems](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEClipCopy

```
PDEClip PDEClipCopy (PDEClip srcClip);
```

Description

Makes a deep copy of a **PDEClip** object.

Call **PDERelease** to dispose of the returned clip object when finished with it.

Parameters

srcClip	The clipping path to copy.

Return Value

The deep copy of **srcClip**.

Known Exceptions

Raises if unable to allocate memory.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEClipCreate

```
PDEClip PDEClipCreate (void);
```

Description

Creates an empty clip object. This represents a clipping object that has no effect on elements that refer to it.

Call [PDERelease](#) to dispose of the returned clip object when finished with it.

Parameters

None

Return Value

The newly created clip object.

Notifications

Raises if unable to allocate memory.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEClipFlattenedEnumElems

```
ASBool PDEClipFlattenedEnumElems (PDEClip clip,
PDEClipEnumProc enumProc, void* enumProcClientData);
```

Description

For a given **PDEClip**, enumerates all of the **PDEElements** in a flattened manner. In other words, **PDEContainers** and **PDEGroups** nested in the **PDEClip** will not be handed back, but any **PDEPaths** and **PDETexts** nested in them will be. Additionally, **PDEPlace** objects inside the **PDEClip** are not returned.

Parameters

clip	The PDEClip to enumerate.
enumProc	Called with each flattened element. Enumeration continues until all elements have been enumerated, or until enumProc returns false .
enumProcClientData	Pointer to user-supplied data to pass to enumProc each time it is called.

Return Value

Returns value of **enumProc**. **true** if successful, **false** otherwise.

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm

Header File

PERCalls.h

Related Methods

PDEClipCreate
PDEClipGetElem
PDEClipGetNumElems

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEClipGetElem

```
PDEElement PDEClipGetElem (PDEClip clip, ASInt32 index);
```

Description

Gets an element from a clip object.

NOTE: This method does not change the reference count of the returned **PDEElement**.

Parameters

clip	The clip object from which an element is obtained.
index	Index of element to get from clip .

Return Value

The element from the clip object.

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PERCalls.h

Related Methods

[PDEClipGetNumElems](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEClipGetNumElems

```
ASInt32 PDEClipGetNumElems (PDEClip clip);
```

Description

Gets the number of top-level elements in a clip object. Top-level elements may be a path or char-path, a marked content container or place, or a group.

Paths are represented as **PDEPath** objects; char-paths are represented as **PDEText** objects.

Parameters

clip	The clip object to examine.

Return Value

Number of path and charpath elements in **clip**. If **clip** contains **PDEGroups**, this method returns the top-level **PDEPath**, **PDEText**, **PDEContainer**, **PDEGroup**, or **PDEPlace** object. Use **PDEClipFlattenedEnumElems** to see only the **PDEPath** and **PDEText** objects.

NOTE: **PDEGroup** is not a persistent object. You cannot save to PDF and re-get group objects.

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm

Header File

PERCalls.h

Related Methods

PDEClipFlattenedEnumElems
PDEClipGetElem

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEClipRemoveElems

```
void PDEClipRemoveElems (PDEClip clip, ASInt32 index,  
ASInt32 count);
```

Description

Removes one or more elements from a clip object.

NOTE: This method decrements the reference count of each of the elements.

Parameters

clip	The clip object from which an element is removed.
index	First element to remove.
count	Number of elements to remove.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PEWCalls.h

Related Methods

[PDEClipAddElem](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEColorSpace

PDEColorSpaceCreate

```
PDEColorSpace PDEColorSpaceCreate (ASAtom family,
PDEColorSpaceStruct* csStruct);
```

Description

Creates a new color space object of the specified type.

Call [PDERelease](#) to dispose of the returned color space object when finished with it.

Parameters

family	Supports all PDF 1.3 color spaces, which include: Device-dependent names: DeviceCMYK , DeviceGray , DeviceN , or DeviceRGB . Device-independent names: CalGray , CalRGB , Lab , or ICCBased . Special names: Indexed , Pattern , or Separation .
csStruct	Data for the type of color space you want to create.

Return Value

The newly created color space object.

Known Exceptions

[cosErrExpectedArray](#)
[genErrBadParm](#)
[peErrUnknownPDEColorSpace](#)

Header File

PEWCalls.h

Related Methods

[PDEColorSpaceCreateFromCosObj](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEColorSpaceCreateFromCosObj

```
PDEColorSpace PDEColorSpaceCreateFromCosObj
(const CosObj* cosObjP);
```

Description

Creates a new color space object from a Cos object.

Call [PDERelease](#) to dispose of the returned color space object when finished with it.

Parameters

cosObjP	Supports all PDF 1.3 color spaces, which include: Device-dependent names: DeviceCMYK , DeviceGray , DeviceN , or DeviceRGB . Device-independent names: CalGray , CalRGB , Lab , or ICCBased . Special names: Indexed , Pattern , or Separation .

Return Value

The newly created color space object.

Known Exceptions

[cosErrExpectedArray](#)
[genErrBadParm](#)
[peErrUnknownPDEColorSpace](#)

Header File

PEWCalls.h

Related Methods

[PDEColorSpaceCreate](#)
[PDEColorSpaceCreateFromName](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEColorSpaceCreateFromName

```
PDEColorSpace PDEColorSpaceCreateFromName (ASAtom name);
```

Description

Creates a new color space object.

Call [PDERelease](#) to dispose of the returned color space object when finished with it.

Parameters

name	The ASAtom for the name of the color space created. The name must be one of the following: DeviceCMYK , DeviceGray , or DeviceRGB .

Return Value

The newly created color space object.

Known Exceptions

[cosErrExpectedName](#)
[genErrBadParm](#)
[peErrUnknownPDEColorSpace](#)

Header File

PEWCalls.h

Related Methods

[PDEColorSpaceCreate](#)
[PDEColorSpaceCreateFromCosObj](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEColorSpaceGetBase

```
ASAtom PDEColorSpaceGetBase (PDEColorSpace colorSpace);
```

Description

Gets the name of the base color space. This is a helper routine for indexed color spaces.

Call this method to obtain the base color space and color values for an uncolored pattern in PDFEdit.

NOTE: The base color values are in the **color** array in the **PDEColorValue** field for stroke and fill of a **PDEGraphicStateP**. Or, they are in the **colorObj2** object if the base color space is **DeviceN**. To get the color values, a client gets the base color space, determines the type and number of components of the value, and looks them up in the **PDEColorValue** field.

Parameters

colorSpace	The base color space.

Return Value

The **ASAtom** for the name of the base color space. Use **ASAtomGetString** to obtain a C string for the **ASAtom**.

Known Exceptions

peErrUnknownPDEColorSpace
peErrWrongPDEObjectType

Header File

PERCalls.h

Related Methods

PDEColorSpaceGetBaseNumComps

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEColorSpaceGetBaseNumComps

```
ASInt32 PDEColorSpaceGetBaseNumComps
(PDEColorSpace colorSpace);
```

Description

Gets the number of components in the base color space of an indexed color space.

For example, for **[/Indexed /DeviceRGB...]**, the number of components is 3.

Parameters

colorSpace	The indexed color space.

Return Value

Number of components in **colorSpace**.

Known Exceptions

[peErrUnknownPDEColorSpace](#)
[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Related Methods

[PDEColorSpaceGetBase](#)
[PDEColorSpaceGetNumComps](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEColorSpaceGetCosObj

```
void PDEColorSpaceGetCosObj (PDEColorSpace colorSpace,
                             CosObj* cosObjP);
```

Description

Gets the color space object for an image.

Use only for images. For image masks, use [PDEElementGetGState](#) to obtain color information.

NOTE: (For the Adobe PDF Library v1 only) If you get the [PDEColorSpace](#) for an inline image, then get the [CosObj](#) for that color space with [PDEColorSpaceGetCosObj](#), this [CosObj](#) is limited. Cos objects that are the result of parsing inline dictionaries in the PDF page contents are a special class of Cos objects. You should never depend on these objects lasting the lifetime of the document. You should extract the information you need from the object immediately and refer to it no further in your code.

Parameters

colorSpace	The color space whose Cos object is obtained.
cosObjP	<i>(Filled by the method)</i> Cos object for the color space.

Return Value

Cos object for **colorSpace**. Any color space that is in the Resources dictionary of the page is returned as a Cos object.

NOTE: This Cos object is subject to the warning above.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEColorSpaceGetCTable

```
void PDEColorSpaceGetCTable (PDEColorSpace colorSpace,
ASUns8* colorTableP);
```

Description

Gets the component information for an indexed color space.

Parameters

colorSpace	The color space whose component information table is obtained.
colorTableP	<i>(Filled by the method)</i> The color lookup table, which is numComps * (hiVal + 1) bytes long, where numComps = number of components in the base colorSpace . Each entry in the table contains numComps bytes, and the table is indexed 0 to hiVal , where hiVal is the highest index in the color table. The table is indexed from 0 to hival , thus the table contains hival + 1 entries.

Return Value

None

Known Exceptions

```
peErrUnknownPDEColorSpace
peErrWrongPDEObjectType
```

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEColorSpaceGetHiVal

```
ASInt32 PDEColorSpaceGetHiVal (PDEColorSpace colorSpace);
```

Description

Gets the highest index for the color lookup table for an indexed color space. Since the color table is indexed from zero to **hiVal**, the actual number of entries is **hiVal** + 1.

Parameters

colorSpace	An indexed color space.

Return Value

The highest index (**hiVal**) in the color lookup table.

Known Exceptions

[peErrUnknownPDEColorSpace](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEColorSpaceGetName

```
ASAtom PDEColorSpaceGetName (PDEColorSpace colorSpace);
```

Description

Gets the name of a color space object.

Parameters

colorSpace	A color space object.

Return Value

The color space object's name. Supports all PDF 1.3 color spaces, which include:
 Device-dependent names: **DeviceCMYK**, **DeviceGray**, **DeviceN**, or **DeviceRGB**.
 Device-independent names: **CalGray**, **CalRGB**, **Lab**, or **ICCBased**.
 Special names: **Indexed**, **Pattern**, or **Separation**.

Known Exceptions

[peErrUnknownPDEColorSpace](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEColorSpaceGetNumComps

```
ASInt32 PDEColorSpaceGetNumComps (PDEColorSpace colorSpace);
```

Description

Calculates the number of components in a color space.

Parameters

colorSpace	A color space object.

Return Value

Number of components in **colorSpace**. For:

DeviceGray, CalGray, Separation: Returns 1.

DeviceRGB, CalRGB: Returns 3.

DeviceCMYK, Lab: Returns 4.

DeviceN, ICCBased: Returns the number of components dependent on the specific color space object.

Indexed: Returns 1. Call [PDEColorSpaceGetBaseNumComps](#) to get the number of components in the base color space.

Known Exceptions

[peErrUnknownPDEColorSpace](#)

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Related Methods

[PDEColorSpaceGetBaseNumComps](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEContainer

PDEContainerCreate

```
PDEContainer PDEContainerCreate (ASAtom mcTag,
CosObj* cosObjP, ASBool isInline);
```

Description

Creates a container object.

Call [PDERelease](#) to dispose of the returned container object when finished with it.

Parameters

mcTag	Tag name for the container.
cosObjP	Optional Marked Content dictionary for the container.
isInline	If true , emits container into the page content stream inline.

Return Value

The newly created container object.

Known Exceptions

[pdErrOpNotPermitted](#)

Header File

PEWCalls.h

Related Methods

[PDEContainerGetMCTag](#)
[PDEContainerSetMCTag](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in [PIRequir.h](#)) is set to **0x00040000** or higher.



PDEContainerGetContent

```
PDEContent PDEContainerGetContent (PDEContainer pdeContainer);
```

Description

Gets the **PDEContent** for a **PDEContainer**.

NOTE: This method does not change the reference count of the returned **PDEContent**.

Parameters

pdeContainer	The container whose content is obtained.

Return Value

The **PDEContent** for the **pdeContainer**.

Known Exceptions

[pdErrOpNotPermitted](#)
[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Related Methods

[PDEContainerSetContent](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEContainerGetDict

```
ASBool PDEContainerGetDict (PDEContainer pdeContainer,
                             CosObj* placeDictP, ASBool* isInline);
```

Description

Gets the Marked Content dictionary for a container .

Parameters

pdeContainer	A container.
placeDictP	(Filled by the method) Marked Content dictionary for pdeContainer. NULL if pdeContainer has no Marked Content dictionary.
isInline	(Filled by the method) true if the dictionary is inline, false otherwise. Undefined if pdeContainer has no Marked Content dictionary.

Return Value

true if pdeContainer has a Marked Content dictionary, false otherwise.

Known Exceptions

```
peErrWrongPDEObjectType
cosErrInvalidObj
```

Header File

PERCalls.h

Related Methods

```
PDEContainerSetDict
```

Product

base, pro, pdf

Availability

Available if PI_PDFEDIT_READ_VERSION (in PIRequir.h) is set to 0x00040000 or higher.

PDEContainerGetMCTag

```
ASAtom PDEContainerGetMCTag (PDEContainer pdeContainer);
```

Description

Gets the Marked Content tag for a container.

Parameters

pdeContainer	A container.

Return Value

Marked Content tag of **pdeContainer**. Returns **ASAtomNull** if **pdeContainer** has no Marked Content tag.

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm

Header File

PERCalls.h

Related Methods

PDEContainerCreate
PDEContainerSetMCTag

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEContainerGetXAPMetadata

ASBool PDEContainerGetXAPMetadata (**PDEContainer** pdeContainer, **ASText*** metadataASText);

Description

Gets the XMP metadata associated with a **PDEContainer**.

NOTE: The term “XAP” refers to an early internal code name for Adobe’s Extensible Metadata Platform (XMP).

If there is XMP metadata, it is returned as an **ASText** in the output parameter **metadataASText**. The **ASText** returned becomes the property of the client, who is free to alter or destroy it.

Parameters

pdeContainer	The container whose metadata is retrieved.
metadataASText	<i>(Filled by the method)</i> If there is XMP metadata, it is returned as an ASText object in this parameter. The ASText object returned becomes the property of the client, who is free to alter or destroy it.

Return Value

true exactly when the **pdeContainer** has XMP metadata.

Known Exceptions

ErrSysPDModel
pdMetadataErrCouldntCreateMetaXAP
peErrWrongPDEObjectType
cosErrInvalidObj

Header File

PDMetaDataCalls.h

Related Methods

PDEContainerSetXAPMetadata

Product

base, pro, pdf

Availability

Available if **PI_PDMETADATA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEContainerSetContent

```
void PDEContainerSetContent (PDEContainer pdeContainer,  
PDEContent pdeContent);
```

Description

Sets the content for a container. The existing **PDEContent** is released by this method.

NOTE: This method decrements the reference count of the previous content of the container and increments the reference count of the new **PDEContent**.

Parameters

pdeContainer	A container.
pdeContent	The content of pdeContainer .

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PEWCalls.h

Related Methods

[PDEContainerGetContent](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEContainerSetDict

```
void PDEContainerSetDict (PDEContainer pdeContainer,
CosObj* placeDictP, ASBool isInline);
```

Description

Changes the Marked Content dictionary for a container.

Parameters

pdeContainer	The container whose dictionary is changed.
placeDictP	Marked Content dictionary being set into pdeContainer.
isInline	If true, the dictionary is emitted inline.

Return Value

None

Known Exceptions

```
peErrWrongPDEObjectType
genErrBadParm
```

Header File

PEWCalls.h

Related Methods

```
PDEContainerGetDict
```

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEContainerSetMCTag

```
void PDEContainerSetMCTag (PDEContainer pdeContainer,
    ASAtom mcTag) ;
```

Description

Sets the Marked Content tag for a **PDEContainer**.

Parameters

pdeContainer	The container to tag.
mcTag	Marked Content tag.

Return Value

None

Known Exceptions

```
peErrWrongPDEObjectType
genErrBadParm
```

Header File

PEWCalls.h

Related Methods

```
PDEContainerCreate
PDEContainerGetMCTag
```

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEContainerSetXAPMetadata

```
void PDEContainerSetXAPMetadata (PDEContainer pdeContainer,
PDDoc pdDoc, ASText metadataASText);
```

Description

Sets the XMP metadata associated with a **PDEContainer**.

NOTE: The term “XAP” refers to an early internal code name for Adobe’s Extensible Metadata Platform (XMP).

Replaces the XMP metadata associated with **pdeContainer** with the XMP metadata stored in **metadataASText**. The contents of **metadataASText** must be well-formed XML and Resource Description Format (RDF) as defined by the W3C (see <http://www.w3.org/RDF>) that also forms valid XMP. **PDEContainerSetXAPMetadata** will not destroy **metadataASText** or alter its text.

Parameters

pdeContainer	The container whose metadata is set.
pdDoc	The document containing pdeContainer .
metadataASText	Well-formed XML and RDF that also forms valid XMP.

Return Value

None

Known Exceptions

ErrSysPDModel
peErrWrongPDEObjectType
cosErrInvalidObj

Header File

PDMetaDataCalls.h

Related Methods

PDEContainerGetXAPMetadata

Product

base, pro, pdf

Availability

Available if **PI_PDMETADATA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEContent

PDEContentAddElem

```
void PDEContentAddElem (PDEContent pdeContent,
                        ASInt32 addAfterIndex, PDEElement pdeElement);
```

Description

Inserts an element into a **PDEContent**.

NOTE: This method increments the reference count of **pdeElement**.

Parameters

pdeContent	The content to which pdeElement is added.
addAfterIndex	Location after which pdeElement is added. Should be kPDEBeforeFirst to add to the beginning of the display list.
pdeElement	The element to add to pdeContent . The reference count of pdeElement is incremented.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PEWCalls.h

Related Methods

[PDEContentRemoveElem](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

Header File

PEWCalls.h

**Related Methods**

[PDEContentRemoveElem](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.



PDEContentCreate

```
PDEContent PDEContentCreate (void);
```

Description

Creates an empty content object.

NOTE: Do not use this method to create a **PDEContent** to be put into a **PDPage**. Instead, call **PDPageAcquirePDEContent**.

Call **PDERelease** to dispose of the returned content object when finished with it.

Parameters

None

Return Value

An empty content object.

Known Exceptions

peErrPStackUnderflow

Header File

PEWCalls.h

Related Methods

PDEContentCreateFromCosObj

PDPageAcquirePDEContent

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEContentCreateFromCosObj

PDEContent PDEContentCreateFromCosObj (const **CosObj*** contents, const **CosObj*** resources);

Description

Creates a content object from a Cos object. This is the main method for obtaining a **PDEContent**.

Call **PDERelease** to dispose of the returned content object when finished with it.

Parameters

contents	Cos object that is the source for the content. May be a page contents, a Form XObject , a Type 3 font CharProc, or an appearance dictionary for an annotation.
resources	The object's Resources dictionary. If the Form or Type 3 font or appearance dictionary contains a Resources dictionary, this dictionary must be passed in resources . Otherwise, it must be the page resources object of the page containing the Form or Type 3 font contents object.

Return Value

The content from the Cos object.

Known Exceptions

pdErrOpNotPermitted
peErrPStackUnderflow

Header File

PERCalls.h

Related Methods

PDEContentCreate
PDEContentToCosObj

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEContentFlattenOC

`ASBool` PDEContentFlattenOC (`PDEContent` content,
`PDOCContext` context)

Description

Flattens the content, removing any **PDEElements** that are not visible in the given optional-content context, and removing the optional-content information from any visible **PDEElements**.

Parameters

<code>content</code>	The content to be modified.
<code>context</code>	The optional-content context in which content is checked for visibility.

Return Value

true if the operation is successful, **false** otherwise.

Header File

`PDCalls.h`

Related Methods

[PDDocFlattenOC](#)
[PDPageFlattenOC](#)

PDEContentGetAttrs

```
void PDEContentGetAttrs (PDEContent pdeContent,  
PDEContentAttrsP attrsP, ASInt32 attrsSize);
```

Description

Gets the attributes of a content.

Parameters

pdeContent	A content object.
attrsP	(Filled by the method) Pointer to a PDEContentAttrs structure containing the attributes of the content.
attrsSize	Size of the attrsP buffer, in bytes.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEContentGetDefaultColorSpace

```
PDEColorSpace PDEContentGetDefaultColorSpace
(PDEContent pdeContent, ASAtom colorSpaceName);
```

Description

Gets a default color space from a **PDEContent**.
See Section 4.5.4 in the *PDF Reference* for more information about default color spaces.

NOTE: This method does not change the reference count of the returned **PDEColorSpace**.

Parameters

pdeContent	A content object.
colorSpaceName	An ASAtom for the name of the desired color space. Must be an ASAtom for one of DefaultRGB , DefaultCMYK , or DefaultGray .

Return Value

The desired color space in **pdeContent**. Returns **NULL** if **colorSpaceName** does not correspond to a known default, such as **DefaultRGB**.

Header File

PERCalls.h

Related Methods

[PDEContentGetNumElems](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEContentGetElem

```
PDEElement PDEContentGetElem (PDEContent pdeContent,  
ASInt32 index);
```

Description

Gets the requested element from a content.

NOTE: This method does not change the reference count of the element.

NOTE: This method does not copy the element.

Parameters

pdeContent	A content object.
index	Index of element to obtain.

Return Value

The requested element.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Related Methods

[PDEContentGetNumElems](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEContentGetNumElems

```
ASInt32 PDEContentGetNumElems (PDEContent pdeContent) ;
```

Description

Gets the number of elements in a **PDEContent**.

Parameters

pdeContent	A content object.

Return Value

Number of elements in **pdeContent**.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Related Methods

[PDEContentGetElem](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEContentGetResources

```
ASInt32 PDEContentGetResources (PDEContent pdeContent ,
ASInt32 type, PDEObject* resourcesP);
```

Description

Gets the number of resources of the specified type and, optionally, pointers to the resource objects.

Parameters

pdeContent	A content object.
type	Type of resources to query or obtain: PDEFont, PDEXGroup, or PDEColorSpace. Must be one of PDEContentGetResourceFlags.
resourcesP	(Filled by the method) If non-NULL, it must point to an array of PDEObject pointers. On return, the array contains pointers to the requested resources. If resourcesP is NULL, only the number of resources of type is returned. NOTE: The object in resourcesP may only be valid for this method. Use PDEAcquire if you need to hold on to the object longer than the scope of resourcesP

Return Value

Number of resources of type returned in resourcesP.

Known Exceptions

```
peErrWrongPDEObjectType
genErrBadParm
```

Header File

```
PERCalls.h
```

Product

```
base, pro, pdf
```

Availability

Available if PI_PDFEDIT_READ_VERSION (in PIRequir.h) is set to 0x00040000 or higher.

PDEContentRemoveElem

```
void PDEContentRemoveElem (PDEContent pdeContent,  
ASInt32 index);
```

Description

Removes an element from a **PDEContent**.

NOTE: This decrements the reference count of the element removed.

Parameters

pdeContent	A content object.
index	Index in pdeContent of the element to remove, whose reference count is decremented.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PEWCalls.h

Related Methods

[PDEContentAddElem](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEContentToCosObj

```
void PDEContentToCosObj (PDEContent pdeContent, ASInt32 flags,
PDEContentAttrsP attrs, ASInt32 attrsSize, CosDoc cosDoc,
PDEFilterArrayP filtersP, CosObj* contentsP,
CosObj* resourcesP);
```

Description

This is the main method for converting a **PDEContent** into PDF contents and resources.

NOTE: Do not use this method to put a **PDEContent** into a **PDPage**. Instead, call **PDPageSetPDEContent**.

This method does not change the **PDEContent** object nor its reference count.

The caller of this function is responsible for adding the contents and the resources returned from this method to the Page Object.

Parameters

pdeContent	A content object.
flags	Flags specifying the type of object to create (page contents, form, or charproc) and how it is created. Must be one or more of PDEContentToCosObjFlags .
attrs	A pointer to a PDEContentAttrs structure that contains the appropriate form attributes or cache device/char-width attributes, and so on. If zero, no attributes are set.
attrsSize	Size of the attrs buffer, in bytes. Zero if attrs is zero.
cosDoc	Document in which the contents and resources are created.
filtersP	A pointer to a PDEFilterArray structure that specifies which filters to use in encoding the contents; may be NULL . If filtersP contains any encodeParms , they must belong to cosDoc .
contentsP	<i>(Filled by the method)</i> Cos object for the resulting contents in pdeContent .

resourcesP	(Filled by the method) Cos object for the resulting resources in pdeContent .

NOTE: The client is responsible for putting the **resourcesP** dictionary into the **contentsP** stream for non-page objects. The client must do this for **XObject** Forms and appearance dictionaries in annotations. For Type 3 fonts, the resource dictionaries must be merged and put into the Type 3 font dictionary. For a page, the contents and resources must be put into the page object.

Return Value

None

Known Exceptions

[peErrUnknownResType](#)
[pageErrErrorParsingImage](#)
[pdErrBadResMetrics](#)
[peErrWrongPDEObjectType](#)
[peErrUnknownPDEColorSpace](#)

Header File

PEWCalls.h

Related Methods
[PDEContentCreateFromCosObj](#)
Example

```

PDEContentToCosObj( pdeContent, 0, NULL, 0, 0, dP/*CosDoc*/, 0,
&contents, &resources );

/*Get the CosObj for the page */
cosPage = PDPageGetCosObj (pdPage);
CosDictPut(cosPage, ASAtomFromString("Contents"), contents);

CosDictPut(cosPage, ASAtomFromString("Resources"), resources);

```

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEDeviceNColors

PDEDeviceNColorsCreate

```
PDEDeviceNColors PDEDeviceNColorsCreate  
(ASFixed* pColorValues, ASInt32 numValues);
```

Description

Creates an object that can be used to store **n** color components when in a **DeviceN** color space.

Parameters

pColorValues	Pointer to an array of ASFixed values.
numValues	The length of the array.

Return Value

An object containing values specifying a color in a **PDEDeviceNColors** color space.

Known Exceptions

[genErrNoMemory](#)

Header File

PEWCalls.h

Related Methods

[PDEDeviceNColorsGetColorValue](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEDeviceNColorsGetColorValue

ASFixed PDEDeviceNColorsGetColorValue
(**PDEDeviceNColors** colors, **ASInt32** index);

Description

Gets the value of a color component of a **PDEDeviceNColors** color space.

Parameters

colors	A PDEDeviceNColors object returned by PDEDeviceNColorsCreate .
index	Index of the color component to return.

Return Value

The value of the requested color component.

Known Exceptions

genErrBadParm

Header File

PERCalls.h

Related Methods

PDEDeviceNColorsCreate

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEElement

PDEElementCopy

```
PDEElement PDEElementCopy (PDEElement pdeElement,  
ASInt32 flags);
```

Description

Makes a copy of an element. The caller is responsible for releasing the copy with [PDERelease](#).

Parameters

<code>pdeElement</code>	The element to copy.
<code>flags</code>	Bit field of PDEElementCopyFlags .

Return Value

A copy of `pdeElement`.

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

`PEWCalls.h`

Related Methods

[PDEContentGetElem](#)

Product

base, pro, pdf

Availability

Available if `PI_PDFEDIT_WRITE_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.

PDEElementGetAllVisibilities

```
ASInt32 PDElementGetAllVisibilities (PDElement elem,
PDEContent content, PDOCContext ocContext,
ASBool *visibilities, ASInt32 capacity);
```

Description

Tests whether all occurrences of the element are visible in a given content and optional-content context. Traverses the content to find each occurrence of the element, in the supplied content and in all nested contents. To find the visibility of a content element without considering its parent, use [PDElementIsCurrentlyVisible](#).

Returns the number of occurrences and an array of boolean values, **true** for each occurrence of the element that is visible in the context, taking into account the context's **NonOCDrawing** and **PDOCDrawEnumType** values.

Parameters

elem	The element for which to obtain visibilities.
content	The content containing the element.
ocContext	The optional-content context in which the element is tested.
visibilities	<i>(Filled by the method)</i> An array of boolean values, true for each occurrence of the element that is visible in the context.
capacity	The size of the visibilities array.

Return Value

The number of occurrences of the element in the content.

Header File

PERCalls.h

Related Methods

```
PDElementIsCurrentlyVisible
PDElementMakeVisible
```

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDEElementGetBBox

```
void PDEElementGetBBox (PDEElement pdeElement,
ASFixedRectP bboxP);
```

Description

Gets the bounding box for an element.

The returned bounding box is guaranteed to encompass the element, but is *not* guaranteed to be the smallest box that could contain the element. For example, for an arc, **bboxP** encloses the Bezier control points—not just the curve itself.

Parameters

pdeElement	An element whose bounding box is obtained.
bboxP	(Filled by the method) Pointer to a ASFixedRect structure specifying the bounding box of pdeElement , specified in user space coordinates.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm

Notifications

PDEElementGetClip
PDEElementGetGState
PDEElementGetMatrix

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEElementGetClip

```
PDEClip PDEElementGetClip (PDEElement pdeElement);
```

Description

Gets the current clip for an element.

NOTE: This method does not change the reference count of the clip object.

Parameters

pdeElement	An element whose clip is obtained.
NOTE: A clip may be shared by many elements. Use care when modifying a clip. Copy it first if you want to modify the clip for a specific element.	

Return Value

Clip object for **pdeElement**.

Known Exceptions

peErrWrongPDEObjectType

Header File

PERCalls.h

Related Methods

[PDEElementGetBBox](#)
[PDEElementGetGState](#)
[PDEElementGetMatrix](#)
[PDEElementIsAtRect](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEElementGetGState

```
void PDEElementGetGState (PDEElement pdeElement,  
PDEGraphicStateP stateP, ASInt32 stateSize);
```

Description

Gets the graphics state information for an element.

This method is only valid for [PDEForm](#), [PDEImage](#), [PDEPath](#), and [PDEShading](#) elements.

Parameters

pdeElement	An element whose graphics state is obtained.
stateP	<i>(Filled by the method)</i> Pointer to a PDEGraphicStateP structure that contains graphics state information for pdeElement . This PDEGraphicStateP may contain PDEObjects for color spaces or an ExtGState . They are <i>not</i> acquired by this method. NOTE: For a PDEImage , only the ExtGState value is used for images. For indexed images, the fill color space and values are categorized in the PDEImage object.
stateSize	Size of the stateP buffer, in bytes.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PERCalls.h

Related Methods

[PDEElementSetGState](#)
[PDEElementGetBBox](#)
[PDEElementGetClip](#)
[PDEElementGetMatrix](#)

Product

base, pro, pdf

**Availability**

Available if **PI_PDFEDIT_READ_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEElementGetMatrix

```
void PDEElementGetMatrix (PDEElement pdeElement,  
    ASFixedMatrixP matrixP);
```

Description

Gets the transformation matrix for an element.

This matrix provides the transformation from user space to device space for the element. If there is no **cm** operator (**concatmatrix**) in the page stream, the matrix is the identity matrix.

For the Adobe PDF Library v1, the element may *not* be a [PDEContainer](#), a [PDEGroup](#), a [PDEPlace](#), or a [PDEText](#).

For the Adobe PDF Library v4, the element may *not* be a [PDEText](#).

Parameters

pdeElement	An element whose transformation matrix is obtained.
matrixP	(Filled by the method) Pointer to ASFixedMatrix that holds a transformation matrix for pdeElement . If pdeElement is a text object, returns the identity matrix.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PERCalls.h

Related Methods

[PDEElementSetMatrix](#)
[PDEElementGetBBox](#)
[PDEElementGetGState](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEElementGetOCMD

```
PDOCMD PDElementGetOCMD (PDEElement elem);
```

Description

Gets an optional-content membership dictionary (OCMD) object associated with the element. The element must be a **PDEForm**, **PDEImage** (XObject image), or **PDEContainer**. If it is not one of these, the method returns **NULL**.

- If the element is a **PDEForm** or **PDEImage**, the method returns the dictionary attached to the element's Cos **XObject** dictionary.
- If the element is a **PDEContainer**, and it is for optional content, the method returns the dictionary. If it is not for optional content, the method returns **NULL**.

Parameters

elem	The element from which the dictionary is obtained.

Return Value

The dictionary object, or **NULL** if the element is not a **PDEForm**, **PDEImage** (XObject image), or **PDEContainer**, or if it is a container that is not for optional content.

Header File

PERCalls.h

Related Methods

[PDAnnotGetOCMD](#)
[PDEElementSetOCMD](#)
[PDEElementRemoveOCMD](#)
[PDOCMDFindOrCreate](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDEElementHasGState

```
ASBool PDEElementHasGState (PDEElement pdeElement,
                             PDEGraphicStateP stateP, ASInt32 stateSize);
```

Description

Gets the graphics state information for an element.

Parameters

pdeElement	The PDEElement whose graphics state is to be obtained.
stateP	(Filled by the method) Pointer to a PDEGraphicStateP structure that contains graphics state information for pdeElement.
stateSize	The size of the stateP buffer, sizeof (PDEGraphicState)

Return Value

Returns true if the element has a graphics state; false otherwise.

Known Exceptions

genErrBadParm

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if PI_PDFEDIT_READ_VERSION (in PIRequir.h) is set to 0x00050000 or higher.

PDEElementIsAtPoint

```
ASBool PDEElementIsAtPoint (PDEElement elem,
ASFixedPoint point);
```

Description

Tests whether a point is on an element.

Parameters

elem	The element to test. If PDEElement is a PDEText or PDEImage , it uses the bounding box of the PDEElement to make the check. If the PDEElement is a PDEPath and it is stroked, it checks if the point is on the path. If the PDEElement is a PDEPath and it is filled, it checks if the point is in the fill area, taking into consideration whether it is filled using the non-zero winding number rule or the even-odd rule.
point	The point, specified in user space coordinates.

Return Value

true if the point is on the element, **false** otherwise.

Known Exceptions

```
peErrWrongPDEObjectType
```

Header File

```
PERCalls.h
```

Related Methods

```
PDEElementIsAtRect
PDETextIsAtPoint
PDETextIsAtRect
```

Product

```
base, pro, pdf
```

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDElementIsAtRect

```
ASBool PDElementIsAtRect (PDElement elem, ASFixedRect rect);
```

Description

Tests whether any part of a rectangle is on an element.

Parameters

elem	The element to test. If PDElement is a PDEText or PDEImage , it uses the bounding box of the PDElement to make the check. If the PDElement is a PDEPath and it is stroked, it checks if the rectangle is on the path. If the PDElement is a PDEPath and it is filled, it checks if the rectangle is in the fill area, taking into consideration whether it is filled using the non-zero winding number rule or the even-odd rule.
rect	The rectangle, specified in user space coordinates.

Return Value

true if any part of the rectangle is on the element, **false** otherwise.

Known Exceptions

peErrWrongPDEObjectType

Header File

PERCalls.h

Related Methods

PDElementIsAtPoint
PDETextIsAtPoint
PDETextIsAtRect

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDElementIsCurrentlyVisible

```
ASBool PDElementIsCurrentlyVisible (PDElement elem,
PDEContent content, PDOCContext ocContext);
```

Description

Tests whether an element is visible in a given content and optional-content context. Traverses the content to find the first occurrence of the element, in the supplied content and in all nested contents. Returns **true** if the first occurrence of the element is visible in the context, taking into account the context's **NonOCDrawing** and **PDOCDrawEnumType** values.

The content can be **NULL**. In this case:

- If the element is a **PDEForm**, **PDEImage**, or **PDEContainer**, the method checks the object to see if it has an optional-content membership dictionary (OCMD) attached to it. If so, the method returns true if the object is visible, without considering whether the **PDEContent** that the element belongs to is visible.
- If the element is not one of these types, the method returns **true**.

Parameters

elem	The element to test.
content	The content containing the element.
ocContext	The optional-content context in which the element is tested.

Return Value

Returns **true** if the element is visible in the given content and context, **false** if it is hidden.

Header File

PERCalls.h

Related Methods

[PDElementGetAllVisibilities](#)
[PDElementMakeVisible](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDEElementMakeVisible

```
ASBool PDEElementMakeVisible (PDEElement elem,  
                               PDEContent content, PDOCContext ocContext);
```

Description

Makes an element visible in a given content and optional-content context, by manipulating the ON-OFF states of the optional-content groups.

Parameters

elem	The element for which to set the visibility state.
content	The content containing the element.
ocContext	The optional-content context in which the element is made visible.

Return Value

true if the element is successfully made visible in the given content and context, false otherwise.

Header File

PERCalls.h

Related Methods

```
PDEElementGetAllVisibilities  
PDEElementIsCurrentlyVisible  
PDOCMDsMakeContentVisible
```

Product

base, pro, pdf

Availability

Available if PI_PDFEDIT_WRITE_VERSION (in PIRequir.h) is set to 0x00060000 or higher.

PDEElementRemoveOCMD

```
void PDEElementRemoveOCMD (PDEElement elem);
```

Description

Dissociates an optional-content membership dictionary (OCMD) object from the element. The element must be a **PDEForm**, **PDEImage** (XObject image), or **PDEContainer**. If it is not one of these, nothing happens.

- If the element is a **PDEForm** or **PDEImage**, the method removes the dictionary from the element's Cos **XObject** dictionary.
- If the element is a **PDEContainer** for optional content, the method removes the dictionary, but does not destroy the container.

Parameters

elem	The element for which to remove the dictionary.

Return Value

None

Header File

PERCalls.h

Related Methods

[PDEElementGetOCMD](#)
[PDEElementSetOCMD](#)
[PDOCMDFindOrCreate](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDEElementSetClip

```
void PDEElementSetClip (PDEElement pdeElement,  
PDEClip pdeClip);
```

Description

Sets the current clip for an element.

pdeElement's previous clip's reference count is decremented (if it had one), and **pdeClip**'s reference count is incremented.

Parameters

pdeElement	An element whose clip is set.
pdeClip	The clip to set for pdeElement .

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PEWCalls.h

Related Methods

[PDEElementGetClip](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEElementSetGState

```
void PDEElementSetGState (PDEElement pdeElement,  
PDEGraphicStateP stateP, ASInt32 stateSize);
```

Description

Sets the graphics state information for an element.

This method is valid only for [PDEForm](#), [PDEImage](#), [PDEPath](#), and [PDEShading](#) elements.

NOTE: This method causes any of **stateP**'s color space or ExtGState objects to have their reference count incremented and previous graphic state objects to be decremented.

Parameters

pdeElement	An element whose graphics state is set.
stateP	Pointer to a PDEGraphicStateP structure with graphics state information to set for pdeElement . Any of stateP 's color space or ExtGState objects have their reference count incremented.
stateSize	Size of the stateP buffer, in bytes.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)
Will raise a [genErrBadParm](#) if the first parameter, **pdeElement**, does not have a graphics state associated with it.

Header File

PEWCalls.h

Related Methods

[PDEElementGetGState](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEElementSetMatrix

```
void PDEElementSetMatrix (PDEElement pdeElement,
ASFixedMatrixP matrixP);
```

Description

Sets the transformation matrix for an element.

The element may not be a [PDEContainer](#), a [PDEGroup](#), a [PDEPlace](#), or a [PDEText](#).

Parameters

pdeElement	An element whose transformation matrix is set.
matrixP	Pointer to ASFixedMatrix that holds the transformation matrix to set for pdeElement .

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PEWCalls.h

Related Methods

[PDEElementGetMatrix](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEElementSetOCMD

```
void PDEElementSetOCMD (PDEElement elem, PDOCMD pdOCMD);
```

Description

Associates an optional-content membership dictionary (OCMD) object with the element. The element must be a **PDEForm**, **PDEImage** (XObject image), or **PDEContainer**. If it is not one of these, nothing happens.

- If the element is a **PDEForm** or **PDEImage**, the method attaches the dictionary to the element's Cos **XObject** dictionary.
- If the element is a **PDEContainer**, and it is already for optional content, the optional-content information is replaced. If it is not already for optional content, a new **PDEContainer** for optional content is created and nested inside the specified container.

Parameters

elem	The element for which to set the dictionary.
pdOCMD	The new dictionary.

Return Value

None

Header File

PERCalls.h

Related Methods

[PDEElementGetOCMD](#)
[PDEElementRemoveOCMD](#)
[PDOCMDFindOrCreate](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDEndContainer

PDEndContainerCreate

```
PDEndContainer PDEndContainerCreate ();
```

Description

Creates a new end container object.

Parameters

None

Return Value

The newly created object.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



PDEndGroup

PDEndGroupCreate

```
PDEndGroup PDEndGroupCreate ();
```

Description

Creates a new end group object.

Parameters

None

Return Value

The newly created object.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDEExtGState

PDEExtGStateAcquireSoftMask

```
PDESoftMask PDEExtGStateAcquireSoftMask
(PDEExtGState pdeExtGState) ;
```

Description

Acquires the soft mask of the extended graphic state.

Parameters

pdeExtGState	The extended graphics state object.

Return Value

The soft mask or **NULL** if the ExtGState dictionary does not contain the **SMask** key.

Known Exceptions

peErrWrongPDEObjectType if **pdeExtGState** is **NULL** or is not of type **kPDEExtGState**.

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEExtGStateCreate

```
PDEExtGState PDEExtGStateCreate (CosObj* cosObjP);
```

Description

Creates a new **PDEExtGState** from a Cos object. See Section 4.3.4 in the [PDF Reference](#) for more information about extended graphics states.

Call **PDERelease** to dispose of the returned **PDEExtGState** when finished with it.

Parameters

cosObjP	A Cos object for a dictionary of type ExtGState .

Return Value

The **PDEExtGState** for **cosObjP**.

Header File

PEWCalls.h

Related Methods

[PDEElementSetGState](#)

[PDEExtGStateGetCosObj](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEExtGStateCreateNew

```
PDEExtGState PDEExtGStateCreateNew (CosDoc doc);
```

Description

Creates a new extended graphics state object.

Parameters

<code>doc</code>	The document the object will be used within.

Return Value

The newly created object.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDEExtGStateGetAIS

```
ASBool PDEExtGStateGetAIS (PDEExtGState pdeExtGState) ;
```

Description

Returns the value of the Alpha Is Shape (AIS) member of the graphics state. If AIS is **true**, the sources of alpha are treated as shape; otherwise they are treated as opacity values. If the value is not set, the default value of **false** is returned.

Parameters

pdeExtGState	The extended graphics state object.

Return Value

See above.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEExtGStateGetBlendMode

```
ASAtom PDEExtGStateGetBlendMode (PDEExtGState pdeExtGState);
```

Description

Returns the blend mode for the color composite for each object painted. Valid names are **Compatible**, **Normal**, **Multiply**, **Screen**, **Difference**, **Darken**, **Lighten**, **ColorDodge**, **ColorBurn**, **Exclusion**, **HardLight**, **Overlay**, **SoftLight**, **Luminosity**, **Hue**, **Saturation** and **Color**.

Parameters

pdeExtGState	The extended graphics state object.

Return Value

If the value has not been set a value of **Compatible** is returned. See above.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEExtGStateGetCosObj

```
void PDEExtGStateGetCosObj (PDEExtGState extGState,  
CosObj* cosObjP);
```

Description

Gets a Cos object for a PDEExtGState.

Parameters

extGState	A PDEExtGState whose Cos object is obtained.
cosObjP	(Filled by the method) Cos object for extGState.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType

Header File

PERCalls.h

Related Methods

PDEExtGStateCreate

Product

base, pro, pdf

Availability

Available if PI_PDFEDIT_READ_VERSION (in PIRequir.h) is set to 0x00040000 or higher.



PDEExtGStateGetOpacityFill

```
ASFixed PDEExtGStateGetOpacityFill
(PDEExtGState pdeExtGState) ;
```

Description

Returns the opacity value for painting operations other than stroking.

Parameters

pdeExtGState	The extended graphics state object.

Return Value

Returns the value of the **/ca** key in the ExtGState dictionary. If the value is not found, the default value of **1** is returned.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEExtGStateGetOpacityStroke

```
ASFixed PDEExtGStateGetOpacityStroke
(PDEExtGState pdeExtGState) ;
```

Description

Returns the opacity value for stroke painting operations for paths and glyph outlines.

Parameters

<code>pdeExtGState</code>	The extended graphics state object.

Return Value

Returns the value of the **/CA** key in the ExtGState dictionary. If the value is not found, the default value of **1** is returned.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

`PERCalls.h`

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDEExtGStateGetOPFill

```
ASBool PDEExtGStateGetOPFill (PDEExtGState pdeExtGState);
```

Description

Returns whether overprint is enabled for painting operations other than stroking.

Parameters

pdeExtGState	The extended graphics state object.

Return Value

Returns the value of the **/op** key in the ExtGState dictionary. If the value is not found, the default value of false is returned.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEExtGStateGetOPM

```
ASInt32 PDEExtGStateGetOPM (PDEExtGState pdeExtGState);
```

Description

Returns the overprint mode used by this graphics state.

Parameters

pdeExtGState	The extended graphics state object.

Return Value

Cos integer value.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEExtGStateGetOPStroke

```
ASBool PDEExtGStateGetOPStroke (PDEExtGState pdeExtGState) ;
```

Description

Returns whether overprint is enabled for stroke painting operations.

Parameters

pdeExtGState	The extended graphics state object.

Return Value

Returns the value of the **/OP** key in the ExtGState dictionary. If the value is not found, the default value of **false** is returned.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEExtGStateGetSA

```
ASBool PDEExtGStateGetSA (PDEExtGState pdeExtGState) ;
```

Description

Returns whether stroke adjustment is enabled in the graphics state.

Parameters

pdeExtGState	The extended graphics state object.

Return Value

Returns the value of the **/SA** key in the ExtGState dictionary. If the value is not set, the default value of **false** is returned.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEExtGStateGetTK

```
ASBool PDEExtGStateGetTK (PDEExtGState pdeExtGState) ;
```

Description

Returns whether text knockout is enabled in the graphics state.

Parameters

pdeExtGState	The extended graphics state object.

Return Value

Returns the value of the **/TK** key in the ExtGState dictionary. If the value is not found, the default value of **true** is returned.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEExtGStateHasSoftMask

```
ASBool PDEExtGStateHasSoftMask (PDEExtGState pdeExtGState) ;
```

Description

Returns whether the graphics state contains a soft mask.

Parameters

pdeExtGState	The extended graphics state object.

Return Value

Returns **true** if the ExtGState dictionary contains the **/SMask** key; otherwise **false** is returned.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEExtGStateSetAIS

```
void PDEExtGStateSetAIS (PDEExtGState pdeExtGState,
ASBool alphaIsShape);
```

Description

Specifies if the alpha is to be interpreted as a shape or opacity mask.

Parameters

pdeExtGState	The extended graphics state object.
alphaIsShape	Indicates whether the sources of alpha are to be treated as shape (true) or opacity (false). This determines the interpretation of the constant alpha (ca or CA) and soft mask (SMask) parameters of the graphics state, as well as a soft-mask image (Smask entry) of an image XObject.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEExtGStateSetBlendMode

```
void PDEExtGStateSetBlendMode (PDEExtGState pdeExtGState,
ASAtom atom);
```

Description

Sets the blend mode for the color composites for each object painted.

Valid mode names are **Compatible**, **Normal**, **Multiply**, **Screen**, **Difference**, **Darken**, **Lighten**, **ColorDodge**, **ColorBurn**, **Exclusion**, **HardLight**, **Overlay**, **SoftLight**, **Luminosity**, **Hue**, **Saturation** and **Color**.

Parameters

pdeExtGState	The extended graphics state object.
atom	The new blend mode.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEExtGStateSetOpacityFill

```
void PDEExtGStateSetOpacityFill (PDEExtGState pdeExtGState,  
    ASFixed opacity);
```

Description

Sets the opacity value for painting operations other than stroking. The value must be in the range from 0 to 1 inclusive. Corresponds to the **/ca** key within the ExtGState's dictionary. The value from 0 to 1 refers to a float number (not an **ASFixed** value) that should be converted to **ASFixed** using **ASFloatToFixed**.

Parameters

pdeExtGState	The extended graphics state object.
opacity	The new opacity value.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEExtGStateSetOpacityStroke

```
void PDEExtGStateSetOpacityStroke (PDEExtGState pdeExtGState,  
ASFixed opacity);
```

Description

Sets the opacity value for stroke operations. The value must be in the range from 0 to 1 inclusive. Corresponds to the /CA key within the **ExtGState**'s dictionary. The value from 0 to 1 refers to a float number (not an **ASFixed** value) that should be converted to **ASFixed** using **ASFloatToFixed**.

Parameters

pdeExtGState	The extended graphics state object.
opacity	The new opacity value.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEExtGStateSetOPFill

```
void PDEExtGStateSetOPFill (PDEExtGState pdeExtGState,
ASBool overprint);
```

Description

Specifies if overprint is enabled for painting operations other than stroking. Corresponds to the **/op** key within the ExtGState’s dictionary.

Parameters

pdeExtGState	The extended graphics state object.
overprint	Pass true to enable; false to disable.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEExtGStateSetOPM

```
void PDEExtGStateSetOPM (PDEExtGState pdeExtGState,  
ASInt32 overprintMode);
```

Description

Sets the overprint mode. Corresponds to the **/OPM** key within the **ExtGState**'s dictionary.

Parameters

pdeExtGState	The extended graphics state object.
overprintMode	Overprint mode.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEExtGStateSetOPStroke

```
void PDEExtGStateSetOPStroke (PDEExtGState pdeExtGState,  
ASBool overprint);
```

Description

Specifies if overprint is enabled for stroke operations. Corresponds to the **/OP** key within the ExtGState's dictionary.

Parameters

pdeExtGState	The extended graphics state object.
overprint	Pass true to enable; false to disable.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEExtGStateSetSA

```
void PDEExtGStateSetSA (PDEExtGState pdeExtGState,
ASBool strokeAdjust);
```

Description

Specifies whether stroke adjustment is enabled in the graphics state.

Parameters

pdeExtGState	The extended graphics state object.
strokeAdjust	Pass true to enable; false to disable.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDEExtGStateSetSoftMask

```
void PDEExtGStateSetSoftMask (PDEExtGState pdeExtGState,  
PDESoftMask pdeSoftMask) ;
```

Description

Sets the soft mask of the extended graphics state.

Parameters

pdeExtGState	The extended graphics state object.
pdeSoftMask	The soft mask object.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEExtGStateSetTK

```
void PDEExtGStateSetTK (PDEExtGState pdeExtGState, ASBool tk);
```

Description

Specifies whether text knockout is enabled in the graphics state. This corresponds to the **/TK** key in the ExtGState’s dictionary.

Parameters

pdeExtGState	The extended graphics state object.
tk	Pass true to enable; false to disable.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDFont

PDFontCreate

```
PDFont PDFontCreate (PDFontAttrsP attrsP,
ASInt32 attrsSize, ASInt32 firstChar, ASInt32 lastChar,
ASInt32* widthsP, char** encoding, ASAtom encodingBaseName,
ASStm fontStm, ASInt32 len1, ASInt32 len2, ASInt32 len3);
```

Description

Creates a new **PDFont** from the specified parameters.

The **PDFont** may be represented as an embedded font (a **FontFile** entry in the font descriptor of the PDF file). To create a **PDFont** that is stored as an embedded font, the **FontFile** stream may be passed in **fontStm**, and the **len1**, **len2**, and **len3** parameters contain the **Length1**, **Length2**, and **Length3** values of the **FontFile** stream attributes dictionary. See Section 5.8 in the *PDF Reference* for more information about embedded fonts.

The caller must close **fontStm** with **ASStmClose** after invoking **PDFontCreate**.

Call **PDFRelease** to dispose of the returned font object when finished with it.

Parameters

attrsP	Pointer to a PDFontAttrs structure for the font attributes.
attrsSize	Size of the attrsP buffer, in bytes.
firstChar	First character index for the widths array, widthsP .
lastChar	Last character index for the widths array, widthsP .
widthsP	Pointer to widths array.
encoding	An array of 256 pointers to glyph names specifying the custom encoding. If any pointer is NULL , no encoding information is written for that entry.
encodingBaseName	Encoding base name if the encoding is a custom encoding. If encoding is NULL , encodingBaseName is used as the value of the encoding, and must be one of WinAnsiEncoding , MacRomanEncoding , or MacExpertEncoding . If no encoding value is desired, use ASAtomNull .
fontStm	Stream with font information.

len1	Length in bytes of the ASCII portion of the Type 1 font file after it has been decoded. For other font formats, such as TrueType or CFF, only len1 is used, and it is the size of the font.
len2	Length in bytes of the encrypted portion of the Type 1 font file after it has been decoded.
len3	Length in bytes of the portion of the Type 1 font file that contains the 512 zeros, plus the cleartomark operator, plus any following data.

Return Value

The specified **PDEFont**.

Known Exceptions

[peErrCantCreateFontSubset](#)
[peErrCantGetAttrs](#)
[peErrCantGetWidths](#)
[peErrCantEmbedFont](#)
[genErrResourceLoadFailed](#)

Header File

PEWCalls.h

Related Methods

[PDEFontCreateFromCosObj](#)
[PDEFontCreateFromSysFont](#)
[PDEFontCreateFromSysFontEx](#)
[PDEFontCreateFromSysFontWithParams](#)
[PDEFontCreateWithParams](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEFontCreateFromCosObj

```
PDEFont PDEFontCreateFromCosObj (const CosObj* cosObjP);
```

Description

Creates a **PDEFont** corresponding to a Cos object of type Font.

Call **PDEFRelease** to dispose of the returned font object when finished with it.

Parameters

cosObjP	Cos object for which a PDEFont is created.

Return Value

The **PDEFont** created from **cosObjP**.

Known Exceptions

```
peErrCantCreateFontSubset  
peErrCantGetAttrs  
peErrCantGetWidths  
peErrCantEmbedFont  
genErrBadParm  
genErrResourceLoadFailed
```

Header File

PEWCalls.h

Related Methods

```
PDEFontCreate  
PDEFontCreateFromSysFont  
PDEFontCreateFromSysFontWithParams  
PDEFontCreateWithParams  
PDEFontGetCosObj
```

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEFontCreateFromSysFont

```
PDEFont PDEFontCreateFromSysFont (PDSysFont sysFont,  
ASInt32 flags);
```

Description

Gets a **PDEFont** corresponding to a font in the system.

Call **PDERelease** to dispose of the returned font object when finished with it.

The **PDEFontCreateFlags** flags **kPDEFontCreateEmbedded** and **kPDEFontWillSubset** must *both* be set in order to subset a font.

If you create a **PDEFont** that is a subset, call **PDEFontSubsetNow** on this font afterwards.

NOTE: If you have environment with no Acrobat Language kit installed, trying to acquire a **PDEFont** from the system font may raise an exception for some of the OS fonts. In other words, if you use **PDEnumSysFonts** to get the system font attributes, not all of the system fonts will necessarily be used to create the **PDEFont**.

NOTE: If you want to use WinAnsiEncoding on UNIX, do not use this method. Use **PDEFontCreateFromSysFontWithParams** or **PDEFontCreateFromSysFontAndEncoding** instead.

Parameters

sysFont	A PDSysFont object referencing a system font.
flags	Indicates whether to embed the font and whether to subset the font. Must be one of PDEFontCreateFlags . If you want to subset a font, set <i>both</i> the kPDEFontCreateEmbedded and kPDEFontWillSubset flags.

Return Value

The **PDEFont** corresponding to **sysFont**.

Known Exceptions

```
peErrCantCreateFontSubset  
peErrCantGetAttrs  
peErrCantGetWidths  
peErrCantEmbedFont  
genErrBadParm  
genErrResourceLoadFailed
```

Header File

PEWCalls.h

**Related Methods**

[PDFontCreate](#)
[PDFontCreateFromCosObj](#)
[PDFontCreateFromSysFontAndEncoding](#)
[PDFontCreateFromSysFontWithParams](#)
[PDEnumSysFonts](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEFontCreateFromSysFontAndEncoding

```
PDEFont PDEFontCreateFromSysFontAndEncoding
(PDSysFont sysFont, PDSysEncoding sysEnc,
ASAtom useThisBaseFont, ASInt32 createFlags);
```

Description

Create a **PDEFont** from **sysFont** and **sysEnc**. If it fails, it raises an exception. User can call **PDSysFontGetCreateFlags** to see if the combination of **sysFont** and **sysEnc** makes sense.

NOTE: If you want to use WinAnsiEncoding on UNIX, use this method or **PDEFontCreateFromSysFontWithParams**.

Parameters

sysFont	A PDSysFont object referencing a system font.
sysEnc	A PDSysEncoding object.
useThisBaseFont	The base font. An exception will be raised if the base font name passed is a subset name (XXXXXX+FontName) or an empty string.
createFlags	One of the PDEFontCreateFlags .

Return Value

The newly created **PDEFont** object.

Known Exceptions

Numerous

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEFontCreateFromSysFontEx

```
PDEFont PDEFontCreateFromSysFontEx (PDSysFont sysFont,  
ASInt32 flags, ASAtom snapshotName, ASFixed* mmDesignVec);
```

Description

Creates a **PDEFont** corresponding to a font in the system.

If the font is a multiple master font, **mmDesignVector** points to the design vector, whose length must equal the number of design axes of the font.

Call **PDEFRelease** to dispose of the returned font object when finished with it.

The **PDEFontCreateFlags** flags **kPDEFontCreateEmbedded** and **kPDEFontWillSubset** must *both* be set in order to subset a font.

If you create a **PDEFont** that is subsetting, call **PDEFontSubsetNow** on this font afterwards.

NOTE: If you have environment with no Acrobat Language kit installed, trying to acquire a **PDEFont** from the system font may raise an exception for some of the system fonts. In other words, if you use **PDEnumSysFonts** to get the system font attributes, not all of the system fonts are necessarily used to create the **PDEFont**.

Parameters

sysFont	A PDSysFont object referencing a system font.
flags	Indicates whether to embed the font and whether to subset the font. Must be one of PDEFontCreateFlags . If you want to subset a font, set <i>both</i> the kPDEFontCreateEmbedded and kPDEFontWillSubset flags.
snapshotName	Name to be associated with this particular instantiation of the PDEFont .
mmDesignVec	Pointer to multiple master font design vector.

Return Value

The **PDEFont** corresponding to **sysFont**.

Known Exceptions

```
peErrCantCreateFontSubset  
peErrCantGetAttrs  
peErrCantGetWidths  
peErrCantEmbedFont  
genErrBadParm  
genErrResourceLoadFailed
```

**Header File**

PEWCalls.h

Related Methods

[PDEFontCreateFromCosObj](#)
[PDEFontCreateFromSysFont](#)
[PDEFontCreateFromSysFontWithParams](#)
[PDEFontCreateWithParams](#)
[PDEnumSysFonts](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDFontCreateFromSysFontWithParams

```
PDFont PDFontCreateFromSysFontWithParams (PDSysFont sysFont,
PDFontCreateFromSysFontParams params);
```

Description

Used to obtain a **PDFont** corresponding to a font in the system.

NOTE: If you want to use WinAnsiEncoding on UNIX, use this method or **PDFontCreateFromSysFontAndEncoding** instead.

Parameters

sysFont	The system font.
params	The parameters structure.

Return Value

The newly created **PDFont** object.

Known Exceptions

peErrCantCreateFontSubset
genErrBadParm

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDFontCreateToUnicodeNow

```
void PDFontCreateToUnicodeNow (PDFont font, CosDoc cosDoc);
```

Description

This function creates the **/ToUnicode** table. The user can check the return value of [PDFontGetCreateNeedFlags](#) to see if calling [PDFontCreateToUnicodeNow](#) is needed.

Parameters

font	An object of type PDFont.
cosDoc	The container document.

Return Value

None

Known Exceptions

[genErrBadParm](#)
[peErrWrongPDEObjectType](#)

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDEFontCreateWidthsNow

```
void PDEFontCreateWidthsNow (PDEFont font, CosDoc cosDoc);
```

Description

This function creates width entries for font. User can check the return value of [PDEFontGetCreateNeedFlags](#) to see if calling [PDEFontCreateWidthsNow](#) is needed.

Parameters

font	The font to create width entries for.
cosDoc	The container document.

Return Value

None

Known Exceptions

[genErrBadParm](#)
[peErrWrongPDEObjectType](#)

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEFontCreateWithParams

```
PDEFont PDEFontCreateWithParams (PDEFontCreateParams params);
```

Description

Creates a new **PDEFont** from **params**.

The **PDEFont** may be represented as an embedded font (a **FontFile** value in PDF). To create a **PDEFont** that will be stored as an embedded font, the **FontFile** stream may be passed as **fontStm**, and the **len1**, **len2**, and **len3** parameters contain the **Length1**, **Length2**, and **Length3** values of the **FontFile**. The caller must close the **fontStm** after calling this method. This method supports multi-byte fonts.

This method extends **PDEFontCreate** to support multi-byte fonts.

Call **PDERelease** to dispose of the returned font object when finished with it.

Parameters

params	Pointer to a structure containing all font parameters necessary to fully define a font.

Return Value

A **PDEFont** object of the font described by the parameters.

Known Exceptions

```
peErrCantCreateFontSubset
peErrCantGetAttrs
peErrCantGetWidths
peErrCantEmbedFont
genErrBadParm
genErrResourceLoadFailed
```

Header File

PEWCalls.h

Related Methods

```
PDEFontCreate
PDEFontCreateFromCosObj
PDEFontCreateFromSysFont
PDEFontCreateFromSysFontEx
```

Product

base, pro, pdf

**Availability**

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEFontEmbedNow

```
void PDEFontEmbedNow (PDEFont font, CosDoc cosDoc);
```

Description

This function embeds font stream. User can check the return value of [PDEFontGetCreateNeedFlags](#) to see if calling [PDEFontEmbedNow](#) is needed.

Parameters

font	The font to embed.
cosDoc	The container document.

Return Value

None

Known Exceptions

[genErrBadParm](#)
[peErrWrongPDEObjectType](#)

Related Methods

[PDEFontEmbedNowDontSubset](#)
[PDEFontIsEmbedded](#)

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEFontEmbedNowDontSubset

```
void PDEFontEmbedNowDontSubset (PDEFont font, CosDoc doc);
```

Description

Embeds the given **PDEFont** inside **doc** without creating a subset. Use this method instead of **PDEFontSubsetNow** if you created **font** with the **willSubset** flag but changed your mind.

Parameters

font	The font to embed.
doc	The container document.

Return Value

None

Related Methods

[PDEFontEmbedNow](#)
[PDEFontIsEmbedded](#)

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDFontGetAttrs

```
void PDFontGetAttrs (PDFont font, PDFontAttrsP attrsP,
ASInt32 attrsSize);
```

Description

Gets the attributes for a font object.

Parameters

font	A PDFont whose attributes are found.
attrsP	(Filled by the method) Pointer to a PDFontAttrs structure for the font attributes.
attrsSize	Size of the attrsP buffer, in bytes.

Return Value

None

Known Exceptions

- peErrCantGetAttrs
- genErrBadParm
- genErrResourceLoadFailed

Header File

PERCalls.h

Related Methods

PDFontGetNumCodeBytes

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEFontGetCosObj

```
void PDEFontGetCosObj (PDEFont font, CosObj* cosObjP);
```

Description

Gets a Cos object for a **PDEFont**.

Parameters

font	A PDEFont whose Cos object is obtained.
cosObjP	<i>(Filled by the method)</i> The Cos object corresponding to font .

Return Value

None

Known Exceptions

[genErrResourceLoadFailed](#)
[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Related Methods

[PDEFontCreateFromCosObj](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEFontGetCreateNeedFlags

```
ASInt32 PDEFontGetCreateNeedFlags (PDEFont font);
```

Description

This function returns flags indicating what needs to be done to make **PDEFont** complete. **kPDEFontCreateNeedWidths** can be cleared by **PDEFontCreateWidthsNow**. **kPDEFontCreateNeedToUnicode** can be cleared by **PDEFontCreateToUnicodeNow**. **kPDEFontCreateNeedEmbed** can be cleared by **PDEFontEmbedNow**.

Parameters

font	The font object.

Return Value

A value corresponding to **PDEFontCreateNeedFlags**.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDFontGetNumCodeBytes

```
ASInt32 PDFontGetNumCodeBytes (PDFont font, ASUns8* text,
ASInt32 len);
```

Description

Gets the number of bytes comprising the next code in a string of single or multi-byte character codes.

Parameters

font	A PDFont object returned from one of the PDFontCreate methods.
text	Pointer into a string of characters.
len	The length, in bytes, of the string of characters, starting with the character pointed to by text .

Return Value

Number of bytes in the next character code pointed to by **text**.

Known Exceptions

[genErrNoMemory](#)

Header File

PERCalls.h

Related Methods

[PDFontIsMultiByte](#)
[PDFontSumWidths](#)
[PDFontGetOneByteEncoding](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDFontGetOneByteEncoding

```
ASBool PDFontGetOneByteEncoding (PDFont font,
ASAtom* encodingDelta);
```

Description

Gets an array of delta encodings for the given one byte PDFont.

Parameters

font	A PDFont object returned from one of the PDFontCreate methods.
encodingDelta	(Filled by the method) Pointer to an ASAtom array that is filled with the delta encodings for font. Each entry is the ASAtom for a glyph name that differs from the base encoding. See Section 5.5.5 in the PDF Reference for more information about font encodings. The array must be allocated to hold 256 entries.

Return Value

true if encodingDelta is filled, false otherwise.

Known Exceptions

genErrNoMemory

Header File

PERCalls.h

Related Methods

- PDFontIsMultiByte
- PDFontSumWidths
- PDFontGetNumCodeBytes

Product

base, pro, pdf

Availability

Available if PI_PDFEDIT_READ_VERSION (in PIRequir.h) is set to 0x00040000 or higher.

PDFontGetSysEncoding

[PDSysEncoding](#) PDFontGetSysEncoding ([PDFont](#) pdfFont) ;

Description

Gets the system encoding object associated with a font object.

Parameters

pdfFont	A PDFont whose system encoding is found.

Return Value

The system encoding object.

Known Exceptions

[genErrBadParm](#)
[genErrResourceLoadFailed](#)

Header File

PERCalls.h

Related Methods

[PDFontSetSysEncoding](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDEFontGetSysFont

```
PD SysFont PDEFontGetSysFont (PDEFont pdeFont) ;
```

Description

Gets the system font object associated with a font object.

Parameters

pdeFont	A PDEFont whose system font is found.

Return Value

The system font object.

Known Exceptions

[genErrBadParm](#)
[genErrResourceLoadFailed](#)

Header File

PERCalls.h

Related Methods

[PDFindSysFontForPDEFont](#)
[PDEFontSetSysFont](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDEFontGetWidths

```
void PDEFontGetWidths (PDEFont font, ASInt16* widthsP);
```

Description

Gets the widths for a font object.

Parameters

font	A PDEFont whose widths are found.
widthsP	<i>(Filled by the method)</i> Pointer to widths array. widthsP must have room for 256 values. The widths are returned in character space (1000 EM units). An EM is a typographic unit of measurement equal to the size of a font. To convert to text space, divide the value returned by 1000. To convert to user space, multiply the text space value by the font size.

Return Value

None

Known Exceptions

[peErrCantGetWidths](#)
[genErrBadParm](#)
[genErrResourceLoadFailed](#)

Header File

PERCalls.h

Related Methods

[PDEFontCreateWithParams](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDFontGetWidthsNow

```
void PDFontGetWidthsNow (PDFont font, CosDoc doc);
```

Description

Gets a Type0 font's width information for only the characters used in the file. Call this routine when the font was created with the **kPDFontDeferWidths** flag but without the **kPDFontCreateEmbedded** flag (if the font is to be embedded, call **PDFontSubsetNow**, which also gets the width info).

Parameters

font	The font whose widths are found.
doc	The container document.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEFontIsEmbedded

```
ASBool PDEFontIsEmbedded (PDEFont pdfFont);
```

Description

Tests whether a font is an embedded font in the document in which it was created.

Parameters

pdfFont	A PDEFont object to test.

Return Value

true if the font is embedded, **false** if it is not, or if it was created in one document and embedded in a different document.

Header File

PERCalls.h

Related Methods

[PDEFontEmbedNow](#)

[PDEFontEmbedNowDontSubset](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



PDFontIsMultiByte

`ASBool PDFontIsMultiByte (PDFont font);`

Description

Tests whether a font contains any multi-byte characters.

Parameters

<code>font</code>	A <code>PDFont</code> object returned from one of the <code>PDFontCreate</code> methods to test.

Return Value

`true` if the font contains any multi-byte characters, `false` otherwise.

Header File

`PERCalls.h`

Related Methods

- `PDFontGetNumCodeBytes`
- `PDFontSumWidths`
- `PDFontGetOneByteEncoding`

Product

base, pro, pdf

Availability

Available if `PI_PDFEDIT_READ_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.



PDFontSetSysEncoding

```
void PDFontSetSysEncoding (PDFont pdfFont,
    PDSysEncoding sysEnc);
```

Description

Sets the system encoding object associated with a font object.

NOTE: Changing the system encoding may produce unexpected results.

Parameters

pdfFont	A PDFont whose system encoding is set.
sysEnc	The new system encoding object.

Return Value

None

Known Exceptions

- genErrBadParm
- genErrResourceLoadFailed

Header File

PEWCalls.h

Related Methods

PDFontGetSysEncoding

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



PDEFontSetSysFont

```
void PDEFontSetSysFont (PDEFont pdeFont, PDSysFont sysFont);
```

Description

Sets the system font object to be used with a font object that does not currently have a system font associated with it.

Parameters

pdeFont	A PDEFont whose system font is set.
sysFont	The new system font object.

Return Value

None

Known Exceptions

- genErrBadParm
- genErrResourceLoadFailed

Header File

PEWCalls.h

Related Methods

PDEFontGetSysFont

Product

base, pro, pdf

Availability

Available if `PI_PDFEDIT_WRITE_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

PDFontSubsetNow

```
void PDFontSubsetNow (PDFont font, CosDoc cosDoc);
```

Description

Subsets a given **PDFont** in a **CosDoc**.

If you created **font** with **PDFontCreateFromSysFont**, you must have set *both* the **kPDFontCreateEmbedded** and **kPDFontWillSubset** set in the **flags** parameter to be able to subset **font**.

NOTE: This method does not change the reference count.

Parameters

font	The PDFont to subset.
cosDoc	The CosDoc whose font is subsetted.

Return Value

None

Known Exceptions

[peErrCantCreateFontSubset](#)
[peErrCantGetAttrs](#)
[peErrCantGetWidths](#)
[peErrCantEmbedFont](#)
[genErrBadParm](#)
[genErrResourceLoadFailed](#)

Header File

PEWCalls.h

Related Methods

[PDFontCreateFromSysFont](#)

Example

```
PDDoc pdoc;
PDPage pdPage;
PDEFont gFont;

PDPageAcquirePDEContent(pdPage, extensionID);

/* Make sure you specify both font flags */
gFont = PDEFontCreateFromSysFont(sysFont, (kPDEFontCreateEmbedded|
kPDEFontWillSubset));

/* Add your text objects here */
...

PDPageSetPDEContent(pdPage, extensionID);
/* Make sure to call PDEFontSubsetNow after setting the PDEContent but
before releasing it */
PDEFontSubsetNow(gFont, PDDocGetCosDoc(pdoc));

PDERelease(gFont);

PDPageReleasePDEContent(pdPage, extensionID);
```

Product

base, pro, pdfi

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDFontSumWidths

```
ASInt32 PDFontSumWidths (PDFont font, ASUns8* text,
ASInt32 len);
```

Description

Gets the sum to the widths of **len** characters from a string of single or multi-byte characters.

Parameters

font	A PDFont object returned from one of the PDFontCreate methods.
text	Pointer into a string of characters.
len	Number of characters in the string.

Return Value

Width of text string in EM space. (In EM space, the width of "M" is about 1000 EM units).

Known Exceptions

```
genErrNoMemory
pdErrBadResMetrics
genErrResourceLoadFailed
peErrWrongPDEObjectType
```

Header File

```
PERCalls.h
```

Related Methods

```
PDFontGetNumCodeBytes
PDFontIsMultiByte
PDFontGetOneByteEncoding
```

Product

```
base, pro, pdf
```

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDFontTranslateGlyphIdsToUnicode

```
ASInt32 PDFontTranslateGlyphIdsToUnicode (PDFont font,
ASUns8* text, ASInt32 textLen, ASUns8* unicodeStr,
ASInt32 size);
```

Description

Translates a string to Unicode values. The **PDFont** must have a **ToUnicode** table.

Parameters

font	The font.
text	The string to convert.
textLen	The length of text , in bytes.
unicodeStr	<i>(Filled by the method)</i> Buffer to hold the translated string.
size	The size of the unicodeStr buffer.

Return Value

0 if the string was successfully translated. If **unicodeStr** is too small for the translated string, it returns the number of bytes required.

Known Exceptions

[genErrBadParm](#)

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEForm

PDEFormAcquireXGroup

```
PDEXGroup PDEFormAcquireXGroup (PDEForm pdeForm) ;
```

Description

Acquires the transparency group dictionary of the XObject form.

Parameters

<code>pdeForm</code>	The from.

Return Value

Transparency group object.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if `PI_PDFEDIT_READ_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.



PDEFormCreateClone

```
PDEForm PDEFormCreateClone (PDEForm form) ;
```

Description

Creates a new form from an existing form object. Creates a copy of the **PDEForm**, including the underlying **CosStream**.

Parameters

form	Form object from which a new PDEForm is created.

Return Value

The newly created form object.

Header File

PEWCalls.h

Related Methods

[PDEFormCreateFromCosObj](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDEFormCreateFromCosObj

```
PDEForm PDEFormCreateFromCosObj (const CosObj* xObjectP,  
const CosObj* resourcesP, ASFixedMatrixP matrixP);
```

Description

Creates a new form from an existing Cos object.
Call [PDERelease](#) to dispose of the returned form object when finished with it.

Parameters

xObjectP	Cos object from which a PDEForm is created.
resourcesP	The xObjectP 's Resources dictionary. If you do not pass in a Resource object, subsequent calls to PDPageAcquirePDEContent will fail (after the file is saved).
matrixP	Pointer to ASFixedMatrix that holds the transformation matrix to use for the form.

Return Value

The newly created form object.

Header File

PEWCalls.h

Related Methods

[PDEFormCreateClone](#)
[PDEFormGetCosObj](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEFormGetContent

`PDEContent` PDEFormGetContent (`PDEForm` form);

Description

Gets a `PDEContent` object for a form.

NOTE: Unlike other “GetContent” methods, this method does increment the reference count of the returned `PDEContent`.

Parameters

<code>form</code>	The form whose content is obtained.

Return Value

Content for `form`.

Known Exceptions

`peErrWrongPDEObjectType`
`peErrPStackUnderflow`

Header File

`PEWCalls.h`

Product

base, pro, pdf

Availability

Available if `PI_PDFEDIT_WRITE_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.



PDEFormGetCosObj

```
void PDEFormGetCosObj (PDEForm form, CosObj* cosObjP);
```

Description

Gets a Cos object for a form.

Parameters

form	The form whose Cos object is obtained.
cosObjP	(Filled by the method) Cos object for the form.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Related Methods

[PDEFormCreateFromCosObj](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEFormHasXGroup

```
ASBool PDEFormHasXGroup (PDEForm pdeForm) ;
```

Description

Determines whether the XObject form has a Transparency XGroup

Parameters

pdeForm	The form.

Return Value

true if the XObject form has a Transparency XGroup.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in PIRequir.h) is set to **0x00050000** or higher.

PDEFormSetContent

```
void PDEFormSetContent (PDEForm form, PDEContent content);
```

Description

Sets the underlying **CosStream** of the form using the specified content object.

Parameters

form	The form whose content is set.
content	The new content for form .

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[peErrPStackUnderflow](#)

Header File

PEWCalls.h

Related Methods

[PDEFormGetContent](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



PDEFormSetXGroup

```
void PDEFormSetXGroup (PDEForm pdeForm, PDEXGroup pdeXGroup);
```

Description

Sets the transparency group dictionary of the form XObject.

Parameters

pdeForm	The font XObject.
pdeXGroup	The transparency dictionary.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



PDEGroup

PDEGroupCreate

```
PDEGroup PDEGroupCreate (void);
```

Description

Creates a **PDEGroup** object.

Parameters

None

Return Value

The newly created **PDEGroup**.

Header File

PEWCalls.h

Related Methods

[PDEGroupSetContent](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEGroupGetContent

```
PDEContent PDEGroupGetContent (PDEGroup pdeGroup);
```

Description

Gets the **PDEContent** for a **PDEGroup**.

NOTE: This method does not change the reference count of the returned **PDEContent**.

Parameters

pdeGroup	The group whose content is obtained.

Return Value

The **PDEContent** in **pdeGroup**.

Known Exceptions

peErrWrongPDEObjectType

Header File

PERCalls.h

Related Methods

PDEGroupSetContent

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEGroupSetContent

```
void PDEGroupSetContent (PDEGroup pdeGroup,
PDEContent pdeContent);
```

Description

Sets the **PDEContent** for a **PDEGroup**. The existing **PDEContent** is released by this method.

NOTE: This method increments the reference count of **pdeContent**.

Parameters

pdeContainer	A container object.
pdeContent	The content to set for pdeGroup .

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PEWCalls.h

Related Methods

[PDEGroupGetContent](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEImage

PDEImageCreate

```
PDEImage PDEImageCreate (PDEImageAttrsP attrsP,
ASInt32 attrsSize, ASFixedMatrixP matrixP, ASInt32 flags,
PDEColorSpace colorSpace, PDEColorValueP colorValueP,
PDEFilterArrayP filtersP, ASStm dataStm, ASUns8* data,
ASInt32 encodedLen);
```

Description

Creates an image object.

The image data may be specified as a stream or as a buffer. If **dataStm** is non-NULL, **data** is ignored.

See [PDEImageSetDataStm](#) for information on handling the stream.

The caller must dispose of **datStm** after calling this method.

Call [PDERelease](#) to dispose of the returned image object when finished with it.

Parameters

attrsP	Pointer to PDEImageAttrs with attributes of the image.
attrsSize	Size of the attrsP buffer, in bytes.
matrixP	Pointer to ASFixedMatrix that holds the transformation matrix to use for the image.
flags	PDEImageDataFlags flags. If the kPDEImageEncodedData flag is set, and the data is provided directly (not as a stream), then encodedLen must specify the length of data .
colorSpace	Color space of the image. When the image is an imagemask, colorSpace is the color space of the colorValueP argument.
colorValueP	Pointer to PDEColorValue structure. If the image is an image mask, colorValueP must be provided.
filtersP	Pointer to PDEFilterArray structure that specifies which filters to use in encoding the contents; may be NULL . Filters will be used to encode the data in the order in which they are specified in the array.
dataStm	Stream holding the image data.



data	Image data. If dataStm is non- NULL , data is ignored. If there is a great deal of data, as for a large image, it is recommended you use the dataStm parameter for the image data or use the PDEImageCreateFromCosObj method.
encodedLen	Encoded length of data , in bytes.

Return Value

The image.

Known Exceptions

[peErrUnknownPDEColorSpace](#)
[pageErrReadLessImageData](#)
[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PEWCalls.h

Related Methods

[PDEImageCreateFromCosObj](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEImageCreateFromCosObj

```
PDEImage PDEImageCreateFromCosObj (const CosObj* imageObjP,
ASFixedMatrixP matrixP, PDEColorSpace colorSpace,
PDEColorValueP colorValueP);
```

Description

Creates an image object from a Cos object.

Call [PDERelease](#) to dispose of the returned image object when finished with it.

Parameters

imageObjP	Cos object for the image.
matrixP	Pointer to ASFixedMatrix that holds the transformation matrix to use for the image.
colorSpace	Color space used for the image, if the image is an image mask; otherwise, set to NULL .
colorValueP	Pointer to PDEColorValue structure. If the image is an imagemask, colorValueP must be provided.

Return Value

Image corresponding to the Cos object.

Known Exceptions

[peErrUnknownPDEColorSpace](#)
[pageErrReadLessImageData](#)
[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PEWCalls.h

Related Methods

[PDEImageCreate](#)
[PDEImageGetCosObj](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in [PIRequir.h](#)) is set to **0x00040000** or higher.

PDEImageDatalsEncoded

```
ASBool PDEImageDataIsEncoded (PDEImage image,
ASUns32* encodedLenP);
```

Description

Determines if image data is encoded or not. Used only for inline images; not relevant to XObject images.

Always returns **false** for XObject images; XObject image data can be obtained from PDEImageGetData or PDEImageGetDataStm, either encoded or decoded.

If an inline image is obtained via the PDEContentCreateFromCosObj or related methods, the inline image data is always decoded. That is, if PDFEdit parses the stream, the data is always decoded. Only if PDEImageCreate is used to explicitly create a new image using encoded data does PDEImageDataIsEncoded return **true**.

Parameters

image	Image to examine.
encodedLenP	(Filled by the method) Length of the encoded data—if the data is encoded, that is, the method returns true .

Return Value

true if PDEImageGetData returns encoded data, **false** otherwise. Returns **false** for XObject images.

Known Exceptions

```
peErrWrongPDEObjectType
```

Header File

```
PERCalls.h
```

Related Methods

```
PDEImageGetData
```

Product

```
base, pro, pdf
```

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in PIREquir.h) is set to **0x00040000** or higher.

PDEImageGetAttrs

```
void PDEImageGetAttrs (PDEImage image, PDEImageAttrsP attrsP,
ASInt32 attrsSize);
```

Description

Gets the attributes for an image.

Parameters

image	Image whose attributes are obtained.
attrsP	<i>(Filled by the method)</i> Pointer to PDEImageAttrs structure with attributes of image .
attrsSize	Size of the attrsP buffer, in bytes.

Return Value

None

Known Exceptions

[peErrUnknownPDEColorSpace](#)

Header File

PERCalls.h

Related Methods

[PDEImageGetColorSpace](#)
[PDEImageGetData](#)
[PDEImageGetDataLen](#)
[PDEImageGetDataStm](#)
[PDEImageGetDecodeArray](#)
[PDEImageGetFilterArray](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in [PIRequir.h](#)) is set to **0x00040000** or higher.

PDEImageGetColorSpace

```
PDEColorSpace PDEImageGetColorSpace (PDEImage image);
```

Description

Gets the color space object for an image.

NOTE: (For the Adobe PDF Library v1 only) If you get the [PDEColorSpace](#) for an inline image, then get the [CosObj](#) for that color space with [PDEColorSpaceGetCosObj](#), this [CosObj](#) is limited. Cos objects that are the result of parsing inline dictionaries in the PDF page contents are a special class of Cos objects. You should never depend on these objects lasting the lifetime of the document. You should extract the information you need from the object immediately and refer to it no further in your code.

NOTE: This method does not change the reference count of the returned [PDEColorSpace](#).

Parameters

image	Image whose color space is obtained.

Return Value

Color space for **image**. Returns **NULL** if **image** is an image mask.

Header File

PERCalls.h

Related Methods

[PDEImageGetAttrs](#)
[PDEImageGetData](#)
[PDEImageGetDataLen](#)
[PDEImageGetDataStm](#)
[PDEImageGetDecodeArray](#)
[PDEImageGetFilterArray](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEImageGetCosObj

```
void PDEImageGetCosObj (PDEImage image, CosObj* cosObjP);
```

Description

Gets a Cos object for an image.

Parameters

image	Image whose Cos object is obtained.
cosObjP	(Filled by the method) Cos object for the image.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Related Methods

- [PDEImageCreateFromCosObj](#)
- [PDEImageIsCosObj](#)

Product

base, pro, pdf

Availability

Available if `PI_PDFEDIT_READ_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.

PDEImageGetData

```
void PDEImageGetData (PDEImage image, ASInt32 flags,
ASUns8* buffer);
```

Description

Gets an image's data.

If the image is an **XObject** image, data is always returned as decoded data.

See the note about inline images under [PDEImageDataIsEncoded](#).

Parameters

image	Image whose data is obtained.
flags	Unused—must be zero.
buffer	Image data. If the data is decoded, buffer must be large enough to contain the number of bytes specified in the PDEImageAttrs structure obtained by PDEImageGetAttrs . If the data is encoded, buffer must be large enough to contain the number of bytes in the encodedLenP parameter obtained by PDEImageDataIsEncoded .

Known Exceptions

[peErrUnknownPDEColorSpace](#)
[genErrBadParm](#)
[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Related Methods

[PDEImageDataIsEncoded](#)
[PDEImageSetColorSpace](#)
[PDEImageGetAttrs](#)
[PDEImageGetColorSpace](#)
[PDEImageGetDataLen](#)
[PDEImageGetDataStm](#)
[PDEImageGetDecodeArray](#)
[PDEImageGetFilterArray](#)

Product

base, pro, pdf



Availability

Available if **PI_PDFEDIT_READ_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.



PDEImageGetDataLen

```
ASInt32 PDEImageGetDataLen (PDEImage image);
```

Description

Gets the length of data for an image.

Parameters

image	Image whose data length is obtained.

Return Value

Number of bytes of image data, specified by the width, height, bits per component, and color space of the image.

Known Exceptions

[peErrUnknownPDEColorSpace](#)
[genErrBadParm](#)
[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Related Methods

[PDEImageGetData](#)

Product

base, pro, pdf

Availability

Available if `PI_PDFEDIT_READ_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.

PDEImageGetDataStm

```
ASStm PDEImageGetDataStm (PDEImage image, ASInt32 flags);
```

Description

Gets a data stream for an image. May only be called for **XObject** images.

The caller must dispose of the returned **ASStm** by calling **ASStmClose**.

Parameters

image	Image whose data stream is obtained.
flags	PDEImageDataFlags flags. If the kPDEImageEncodedData flag is set, data is returned in encoded form. Otherwise, data is decoded.

Return Value

Stream for **image**.

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm

Header File

PERCalls.h

Related Methods

PDEImageSetDataStm
PDEImageGetDataLen

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEImageGetDecodeArray

```
ASInt32 PDEImageGetDecodeArray (PDEImage image,
ASFixed* decode, ASInt32 decodeSize);
```

Description

Gets the decode array for an image.

Parameters

image	The image whose decode array is obtained.
decode	<i>(Filled by the method)</i> Pointer to the decode array. If NULL , the number of decode elements required is returned.
decodeSize	Size of decode in bytes.

Return Value

Number of elements in the decode array.

Header File

PERCalls.h

Related Methods

[PDEImageSetDecodeArray](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEImageGetFilterArray

```
ASInt32 PDEImageGetFilterArray (PDEImage image,
                                PDEFilterArrayP filtersP);
```

Description

Gets the filter array for an image.

Parameters

image	Image whose filter array is obtained.
filtersP	<i>(Filled by the method)</i> Pointer to PDEFilterArray structure to fill with the current filter array for the image. filtersP must be large enough to contain all of the elements. May be NULL to obtain the number of filter elements.

Return Value

Number of filter elements.

Known Exceptions

peErrWrongPDEObjectType

Header File

PERCalls.h

Related Methods

- PDEImageGetAttrs**
- PDEImageGetColorSpace**
- PDEImageGetDataLen**
- PDEImageGetDecodeArray**

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEImageGetMatteArray

```
ASInt32 PDEImageGetMatteArray (PDEImage image, ASFixed* matte,  
ASInt32 numComp) ;
```

Description

Gets the matte array for the image XObject.

Parameters

image	The image XObject.
matte	<i>(Filled by the method)</i> An array of values.
numComp	The number of values in matte .

Return Value

Number of values copied.

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEImageGetSMask

```
PDEImage PDEImageGetSMask (PDEImage image);
```

Description

Gets the soft mask for an image. Use [PDERelease](#) to dispose of the object when it is no longer referenced.

Parameters

image	An object of type PDEImage .

Return Value

An object of type **PDEImage**.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEImageHasSMask

```
ASBool PDEImageHasSMask (PDEImage image) ;
```

Description

Checks whether the image has a soft mask.

Parameters

<code>image</code>	An object of type <code>PDEImage</code> .

Return Value

`true` if the soft mask exists; `false` otherwise.

Known Exceptions

```
peErrWrongPDEObjectType
```

Header File

```
PERCalls.h
```

Product

```
base, pro, pdf
```

Availability

Available if `PI_PDFEDIT_READ_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.



PDEImageIsCosObj

```
ASBool PDEImageIsCosObj (PDEImage image);
```

Description

Determines if an image is an XObject image.

Parameters

image	Image to examine.

Return Value

true if the image is an XObject image, **false** otherwise.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Related Methods

[PDEImageGetCosObj](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEImageSetColorSpace

```
void PDEImageSetColorSpace, (PDEImage image,  
PDEColorSpace space)
```

Description

Sets the color space of the image.

Parameters

image	Image whose color space is obtained.
space	An object of type PDEColorSpace .

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PEWCalls.h

Related Methods

[PDEImageGetColorSpace](#)
[PDEImageSetColorSpace](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEImageSetData

```
void PDEImageSetData (PDEImage image, ASInt32 flags,
ASUns8* buffer, ASInt32 encodedLen);
```

Description

Sets data for an image.

Parameters

image	Image whose data is set.
flags	A set of PDEImageDataFlags flags. If kPDEImageEncodedData is set, the data must be encoded for the current filters, and encodedLen is the length of the encoded data. If the kPDEImageEncodedData flag is not set, data is not encoded and encodedLen is the size of the decoded data.
buffer	Image data.
encodedLen	Length of the encoded data.

Return Value

None

Known Exceptions

[peErrUnknownPDEColorSpace](#)
[genErrBadParm](#)
[peErrWrongPDEObjectType](#)

Header File

PEWCalls.h

Related Methods

[PDEImageGetData](#)
[PDEImageGetDataLen](#)
[PDEImageGetDataStm](#)

Product

base, pro, pdf

Availability

Available if [PI_PDFEDIT_WRITE_VERSION](#) (in [PIRequir.h](#)) is set to [0x00040000](#) or higher.

PDEImageSetDataStm

```
void PDEImageSetDataStm (PDEImage image, ASInt32 flags,
PDEFilterArrayP filtersP, ASStm stm);
```

Description

Sets a data stream for an image. Can only be used for **XObject** images.
The caller must dispose of the stream by calling **ASStmClose**.

Parameters

image	Image whose data stream is set.
flags	PDEImageDataFlags flags. If the kPDEImageEncodedData flag is set, the stream must be encoded.
filtersP	Pointer to PDEFilterArray structure. If not NULL, is used to build Cos objects for the Filter, DecodeParms, and EncodeParms objects. If filtersP is NULL, the existing Filter and DecodeParms are used. EncodeParms is set to DecodeParms if it exists.
stm	Stream for the image data.

Return Value

None

Known Exceptions

- peErrUnknownPDEColorSpace
- peErrWrongPDEObjectType
- genErrBadParm

Header File

PEWCalls.h

Related Methods

- PDEImageGetData
- PDEImageGetDataLen
- PDEImageGetDataStm

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEImageSetDecodeArray

```
void PDEImageSetDecodeArray (PDEImage image, ASFixed* decode,
ASInt32 decodeSize);
```

Description

Sets the decode array of an image.

Normally, the decode array is accessed through the **decode** field in the [PDEImageAttrs](#) structure. However, this method defines a decode array to handle images with a color space that has more than 4 components.

Parameters

image	Image whose decode array is set.
decode	Pointer to the decode array.
decodeSize	Size of decode array in bytes.

Return Value

None

Header File

PEWCalls.h

Related Methods

[PDEImageGetDecodeArray](#)

[PDEImageGetFilterArray](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEImageSetMatteArray

```
void PDEImageSetMatteArray (PDEImage pdeImage,  
    ASFixed* mArray, ASInt32 numComp);
```

Description

Sets the matte array for the image XObject.

Parameters

pdeImage	The image XObject.
mArray	An array of values.
numComp	The number of values in mArray .

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEImageSetSMask

```
void PDEImageSetSMask (PDEImage pdeImage, PDEImage sMask);
```

Description

Sets the soft mask.

Parameters

pdeImage	The image XObject.
sMask	The soft mask.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEObject

PDEAcquire

```
void PDEAcquire (PDEObject obj);
```

Description

Increments the reference count for an object.

Parameters

obj	The element whose count is incremented.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Related Methods

[PDERelease](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEAddTag

```
void PDEAddTag (PDEObject object, ASInt32 clientID,
ASInt32 key, void* valuePtr);
```

Description

Adds an identifier–value pair to an object.

The **clientID**–**tag** combination is a unique identifier for the value. Each client has its own identifier space. It is often convenient to use **ASAtoms** as tags.

NOTE: Tags are a purely memory-resident feature. In addition, management of tags are the responsibility of the client. A client must manage any memory pointed to by a tag. This method only contains a pointer to the data passed in by the client. The data and the pointer will not be saved to a file; the generic pointer type is not in the PDF specification.

Parameters

object	The element to tag. object may be a PDEElement , PDEContent , PDEFont , PDEColorSpace , and so on.
clientID	Identifies the caller/client. For clients, this should be the gExtensionID extension. For the Adobe PDF Library, if there is only one client of the PDFEdit subsystem, clientID should be zero. If there are multiple clients, each should specify a nonzero, non-negative clientID . (A negative clientID is reserved for the implementation.)
tag	The tag to add to object . If tag is 0, this is the same as calling PDERemoveTag . In other words, you cannot tell the difference between a tag whose value is zero and a tag that is nonexistent.
valuePtr	Pointer to a value to associate with object . Only the pointer is stored. If the pointer points to data, it is the responsibility of the client to manage the data and its memory.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType

Header File

PEWCalls.h



Related Methods

[PDEGetTag](#)
[PDERemoveTag](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEGetTag

```
void* PDEGetTag (PDEObject object, ASInt32 clientID,  
ASInt32 tag);
```

Description

Gets an object's value for a given **clientID-tag** identifier that was added by [PDEAddTag](#).

Parameters

object	The element whose value is obtained.
clientID	Identifies the caller/client. For clients, this should be the gExtensionID extension. For the Adobe PDF Library, if there is only one client of the PDFEdit subsystem, clientID should be zero. If there are multiple clients, each should specify a nonzero, non-negative clientID . (A negative clientID is reserved for the implementation.)
tag	The object 's tag. If object has no tag, this is 0.

Return Value

The value associated with the **clientID-tag** identifier.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PEWCalls.h

Related Methods

[PDEAddTag](#)
[PDERemoveTag](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.



PDEObjectGetType

```
ASInt32 PDEObjectGetType (PDEObject obj);
```

Description

Gets the type of an element.

Parameters

obj	The element whose type is obtained.

Return Value

The object type, which is one of [PDEType](#).

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDERelease

```
void PDERelease (PDEObject obj);
```

Description

Decrements the reference count for the object. If the count becomes zero, the object is destroyed.

Do not call **PDERelease** on **PDEContent** that you acquired with **PDPAGEAcquirePDEContent**; call **PDPAGEReleasePDEContent** instead.

NOTE: Objects should only be disposed of with **PDERelease** if the method by which they were obtained incremented the reference count for the object. In general, methods that “get” an object do not increment the reference count. Methods that increment the reference count typically contain the word “acquire” or “create” in the method name and specifically state that you must release the object.

Parameters

obj	The element released.

Return Value

None

Header File

PERCalls.h

Related Methods

[PDEAcquire](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDERemoveTag

```
void PDERemoveTag (PDEObject object, ASInt32 clientID,  
ASInt32 tag);
```

Description

Removes an object's value for a given **clientID**-**tag** identifier that was added by [PDEAddTag](#).

If [PDEAddTag](#) is called with a 0 tag, this is the same as calling [PDERemoveTag](#).

Parameters

object	The element whose tag is removed.
clientID	Identifies the caller/client. For clients, this should be the gExtensionID extension. For the Adobe PDF Library, if there is only one client of the PDFEdit subsystem, clientID should be zero. If there are multiple clients, each should specify a nonzero, non-negative clientID . (A negative clientID is reserved for the implementation.)
tag	The tag value.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PEWCalls.h

Related Methods

[PDEAddTag](#)

[PDEGetTag](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in [PIRequir.h](#)) is set to **0x00040000** or higher.

PDEPath

PDEPathAddSegment

```
void PDEPathAddSegment (PDEPath path, ASInt32 segType,
    ASFixed x1, ASFixed y1, ASFixed x2, ASFixed y2, ASFixed x3,
    ASFixed y3);
```

Description

Adds a segment to a path. The number of **ASFixed** values used depends upon **segType**:

kPDEMoveTo: **x1**, **y1**

kPDELineTo: **x1**, **y1**

kPDECurveTo: **x1**, **y1**, **x2**, **y2**, **x3**, **y3**

kPDECurveToV: **x1**, **y1**, **x2**, **y2**

kPDECurveToY: **x1**, **y1**, **x2**, **y2**

kPDERect: **x1**, **y1**, **x2** (width), **y2** (height)

kPDEClosePath: None

Parameters

path	The path to which a segment is added.
data	A PDEPathElementType value indicating the type of path to add.
x1	x -coordinate of first point.
y1	y -coordinate of first point.
x2	x -coordinate of second point.
y2	y -coordinate of second point.
x3	x -coordinate of third point.
y3	y -coordinate of third point.

Return Value

None

Known Exceptions

genErrBadParm

Header File

PEWCalls.h

**Related Methods**[PDEPathSetData](#)**Product**

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.



PDEPathCreate

```
PDEPath PDEPathCreate (void);
```

Description

Creates an empty path element.

Call [PDERelease](#) to dispose of the returned path object when finished with it.

Parameters

None

Return Value

An empty path element.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEPathGetData

```
ASInt32 PDEPathGetData (PDEPath path, ASInt32* data,
ASInt32 dataSize);
```

Description

Gets the size of the path data and, optionally, the path data.

Parameters

path	The path whose data is obtained.
data	<i>(Filled by the method)</i> Pointer to path data. If data is non-NULL, it contains a variable-sized array of path operators and operands. The format is a 32-bit operator followed by 0 to 3 ASFixedPoint values, depending on the operator. Opcodes are codes for moveto, lineto, curveto, rect, or closepath operators; operands are ASFixedPoint values. If data is NULL, the number of bytes required for data is returned by the method. NOTE: Returns “raw” path data. If you want the points in page coordinates, concatenate the path data points with the PDEElement matrix obtained from PDEElementGetMatrix.
dataSize	Specifies the size of the buffer provided in data . If it is less than the length of the path data, the method copies datasize bytes.

Return Value

Length of data of **path**.

Known Exceptions

peErrWrongPDEObjectType

Header File

PERCalls.h

Related Methods

PDEPathSetData

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEPathGetPaintOp

```
ASInt32 PDEPathGetPaintOp (PDEPath path);
```

Description

Gets the fill and stroke attributes of a path.

Parameters

path	The path whose fill and stroke attributes are obtained.

Return Value

A set of [PDEPathOpFlags](#) flags describing fill and stroke attributes.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Related Methods

[PDEPathSetPaintOp](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEPathSetData

```
void PDEPathSetData (PDEPath path, ASInt32* data,
ASInt32 dataSize);
```

Description

Sets new path data for a path element.

Parameters

path	The path whose data is set.
data	Pointer to path data. It is a variable-sized array of path operators and operands. The format is a 32-bit operator followed by 0 to 3 ASFixedPoint values, depending on the operator. Operators are codes for moveto, lineto, curveto, rect, or closepath operators and must be one of PDEPathElementType. Operands are ASFixedPoint values. The data is copied into the PDEPath object.
dataSize	Size of the new path data, in bytes.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm

Header File

PEWCalls.h

Related Methods

PDEPathGetData

Product

base, pro, pdf

Availability

Available if PI_PDFEDIT_WRITE_VERSION (in PIRequir.h) is set to 0x00040000 or higher.

PDEPathSetPaintOp

```
void PDEPathSetPaintOp (PDEPath path, ASInt32 op);
```

Description

Sets the fill and stroke attributes of a path.

Parameters

path	The path whose fill and stroke attributes are set.
op	The operation to set; must be one of PDEPathOpFlags.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType

Header File

PEWCalls.h

Related Methods

PDEPathGetPaintOp

Product

base, pro, pdf

Availability

Available if `PI_PDFEDIT_WRITE_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.



PDEPattern

PDEPatternCreate

```
PDEPattern PDEPatternCreate (const CosObj* cosObjP);
```

Description

Creates a pattern object that can be used for a Pattern color space. See Section 4.6 in the [PDF Reference](#) for more information about patterns.

Call [PDERelease](#) to dispose of the returned pattern object when finished with it.

Parameters

cosObjP	A CosStream for the pattern.

Return Value

A pattern.

Header File

PEWCalls.h

Related Methods

[PDEPatternGetCosObj](#)

Product

base, pro, pdf

Availability

Available if `PI_PDFEDIT_WRITE_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.



PDEPatternGetCosObj

```
void PDEPatternGetCosObj (PDEPattern pattern,  
CosObj* cosObjP);
```

Description

Gets a Cos object corresponding to a pattern object.

Parameters

pattern	Pattern whose Cos object is obtained.
cosObjP	(Filled by the method) Cos object for the pattern.

Return Value

None

Header File

PERCalls.h

Related Methods

[PDEPatternCreate](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEPlace

PDEPlaceCreate

```
PDEPlace PDEPlaceCreate (ASAtom mcTag, CosObj* cosObjP,  
ASBool isInline);
```

Description

Creates a place object.

Call [PDERelease](#) to dispose of the returned place object when finished with it.

Parameters

mcTag	Tag name for the place. Must not contain any white space characters (for example, spaces or tabs).
cosObjP	Optional Marked Content dictionary associated with the place.
isInline	If true , place is emitted into the page content stream inline.

Return Value

The place object.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEPlaceGetDict

```
ASBool PDEPlaceGetDict (PDEPlace pdePlace, CosObj* placeDictP,
ASBool* isInline);
```

Description

Gets the Marked Content dictionary for a **PDEPlace**.

Parameters

pdePlace	The place whose Marked Content dictionary is obtained.
placeDictP	<i>(Filled by the method)</i> Pointer to the Marked Content dictionary; may be NULL .
isInline	<i>(Filled by the method)</i> If true , the Marked Content dictionary is inline; may be NULL .

Return Value

true if dictionary is obtained, **false** if no dictionary is present.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Related Methods

[PDEPlaceSetDict](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEPlaceGetMCTag

```
ASAtom PDEPlaceGetMCTag (PDEPlace pdePlace);
```

Description

Gets the Marked Content tag for a **PDEPlace**.

Parameters

pdePlace	The place whose Marked Content tag is obtained.

Return Value

Tag for **pdePlace**.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Related Methods

[PDEPlaceSetMCTag](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEPlaceSetDict

```
void PDEPlaceSetDict (PDEPlace pdePlace, CosObj* placeDictP,
ASBool isInline);
```

Description

Sets the Marked Content dictionary for a **PDEPlace**.

Parameters

pdePlace	The place whose Marked Content dictionary is set.
placeDictP	Marked Content dictionary for pdePlace .
isInline	If true , the dictionary is emitted inline.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PEWCalls.h

Related Methods

[PDEPlaceGetDict](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEPlaceSetMCTag

```
void PDEPlaceSetMCTag (PDEPlace pdePlace, ASAtom mcTag) ;
```

Description

Sets the Marked Content tag for a **PDEPlace**.

Parameters

pdePlace	The place whose Marked Content tag is set.
mcTag	The tag for pdePlace .

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PEWCalls.h

Related Methods

[PDEPlaceGetMCTag](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEPS

PDEPSCreate

```
PDEPS PDEPSCreate (PDEPSAttrsP attrsP, ASInt32 attrsSize,
ASStm dataStm, ASUns8* data, ASInt32 dataSize);
```

Description

Creates a **PDEPS** object. **data** and **dataStm** may be **NULL**. If so, use **PDEPSSetData** and **PDEPSSetDataStm** to attach data to the object. If **dataStm** is non-**NULL**, then data will be ignored.

See the description of **PDEPSSetDataStm** for how the stream is handled. If **data** is non-**NULL** and **dataStm** is **NULL**, the data must contain **dataSize** number of bytes as specified in the **PDEPSAttrsP**.

The caller must dispose of the stream after calling this method.

Parameters

attrsP	Pointer to PDEPSAttrs attributes data structure.
attrsSize	Size of attributes data structure (sizeof (PDEPSAttrs)).
dataStm	(May be NULL) Data. See above.
data	(May be NULL) Data stream. See above.
dataSize	Number of bytes of data .

Return Value

An object of type **PDEPS**.

Header File

PEWCalls.h

Related Methods

[PDEPSCreateFromCosObj](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDEPSCreateFromCosObj

[PDEPS](#) PDEPSCreateFromCosObj (const [CosObj](#)* cosObjP);

Description

Creates a **PDEPS** object from a **CosObj** object.

Parameters

cosObjP	Object of type CosObj .

Return Value

An object of type [PDEPS](#).

Known Exceptions

[genErrBadParm](#)

Header File

PEWCalls.h

Related Methods

[PDEPSCreate](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDEPSGetAttrs

```
void PDEPSGetAttrs (PDEPS ps, PDEPSAttrsP attrsP,  
ASInt32 attrsSize);
```

Description

Returns a **PDEPS** object's attributes.

Parameters

ps	An object of type PDEPS .
attrsP	<i>(Filled by the method)</i> Pointer to PDEPSAttrs data structure containing the attributes information.
attrsSize	The size of the attrsP buffer (sizeof (PDEPSAttrs)).

Return Value

A pointer to a data structure of type **PDEPSAttrs**.

Header File

PERCalls.h

Related Methods

[PDEPSCreate](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDEPSGetData

```
ASInt32 PDEPSGetData (PDEPS ps, ASUns8* buffer,
ASInt32 bufferSize, ASInt32 offset);
```

Description

Gets all or part of the image data.

Parameters

ps	An object of type PDEPS .
buffer	<i>(Filled by the method)</i> Receives the data.
bufferSize	Size of the buffer.
offset	Offset into the source data at which to start filling buffer.

Return Value

Returns the number of bytes written into the buffer. If the return value is less than **bufferSize**, then there is no more data.

Header File

PERCalls.h

Related Methods

[PDEPSSetData](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDEPSGetDataStm

```
ASStm PDEPSGetDataStm (PDEPS ps);
```

Description

Gets a stream for the data. The data in the stream is decoded (no filters). The caller must dispose of the stream.

Parameters

ps	An object of type PDEPS .

Return Value

An object of type **ASStm**.

Header File

PERCalls.h

Related Methods

[PDEPSSetDataStm](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDEPSSetData

```
void PDEPSSetData (PDEPS ps, ASUns8* buffer,  
ASInt32 bufferSize);
```

Description

Sets the data for an object of type **PDEPS**.

Parameters

ps	An object of type PDEPS .
buffer	Contains the data.
bufferSize	Length of the data in bytes.

Return Value

None

Header File

PEWCalls.h

Related Methods

[PDEPSGetData](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

PDEPSSetDataStm

```
void PDEPSSetDataStm (PDEPS ps, ASStm stm);
```

Description

Sets a stream for the data. The data must be un-encoded (no filters). The caller must dispose of the stream.

Parameters

ps	An object of type PDEPS .
stm	stm is a stream for the data.

Return Value

None

Header File

PEWCalls.h

Related Methods

[PDEPSGetDataStm](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



PDESofMask

PDESofMaskAcquireForm

```
PDEForm PDESofMaskAcquireForm (PDESofMask pdeSoftMask,
ASFixedMatrixP matrixP);
```

Description

Acquire sthe **PDEForm** that defines the soft mask.

Parameters

pdeSoftMask	An object of type PDESofMask .
matrixP	Matrix defining the transformation from coordinate space to user space.

Return Value

The XObject form of the soft mask.

Known Exceptions

```
genErrBadParm
peErrWrongPDEObjectType
```

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDESftMaskCreate

```
PDESftMask PDESftMaskCreate (CosDoc doc,
PDESftMaskCreateFlags type, PDEForm pdeForm) ;
```

Description

Creates a new soft mask object.

Parameters

doc	The container document.
type	Specifies how the mask is to be computed. One of the PDESftMaskCreateFlags.
pdeForm	The form XObject that defines the soft mask. It is the source of the mask values and the PDColorSpace in which the composite computation is to be done.

Return Value

The newly created object.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDESoftMaskCreateFromCosObj

```
PDESoftMask PDESoftMaskCreateFromCosObj  
(const CosObj* cosObjP);
```

Description

Creates a new soft mask object from its Cos representation.

Parameters

<code>cosObjP</code>	The soft mask dictionary.

Return Value

The newly created object.

Header File

`PEWCalls.h`

Product

base, pro, pdf

Availability

Available if `PI_PDFEDIT_WRITE_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.



PDESoftMaskCreateFromName

```
PDESoftMask PDESoftMaskCreateFromName (ASAtom name);
```

Description

Create a new soft mask from a name.

Parameters

name	The new name for the soft mask.
NOTE: Currently, the only valid name is None .	

Return Value

The newly created object.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDESofMaskGetBackdropColor

```
ASInt32 PDESofMaskGetBackdropColor (PDESofMask pdeSoftMask,
ASFixed* pColorValues, ASInt32 numValues);
```

Description

Gets the array of color values of the backdrop color. Given a pointer to an array and the length of the array, copies the color values to that array and returns the number of values copied. If the pointer to the array is **NULL**, the number of color values is returned.

Parameters

pdeSoftMask	An object of type PDESofMask .
pColorValues	(Filled by the method) Pointer to an array of color values. If NULL , the number of color values is returned.
numValues	Length of the array pColorValues .

Return Value

Number of values copied.

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDESoftMaskGetCosObj

```
void PDESoftMaskGetCosObj (PDESoftMask pdeSoftMask,
CosObj* cosObjP);
```

Description

Gets the associated CosObj of the soft mask.

Parameters

pdeSoftMask	The soft mask.
cosObjP	(Filled by the method) A pointer to the Cos object.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDESoftMaskGetName

```
ASAtom PDESoftMaskGetName (PDESoftMask pdeSoftMask);
```

Description

Gets the soft mask name.

Parameters

pdeSoftMask	The soft mask.

Return Value

Soft mask name if it is a name; returns **ASAtomNull** otherwise.

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDESoftMaskGetTransferFunction

```
CosObj PDESoftMaskGetTransferFunction  
(PDESoftMask pdeSoftMask) ;
```

Description

Gets the transfer function as a **CosObj**.

Parameters

pdeSoftMask	The soft mask.

Return Value

The transfer function as a **CosObj**.

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDESofMaskSetBackdropColor

```
void PDESofMaskSetBackdropColor (PDESofMask pdeSoftMask,
    ASFixed* pColorValues, ASInt32 numValues);
```

Description

Sets the backdrop color values.

Parameters

pdeSoftMask	The soft mask object.
pColorValues	An series of color values.
numValues	The number of values pointed to by pColorValues.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDESofMaskSetTransferFunction

```
void PDESofMaskSetTransferFunction (PDESofMask pdeSoftMask,
CosObj cosTransferFunction) ;
```

Description

Sets the transfer function associated with the soft mask.

Parameters

pdeSoftMask	The soft mask object.
cosTransferFunction	The transfer function dictionary.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDESoftMaskSetXGroup

```
void PDESoftMaskSetXGroup (PDESoftMask pdeSoftMask,
PDEForm pdeForm) ;
```

Description

Sets the **PDEForm** that defines the soft mask.

Parameters

pdeSoftMask	The soft mask object.
pdeForm	The form XObject.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDEShading

PDEShadingCreateFromCosObj

```
PDEShading PDEShadingCreateFromCosObj (const CosObj* shadingP,
ASFixedMatrixP matrixP);
```

Description

Creates a smooth shading object.

Parameters

shadingP	The shading dictionary.
matrixP	The location and transformation matrix of the shading object.

Return Value

A smooth shading object.

Known Exceptions

```
peErrUnknownPDEColorSpace
cosErrInvalidObj
cosErrExpectedName
genErrBadParm
```

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEShadingGetCosObj

```
void PDEShadingGetCosObj (PDEShading shading,  
CosObj* cosObjP);
```

Description

Gets the CosObj for a PDEShading.

Parameters

shading	A smooth shading object.
cosObjP	The Cos dictionary corresponding to shading .

Return Value

None

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDEText

PDETextAdd

```
void PDETextAdd (PDEText pdeText, ASUns32 flags,
ASUns32 index, ASUns8* text, ASUns32 textLen, PDEFont font,
PDEGraphicStateP *gstateP, ASUns32 gstateLen,
PDETextStateP tstateP, ASUns32 tstateLen,
ASFixedMatrixP textMatrixP, ASFixedMatrixP strokeMatrixP);
```

Description

Adds a character or a text run to a **PDEText** object.

NOTE: This method does not change the reference count of **pdeText**; however, the reference count of the objects in the **gstateP** are incremented.

Parameters

pdeText	Text object to which a character or text run is added.
flags	A PDETextFlags that specifies what kind of text to add. Must be either: kPDETextChar — for a text character kPDETextRun — for a text run
index	Index after which to add character or text run.
text	Pointer to the characters to add. NOTE: Passing NULL for text can invalidate the text object but will not raise an error. Callers must not pass NULL for this parameter.
textLen	Length of the text, in bytes.
font	Font for the element.
gstateP	Pointer to a PDEGraphicStateP structure with the graphics state for the element.
gstateLen	Length of graphics state for the element.

tstateP	Pointer to a PDETextState structure with text state for the element. NOTE: PDFEdit ignores the wasSetFlags flag of the PDETextState structure, so you must initialize the PDETextState fields.
tstateLen	Length of text state for the element.
textMatrixP	Pointer to ASFixedMatrix that holds the matrix for the element.
strokeMatrixP	Pointer to ASFixedMatrix that holds the matrix for the line width when stroking text. May be NULL . NOTE: Currently this field is not used. (Acrobat 5)

Return Value

None

Known Exceptions

[pdErrBadResMetrics](#)
[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PEWCalls.h

Related Methods

[PDETextIsAtPoint](#)
[PDETextReplaceChars](#)
[PDETextSplitRunAt](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDETextAddItem

```
void PDETextAddItem (PDEText text, ASUns32 addIndex,
PDETextItem textItem);
```

Description

Adds a text item to a text element at a given index position.

Parameters

text	Text object to which the text item is added.
index	Index of the text item in pdeText.
textItem	The text item to add.

Return Value

None

Known Exceptions

- peErrWrongPDEObjectType
- genErrBadParm
- pdErrBadResMetrics

Header File

PEWCalls.h

Related Methods

- PDETextGetItem
- PDETextRemoveItems
- PDETextItemCreate

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



PDETextCreate

```
PDEText PDETextCreate (void);
```

Description

Creates an empty text object.

Call [PDERelease](#) to dispose of the returned text object when finished with it.

Parameters

None

Return Value

An empty text object.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDETextGetAdvance

```
void PDETextGetAdvance (PDEText pdeText, ASUns32 flags,
ASUns32 index, ASFixedPointP advanceP);
```

Description

Gets the advance width of a character or a text element. Advance width is returned in either character space or user space. The advance width is the amount by which the current point advances when the character is drawn.

Advance width may be horizontal or vertical, depending on the writing style. Thus **advanceP** has both a horizontal and vertical component.

Parameters

pdeText	Text object containing a character or text run whose advance width is found.
flags	A PDETextFlags value that specifies whether index refers to the character offset from the beginning of the text object or the index of the text run in the text object. Must be either: ● kPDETextChar — for a text character ● kPDETextRun — for a text run In addition, set the kPDETextPageSpace flag to obtain the advance width in user space. If it is not set, the advance width is in character space. If this flag is not set, this method returns a value that is independent of any sizes, matrices, or scaling, simply adding up the font's raw glyph widths, supplemented only by unscaled character and word spacing. This differs from the behavior of the older function, PDETextGetAdvanceWidth.
index	Index of the character or text run in pdeText .
advanceP	(Filled by the method) Pointer to a ASFixedPoint value indicating the advance width.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm
pdErrBadResMetrics

Header File

PERCalls.h

**Related Methods**[PDETextGetAdvanceWidth](#)**Product**

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDETextGetAdvanceWidth

```
void PDETextGetAdvanceWidth (PDEText pdeText, ASUns32 flags,
ASUns32 index, ASFixedPointP advanceP);
```

Description

Gets the advance width of a character or a text element. Advance width is returned in either character space or user space. The advance width is the amount by which the current point advances when the character is drawn.

Advance width may be horizontal or vertical, depending on the writing style. Thus **advanceP** has both a horizontal and vertical component.

Parameters

pdeText	Text object containing a character or text run whose advance width is found.
flags	A PDETextFlags value that specifies whether index refers to the character offset from the beginning of the text object or the index of the text run in the text object. Must be either: • kPDETextChar — for a text character • kPDETextRun — for a text run In addition, set the kPDETextPageSpace flag to obtain the advance width in user space. If it is not set, the advance width is in character space.
index	Index of the character or text run in pdeText .
advanceP	<i>(Filled by the method)</i> Pointer to a ASFixedPoint value indicating the advance width.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm
pdErrBadResMetrics

Header File

PERCalls.h

Product

base, pro, pdf



Availability

Available if **PI_PDFEDIT_READ_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDETextGetBBox

```
void PDETextGetBBox (PDEText pdeText, ASUns32 flags,
ASUns32 index, ASFixedRectP bboxP);
```

Description

Gets the bounding box of a character or a text run.

Parameters

pdeText	Text object containing a character or text run whose bounding box is found.
flags	A PDETextFlags that specifies whether index refers to the character offset from the beginning of the text object or the index of the text run in the text object. Must be either: kPDETextChar — for a text character kPDETextRun — for a text run
index	Index of the character or text run in pdeText.
bboxP	(Filled by the method) Pointer to ASFixedRect to set to the bounding box of specified character or text run.

Return Value

None

Known Exceptions

- peErrWrongPDEObjectType
- genErrBadParm
- pdErrBadResMetrics

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if PI_PDFEDIT_READ_VERSION (in PIRequir.h) is set to 0x00040000 or higher.

PDETextGetFont

```
PDEFont PDETextGetFont (PDEText pdeText, ASUns32 flags,
ASUns32 index);
```

Description

Gets the font for a text character or element.

NOTE: This method does not change the reference count of the returned **PDEFont**.

Parameters

pdeText	Text object containing a character or text run whose font is found.
flags	A PDETextFlags that specifies whether index refers to the character offset from the beginning of the text object or the index of the text run in the text object. Must be either: kPDETextChar — for a text character kPDETextRun — for a text run
index	Index of the character or text run in pdeText .

Return Value

Font of the specified character or text run.

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm
pdErrBadResMetrics

Header File

PERCalls.h

Related Methods

PDETextRunSetFont

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDETextGetGState

```
void PDETextGetGState (PDEText pdeText, ASUns32 flags,
ASUns32 index, PDEGraphicStateP stateP, ASUns32 stateSize);
```

Description

Gets the graphics state of a character or a text run.

NOTE: This method does not increment the reference count of the objects in stateP.

Parameters

pdeText	Text object containing a character or text run whose graphics state is found.
flags	A PDETextFlags that specifies whether index refers to the character offset from the beginning of the text object or the index of the text run in the text object. Must be either: kPDETextChar — for a text character kPDETextRun — for a text run
index	Index of the character or text run in pdeText.
stateP	(Filled by the method) Pointer to a PDEGraphicStateP structure with graphics state of specified character or text run.
stateSize	Size of the stateP buffer, in bytes.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm

Header File

PERCalls.h

Related Methods

PDETextRunSetGState

Product

base, pro, pdf

Availability

Available if PI_PDFEDIT_READ_VERSION (in PIRequir.h) is set to 0x00040000 or higher.

PDETextGetItem

```
PDETextItem PDETextGetItem (PDEText text, ASUns32 index);
```

Description

Obtains a text item from a text element at a given index position.

Parameters

text	Text object from which the text item is obtained.
index	Index of the text item in pdeText .

Return Value

The text item object.

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)
[pdErrBadResMetrics](#)

Header File

PERCalls.h

Related Methods

[PDETextAddItem](#)
[PDETextItemCreate](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDETextGetMatrix

```
void PDETextGetMatrix (PDEText pdeText, ASUns32 flags,
ASUns32 index, ASFixedMatrixP matrixP);
```

Description

Returns the matrix of a character or a text element. Unlike [PDETextGetTextMatrix](#), this function doesn't take **fontSize**, **hScale**, and **textRise** in the **textState** into account.

Parameters

pdeText	Text object containing a character or text run whose graphics state is found.
flags	A PDETextFlags that specifies whether index refers to the character offset from the beginning of the text object or the index of the text run in the text object. Must be either: kPDETextChar — for a text character kPDETextRun — for a text run
index	Index of the character or text run in pdeText .
matrixP	(Filled by the method) ASFixedMatrixP that holds the matrix of specified character or text run.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PERCalls.h

Related Methods

[PDETextGetTextMatrix](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDETextGetNumBytes

```
ASUns32 PDETextGetNumBytes (PDEText pdeText, ASUns32 flags,
ASUns32 index);
```

Description

Gets the number of bytes occupied by the character code or text run.

Parameters

pdeText	A PDEText object returned from one of the PDETextCreate methods whose text is examined.
flags	A PDETextFlags that specifies whether index refers to the character offset from the beginning of the text object or the index of the text run in the text object. Must be either: kPDETextChar — for a text character kPDETextRun — for a text run
index	Index of the character or text run in pdeText .

Return Value

Number of bytes occupied by the text run or character.

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm

Header File

PERCalls.h

Related Methods

PDEFontGetNumCodeBytes

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDETextGetNumChars

[ASUns32](#) PDETextGetNumChars ([PDEText](#) pdeText) ;

Description

Gets the number of characters in a text object.

Parameters

pdeText	Text object whose number of characters is found.

Return Value

Total number of characters in **pdeText**.

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PERCalls.h

Related Methods

[PDETextGetNumRuns](#)
[PDETextGetRunForChar](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDETextGetNumRuns

```
ASUns32 PDETextGetNumRuns (PDEText pdeText) ;
```

Description

Gets the number of text runs (show strings) in a text object.

Parameters

pdeText	Text object whose number of text runs is found.

Return Value

Number of text runs in **pdeText**.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Related Methods

[PDETextGetNumBytes](#)
[PDETextGetRunForChar](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDETextGetQuad

```
void PDETextGetQuad (PDEText pdeText, ASUns32 flags,
ASUns32 index, ASFixedQuadP quadP);
```

Description

Gets the quad bounding the specified text run or character.

The advance portion of the quad is based on the left side bearing and advance width.

Parameters

pdeText	Text object containing a character or text run whose quad is found.
flags	A PDETextFlags that specifies whether index refers to the character offset from the beginning of the text object or the index of the text run in the text object. Must be either: kPDETextChar — for a text character kPDETextRun — for a text run In addition, if the kPDETextBounding flag is set, PDETextGetQuad uses the font descriptor's FontBBox , which is the smallest rectangle that encloses all characters in the font. The advance portion is based on the x-coordinates of the left and right sides of FontBBox and the advance width.
index	Index of the character or text run in pdeText .
quadP	(Filled by the method) Pointer to ASFixedQuad that bounds the specified character or text run.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm
pdErrBadResMetrics

Header File

PERCalls.h

Product

base, pro, pdf

**Availability**

Available if **PI_PDFEDIT_READ_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDETextGetRunForChar

```
ASUns32 PDETextGetRunForChar (PDEText pdeText ,
ASUns32 charIndex) ;
```

Description

Gets the index of the text run that contains the nth character in a text object.

Parameters

pdeText	Text object to examine.
charIndex	Number of the character to find in pdeText.

Return Value

Index of the text run with the specified character index into pdeText.

Known Exceptions

```
peErrWrongPDEObjectType
genErrBadParm
```

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDETextGetState

```
void PDETextGetState (PDEText pdeText, ASUns32 flags,
ASUns32 index, PDETextStateP stateP, ASUns32 stateSize);
```

Description

Returns the text state of a character or a text element.

Parameters

pdeText	Text object containing a character or text run whose text state is found.
flags	A PDETextFlags that specifies whether index refers to the character offset from the beginning of the text object or the index of the text run in the text object. Must be either: kPDETextChar — for a text character kPDETextRun — for a text run
index	Index of the character or text run in pdeText.
stateP	(Filled by the method) Pointer to a PDETextState structure to fill with the text state of the specified character or text run.
stateSize	Size of the stateP buffer, in bytes.

Return Value

None

Known Exceptions

- genErrBadParm
- peErrWrongPDEObjectType

Header File

PERCalls.h

Related Methods

- PDETextRunSetTextState
- PDETextGetTextState

Product

base, pro, pdf

Availability

Available if PI_PDFEDIT_READ_VERSION (in PIRequir.h) is set to 0x00050000 or higher.

PDETextGetStrokeMatrix

```
void PDETextGetStrokeMatrix (PDEText pdeText, ASUns32 flags,
                             ASUns32 index, ASFixedMatrixP matrixP);
```

Description

Gets the stroke matrix of a character or a text run.

NOTE: Currently this method returns no valid information. (Acrobat 5)

Parameters

pdeText	Text object containing a character or text run whose stroke matrix is found.
flags	A PDETextFlags that specifies whether index refers to the character offset from the beginning of the text object or the index of the text run in the text object. Must be either: kPDETextChar — for a text character kPDETextRun — for a text run
index	Index of the character or text run in pdeText.
matrixP	(Filled by the method) Pointer to ASFixedMatrix that holds the stroke matrix of specified character or text run. This matrix is the transformation for line widths when stroking. The h and v values of the matrix are ignored.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm

Header File

PERCalls.h

Related Methods

PDETextRunSetStrokeMatrix

Product

base, pro, pdf

Availability

Available if PI_PDFEDIT_READ_VERSION (in PIRequir.h) is set to 0x00040000 or higher.

PDETextGetText

```
ASUns32 PDETextGetText (PDEText pdeText, ASUns32 flags,
ASUns32 index, ASUns8* textBuffer);
```

Description

Gets the text for a text run or character.

Parameters

pdeText	Text object containing a character or text run whose text is found.
flags	A PDETextFlags that specifies whether index refers to the character offset from the beginning of the text object or the index of the text run in the text object. Must be either: kPDETextChar — for a text character kPDETextRun — for a text run
index	Index of the character or text run in pdeText .
textBuffer	<i>(Filled by the method)</i> Text of specified character or text run. textBuffer must be large enough to hold the returned text. If textBuffer is NULL , returns the number of bytes required to hold the data.

Return Value

Number of bytes in text run or character.

Known Exceptions

genErrBadParm
peErrWrongPDEObjectType

Header File

PERCalls.h

Related Methods

PDETextGetTextMatrix
PDETextGetTextState

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDETextGetTextMatrix

```
void PDETextGetTextMatrix (PDEText pdeText, ASUns32 flags,
ASUns32 index, ASFixedMatrixP matrixP);
```

Description

Gets the matrix of a character or a text run.

Parameters

pdeText	Text object containing a character or text run whose matrix is found.
flags	A PDETextFlags that specifies whether index refers to the character offset from the beginning of the text object or the index of the text run in the text object. Must be either: kPDETextChar — for a text character kPDETextRun — for a text run
index	Index of the character or text run in pdeText.
matrixP	(Filled by the method) Pointer to ASFixedMatrix that holds the matrix of specified character or text run. This is the transformation matrix from user space to the current text space. The h and v values of the matrix indicate the origin of the first character.

Return Value

None

Known Exceptions

- peErrWrongPDEObjectType
- genErrBadParm
- pdErrBadResMetrics

Header File

PERCalls.h

Related Methods

- PDETextRunSetTextMatrix
- PDETextGetMatrix

Product

base, pro, pdf

Availability

Available if PI_PDFEDIT_READ_VERSION (in PIRequir.h) is set to 0x00040000 or higher.

PDETextGetTextState

```
void PDETextGetTextState (PDEText pdeText, ASUns32 flags,
ASUns32 index, PDETextStateP stateP, ASUns32 stateSize);
```

Description

Gets the text state of a character or a text element.

NOTE: This function handles only **charSpacing**, **wordSpacing**, and **renderMode** for backward compatibility. For all attributes, use **PDETextGetState** instead.

Parameters

pdeText	Text object containing a character or text run whose text state is found.
flags	A PDETextFlags that specifies whether index refers to the character offset from the beginning of the text object or the index of the text run in the text object. Must be either: kPDETextChar — for a text character kPDETextRun — for a text run
index	Index of the character or text run in pdeText .
stateP	(Filled by the method) Pointer to a PDETextState structure to fill with the text state of the specified character or text run.
stateSize	Size of the stateP buffer, in bytes.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm

Header File

PERCalls.h

Related Methods

PDETextGetState
PDETextRunSetTextState

Product

base, pro, pdf

**Availability**

Available if **PI_PDFEDIT_READ_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDETextIsAtPoint

```
ASBool PDETextIsAtPoint (PDEText pdeText, ASUns32 flags,
ASUns32 index, ASFixedPoint point);
```

Description

Tests whether a point is on specified text. Checks if the point is in a bounding box for the **PDEText**.

Parameters

pdeText	The text to test.
flags	A PDETextFlags that specifies whether index refers to the character offset from the beginning of the text object or the index of the text run in the text object. Must be either: kPDETextChar — for a text character kPDETextRun — for a text run
index	Index of the character or text run in pdeText .
point	The point, specified in user space coordinates.

Return Value

true if the point is on the text, **false** otherwise.

Header File

PERCalls.h

Related Methods

[PDEElementIsAtPoint](#)
[PDEElementIsAtRect](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDETextIsAtRect

```
ASBool PDETextIsAtRect (PDEText pdeText, ASUns32 flags,
ASUns32 index, ASFixedRect rect);
```

Description

Tests whether any part of a rectangle is on the specified text.

Parameters

pdeText	The text to test.
flags	A PDETextFlags flag that specifies whether index refers to the character offset from the beginning of the text object or the index of the text run in the text object. Must be either: kPDETextChar — for a text character kPDETextRun — for a text run
index	Index of the character or text run in pdeText .
rect	The rectangle, specified in user space coordinates.

Return Value

true if the text is on the rectangle, **false** otherwise.

Header File

PERCalls.h

Related Methods

[PDEElementIsAtPoint](#)
[PDEElementIsAtRect](#)
[PDETextIsAtPoint](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDETextRemove

```
void PDETextRemove (PDEText pdeText, ASUns32 flags,  
ASUns32 index, ASUns32 count);
```

Description

Removes characters or text runs from a text object.

NOTE: This method decrements the reference count of **objects associated with the pdeText in the graphic state and font.**

Parameters

pdeText	Text object from which text is removed.
flags	A PDETextFlags that specifies whether index refers to the character offset from the beginning of the text object or the index of the text run in the text object. Must be either: kPDETextChar — for a text character kPDETextRun — for a text run
index	Index of the character or text run in pdeText .
count	Number of characters or text runs to remove.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)
[pdErrBadResMetrics](#)

Header File

PEWCalls.h

Related Methods

[PDETextAdd](#)
[PDETextReplaceChars](#)
[PDETextSplitRunAt](#)

Product

base, pro, pdf

**Availability**

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.



PDETextRemoveItems

```
void PDETextRemoveItems (PDEText text, ASUns32 index,  
ASUns32 count);
```

Description

Removes contiguous text items from a text element starting at a given index position.

Parameters

text	Text object from which the text items are removed.
index	Index of the first text item in pdeText to remove.
textItem	The number of text items to remove.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)
[pdErrBadResMetrics](#)

Header File

PEWCalls.h

Related Methods

[PDETextAddItem](#)
[PDETextGetItem](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDETextReplaceChars

```
void PDETextReplaceChars (PDEText pdeText, ASUns32 flags,
ASUns32 index, ASUns8* textBuffer, ASUns32 numBytes);
```

Description

Replaces characters in a text object.

This method does not change the number of characters in the text object—extra characters are ignored.

Parameters

pdeText	Text object in which characters are replaced.
flags	A PDETextFlags that specifies whether index refers to the character offset from the beginning of the text object or the index of the text run in the text object. Must be either: kPDETextChar — for a text character kPDETextRun — for a text run
index	Index of the character or text run in pdeText .
textBuffer	Replacement text.
numBytes	Number of bytes to replace.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PEWCalls.h

Related Methods

[PDETextAdd](#)
[PDETextIsAtPoint](#)
[PDETextSplitRunAt](#)

Product

base, pro, pdf



Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDETextRunGetCharOffset

```
ASUns32 PDETextRunGetCharOffset (PDEText pdeText,
ASUns32 runIndex);
```

Description

Gets the character offset of the first character of the specified text run.

Parameters

pdeText	Text object containing a character or text run whose graphics state is found.
runIndex	Index of the text run whose first character's index is returned.

Return Value

Character offset of the first character of the specified text run in **pdeText**.

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PERCalls.h

Related Methods

[PDETextGetNumBytes](#)
[PDETextGetNumRuns](#)
[PDETextGetRunForChar](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDETextRunGetNumChars

```
ASUns32 PDETextRunGetNumChars (PDEText pdeText,
ASUns32 runIndex);
```

Description

Gets the number of characters in a text run.

Parameters

pdeText	Text object containing a text run whose number of characters is found.
runIndex	Index of the text run whose number of characters is returned.

Return Value

Number of characters in the specified text run.

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PERCalls.h

Related Methods

[PDETextGetNumRuns](#)
[PDETextGetRunForChar](#)
[PDETextRunGetCharOffset](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDETextRunSetFont

```
void PDETextRunSetFont (PDEText pdeText, ASUns32 runIndex,
PDEFont font);
```

Description

Sets the font of a text run.

NOTE: This method decrements the reference count of the previous font and increments the reference count of the new font.

Parameters

pdeText	Text object containing a text run whose font is set.
runIndex	Index of the text run.
font	Font set for the text run.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm

Header File

PEWCalls.h

Related Methods

PDETextGetFont

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDETextRunSetGState

```
void PDETextRunSetGState (PDEText pdeText, ASUns32 runIndex,  
PDEGraphicStateP stateP, ASUns32 stateSize);
```

Description

Sets the graphics state of a text run.

NOTE: This method increments the reference count of objects in the **stateP**.

Parameters

pdeText	Text object containing a text run whose graphics state is set.
runIndex	Index of the text run.
stateP	Pointer to a PDEGraphicStateP structure with graphics state to set.
stateSize	Size of the stateP buffer, in bytes.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm

Header File

PEWCalls.h

Related Methods

PDETextGetGState

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDETextRunSetMatrix

```
void PDETextRunSetMatrix (PDEText pdeText, ASUns32 runIndex,
ASFixedMatrixP matrixP);
```

Description

Sets the matrix of a text run. Unlike [PDETextRunSetTextMatrix](#), this function doesn't change **fontSize**, **hScale**, and **textRise** in the **textState** of **PDEText**.

Parameters

pdeText	Text object containing a text run.
runIndex	Index of the text run.
matrixP	ASFixedMatrixP pointer.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PEWCalls.h

Related Methods

[PDETextRunSetTextMatrix](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDETextRunSetState

```
void PDETextRunSetState (PDEText pdeText, ASUns32 runIndex,
PDETextStateP stateP, ASUns32 stateSize);
```

Description

Sets the text state of a text run.

Parameters

pdeText	Text object containing a text run whose state is set.
runIndex	Index of the text run.
stateP	Pointer to a PDETextState structure with state to set.
stateSize	Size of the stateP buffer, in bytes.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm

Header File

PEWCalls.h

Related Methods

PDETextRunSetTextState

Product

base, pro, pdf

Availability

Available if PI_PDFEDIT_WRITE_VERSION (in PIRequir.h) is set to 0x00050000 or higher.

PDETextRunSetStrokeMatrix

```
void PDETextRunSetStrokeMatrix (PDEText pdeText,
ASUns32 runIndex, ASFixedMatrixP matrixP);
```

Description

Sets the stroke matrix of a text run.

NOTE: Currently this method is not implemented. (Acrobat 5)

Parameters

pdeText	Text object containing a text run whose stroke matrix is set.
runIndex	Index of the text run.
matrixP	Pointer to ASFixedMatrix that holds the stroke matrix.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PEWCalls.h

Related Methods

[PDETextGetStrokeMatrix](#)

Product

base, pro, pdf

Availability

Available if `PI_PDFEDIT_WRITE_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.

PDETextRunSetTextMatrix

```
void PDETextRunSetTextMatrix (PDEText pdeText,
ASUns32 runIndex, ASFixedMatrixP matrixP);
```

Description

Sets the text matrix of a text run.

Parameters

pdeText	Text object containing a text run whose text matrix is set.
runIndex	Index of the text run.
matrixP	Pointer to ASFixedMatrix that holds the text matrix.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm

Header File

PEWCalls.h

Related Methods

PDETextGetTextMatrix
PDETextRunSetMatrix

Product

base, pro, pdf

Availability

Available if PI_PDFEDIT_WRITE_VERSION (in PIRequir.h) is set to 0x00040000 or higher.

PDETextRunSetTextState

```
void PDETextRunSetTextState (PDEText pdeText,
ASUns32 runIndex, PDETextStateP stateP, ASUns32 stateSize);
```

Description

Sets the text state of a text run.

NOTE: This method has the following side effect (Acrobat 5): It modifies the text matrix of the run. In order to maintain backward compatibility, this method only directly operates on the first four fields of [PDETextState](#). When it is called, it calculates a new text matrix with three additional fields—**fontSize**, **hScale**, and **textRise** (see [PDETextState](#)). To avoid this behavior, use [PDETextRunSetState](#) instead (which was added as an API to address this problem).

Parameters

pdeText	Text object containing a text run whose text state is set.
runIndex	Index of the text run.
stateP	Pointer to a PDETextState structure with text state.
stateSize	Size of the stateP buffer, in bytes.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PEWCalls.h

Related Methods

[PDETextGetTextState](#)
[PDETextRunSetState](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in [PIRequir.h](#)) is set to **0x00040000** or higher.



PDETextSplitRunAt

```
void PDETextSplitRunAt (PDEText pdeText, ASUns32 splitLoc);
```

Description

Splits a text run into two text runs.

Parameters

pdeText	Text object containing a text run to split.
splitLoc	Split location, relative to the text object. The first text run is from character index 0 up to splitLoc . The second text run is from splitLoc + 1 to the end of the run.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)
[pdErrBadResMetrics](#)

Header File

PEWCalls.h

Related Methods

[PDETextIsAtPoint](#)
[PDETextReplaceChars](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDETextItem

PDETextItemCopyText

```
ASUns32 PDETextItemCopyText (PDETextItem textItem,
ASUns8 *buffer, ASUns32 bufferSize);
```

Description

Copies the text from a text item element into a character buffer.

Parameters

textItem	Pointer to the characters to add. NOTE: Passing NULL for text can invalidate the text object but will not raise an error. Callers must not pass NULL for this parameter.
buffer	<i>(Filled by the method)</i> A pointer to a buffer in which to store the copy.
bufferLen	Length of the text buffer, in bytes.

Return Value

The length in bytes of **textItem**.

Known Exceptions

[pdErrBadResMetrics](#)
[peErrWrongPDEObjectType](#)
[genErrBadParm](#)

Header File

PERCalls.h

Related Methods

[PDETextGetItem](#)
[PDETextAddItem](#)
[PDETextItemGetTextLen](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDETextItemCreate

```
PDETextItem PDETextItemCreate (ASUns8* text, ASUns32 textLen,
PDEFont font, PDEGraphicStateP gstateP, ASUns32 gstateLen,
PDETextStateP tstateP, ASUns32 tstateLen,
ASFixedMatrixP textMatrixP);
```

Description

Creates a text item element containing a character or text run, which can be added to a **PDEText** object.

Parameters

text	Pointer to the characters to add. NOTE: Passing NULL for text can invalidate the text object but will not raise an error. Callers must not pass NULL for this parameter.
textLen	Length of the text, in bytes.
font	Font for the element.
gstateP	Pointer to a PDEGraphicStateP structure with the graphics state for the element.
gstateLen	Length of graphics state for the element.
tstateP	Pointer to a PDETextState structure with text state for the element. NOTE: PDFEdit ignores the wasSetFlags flag of the PDETextState structure, so you must initialize the PDETextState fields.
tstateLen	Length of text state for the element.
textMatrixP	Pointer to ASFixedMatrix that holds the matrix for the element.

Return Value

None

Known Exceptions

```
pdErrBadResMetrics
peErrWrongPDEObjectType
genErrBadParm
```

**Header File**

PEWCalls.h

Related Methods

[PDETextAdd](#)

[PDETextAddItem](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDETextItemGetFont

```
PDEFont PDETextItemGetFont (PDETextItem textItem);
```

Description

Gets the font for a text item.

NOTE: This method does not change the reference count of the returned **PDEFont**.

Parameters

textItem	Text item whose font is obtained.

Return Value

Font of the specified text item.

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)
[pdErrBadResMetrics](#)

Header File

PERCalls.h

Related Methods

[PDETextItemSetFont](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDETextItemGetGState

```
void PDETextItemGetGState (PDETextItem textItem,  
    PDEGraphicStateP textStateP, ASUns32 textStateSize);
```

Description

Gets the graphics state for a text item.

Parameters

textItem	Text item whose graphic state is obtained.
textStateP	(Filled by the method) Pointer to a PDEGraphicStateP structure with graphics state of the text item.
textStateSize	Size of the stateP buffer, in bytes.

Return Value

None

Known Exceptions

- peErrWrongPDEObjectType
- genErrBadParm
- pdErrBadResMetrics

Header File

PERCalls.h

Related Methods

PDETextItemSetGState

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



PDETextItemGetTextLen

```
ASInt32 PDETextItemGetTextLen (PDETextItem textItem);
```

Description

Gets the text length for a text item.

Parameters

textItem	Text item whose text length is obtained.

Return Value

The text length, in bytes.

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)
[pdErrBadResMetrics](#)

Header File

PERCalls.h

Related Methods

[PDETextItemCopyText](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDETextItemGetTextMatrix

```
void PDETextItemGetTextMatrix (PDETextItem textItem,  
ASUns32 charOffset, ASFixedMatrix *textMatrixP);
```

Description

Gets the text matrix for a character in a text item.

Parameters

textItem	The text item.
charOffset	The offset of the character whose text matrix is obtained.
textMatrixP	(Filled by the method) Pointer to a ASFixedMatrix structure with the text matrix of the character.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)
[pdErrBadResMetrics](#)

Header File

PERCalls.h

Related Methods

[PDETextItemSetTextMatrix](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDETextItemGetTextState

```
void PDETextItemGetTextState (PDETextItem textItem,  
                              PDTextStateP *textStateP, ASUns32 textStateSize);
```

Description

Gets the text state of a text item.

Parameters

textItem	The text item whose text state is obtained.
textStateP	(Filled by the method) Pointer to a PDTextStateP structure with text state of the text item.
textStateSize	Size of the texStateP structure, in bytes.

Return Value

None

Known Exceptions

- peErrWrongPDEObjectType
- genErrBadParm
- pdErrBadResMetrics

Header File

PERCalls.h

Related Methods

PDETextItemSetTextState

Product

base, pro, pdf

Availability

Available if PI_PDFEDIT_READ_VERSION (in PIRequir.h) is set to 0x00060000 or higher.

PDETextItemRemoveChars

```
void PDETextItemRemoveChars (PDETextItem textItem,
ASUns32 charOffset, ASUns32 count);
```

Description

Removes contiguous characters from a text item.

Parameters

textItem	The text item whose characters are removed.
charOffset	Offset of the first character to remove.
count	The number of characters to remove.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType
genErrBadParm
pdErrBadResMetrics

Header File

PEWCalls.h

Related Methods

PDETextItemReplaceChars
PDETextItemReplaceText

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDETextItemReplaceChars

```
void PDETextItemReplaceChars (PDETextItem textItem,
ASUns32 charIndex, ASUns8 *newChar, ASUns32 newCharLen);
```

Description

Replaces characters in a text item.

This method does not change the number of characters in the text item—extra characters are ignored.

Parameters

textItem	The text item whose characters are replaced.
charIndex	Index position of the characters to replace.
newChar	Replacement text.
newCharLen	Number of bytes to replace.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)
[pdErrBadResMetrics](#)

Header File

PEWCalls.h

Related Methods

[PDETextItemRemoveChars](#)
[PDETextItemReplaceText](#)

Product

base, pro, pdf

Availability

Available if `PI_PDFEDIT_WRITE_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

PDETextItemReplaceText

```
void PDETextItemReplaceText (PDETextItem textItem,  
ASUns8 *newText, ASUns32 newTextLen);
```

Description

Replaces all of the text in a text item.

Parameters

textItem	The text item whose text are replaced.
newText	Replacement text.
newTextLen	Number of bytes to replace.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)
[pdErrBadResMetrics](#)

Header File

PEWCalls.h

Related Methods

[PDETextItemRemoveChars](#)
[PDETextItemReplaceChars](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDETextItemSetFont

```
void PDETextItemSetFont (PDETextItem textItem, PDEFont font);
```

Description

Sets the font for a text item.

NOTE: This method decrements the reference count of the previous font and increments the reference count of the new font.

Parameters

textItem	Text item whose font is set.
font	The new font object.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)
[pdErrBadResMetrics](#)

Header File

PEWCalls.h

Related Methods

[PDETextItemGetFont](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDETextItemSetGState

```
void PDETextItemSetGState (PDETextItem textItem,  
    PDEGraphicStateP textStateP, ASUns32 textStateSize);
```

Description

Sets the graphics state for a text item.

Parameters

textItem	Text item whose graphics state is set.
textStateP	Pointer to a PDEGraphicStateP structure with graphics state of the text item.
textStateSize	Size of the stateP buffer, in bytes.

Return Value

None

Known Exceptions

- peErrWrongPDEObjectType
- genErrBadParm
- pdErrBadResMetrics

Header File

PEWCalls.h

Related Methods

PDETextItemGetGState

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDETextItemSetTextMatrix

```
void PDETextItemSetTextMatrix (PDETextItem textItem,
ASFixedMatrix *textMatrixP);
```

Description

Sets the text matrix for a text item.

Parameters

textItem	The text item whose text matrix is set.
textMatrixP	Pointer to a ASFixedMatrix structure with the new text matrix of the text item.

Return Value

None

Known Exceptions

[peErrWrongPDEObjectType](#)
[genErrBadParm](#)
[pdErrBadResMetrics](#)

Header File

PEWCalls.h

Related Methods

[PDETextItemGetTextMatrix](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDETextItemSetTextState

```
void PDETextItemSetTextState (PDETextItem textItem,  
                              PDTextStateP textStateP, ASUns32 textStateSize);
```

Description

Sets the text state for a text item.

Parameters

textItem	The text item whose text state is set.
textStateP	A PDETextStateP structure with the new text state of the text item.
textStateSize	Size of the texStateP structure, in bytes.

Return Value

None

Known Exceptions

- peErrWrongPDEObjectType
- genErrBadParm
- pdErrBadResMetrics

Header File

PEWCalls.h

Related Methods

PDETextItemGetTextState

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



PDEUnknown

PDEUnknownGetOpName

```
ASAtom PDEUnknownGetOpName (PDEUnknown pdeUnknown);
```

Description

Gets the operator name of an unknown operator.

Parameters

pdeUnknown	Unknown element whose operator name is obtained.

Return Value

An **ASAtom** for the name of the operator for **pdeUnknown**.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEXGroup

PDEXGroupAcquireColorSpace

```
PDEColorSpace PDEXGroupAcquireColorSpace  
(PDEXObject pdeXGroup) ;
```

Description

Acquires the color space of the transparency group.

Parameters

pdeXGroup	The transparency group object.

Return Value

The color space; otherwise **NULL**.

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEXGroupCreate

```
PDEXGroup PDEXGroupCreate (CosDoc doc,
PDEXGroupCreateFlags type);
```

Description

Create a new XGroup of the given type.

Parameters

doc	The document the object will be created in.
type	Must be kPDEXGroupTypeTransparency.

Return Value

The newly created transparency group object.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEXGroupCreateFromCosObj

```
PDEXGroup PDEXGroupCreateFromCosObj (const CosObj* cosObjP);
```

Description

Creates a new XGroup object from its Cos representation.

Parameters

cosObjP	The XGroup object dictionary.

Return Value

The PDEXGroup object.

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



PDEXGroupGetCosObj

```
void PDEXGroupGetCosObj (PDEXGroup pdeXGroup,
CosObj* cosObjP);
```

Description

Gets the CosObj of the transparency group.

Parameters

pdeXGroup	The transparency group object.
cosObjP	(Filled by the method) Pointer to the Cos object.

Return Value

None

Known Exceptions

- genErrBadParm
- peErrWrongPDEObjectType

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEXGroupGetIsolated

```
ASBool PDEXGroupGetIsolated (PDEXObject pdeXGroup);
```

Description

Gets the isolated boolean value of the transparency group.

Parameters

pdeXGroup	The transparency group object.

Return Value

true if the transparency group is isolated; **false** otherwise.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEXGroupGetKnockout

```
ASBool PDEXGroupGetKnockout (PDEXObject pdeXGroup);
```

Description

Gets the knockout boolean value of the transparency group.

Parameters

pdeXGroup	The transparency group object.

Return Value

The knockout value.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDEXGroupSetColorSpace

```
void PDEXGroupSetColorSpace (PDEXObject pdeXGroup,  
                             PDEColorSpace pdeColorSpace);
```

Description

Sets the [PDEXObject](#) that defines the color space into which colors are converted when painted into this group.

Parameters

pdeXGroup	The transparency group object.
pdeColorSpace	The color space to associate with the XGroup.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.



PDEXGroupSetIsolated

```
void PDEXGroupSetIsolated (PDEXGroup pdeXGroup,
ASBool isolated);
```

Description

Sets the XGroup to be isolated or not. Corresponds to the **/I** key within the XGroup’s dictionary.

Parameters

pdeXGroup	The transparency group object.
isolated	true to isolate the XGroup, false otherwise.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEXGroupSetKnockout

```
void PDEXGroupSetKnockout (PDEXObject pdeXGroup,
    ASBool knockout) ;
```

Description

Sets the knockout value.

Parameters

pdeXGroup	The transparency group object.
knockout	The knockout value.

Return Value

None

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDEXObject

PDEXObjectCreate

```
PDEXObject PDEXObjectCreate (const CosObj* cosObjP);
```

Description

Creates a new **PDEXObject** from a Cos object.

Call **PDERelease** to dispose of the returned **PDEXObject** when finished with it.

Parameters

cosObjP	Cos object for the PDEXObject .

Return Value

PDEXObject corresponding to the **cosObjP**.

Header File

PEWCalls.h

Related Methods

[PDEXObjectGetCosObj](#)

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDEXObjectGetCosObj

```
void PDEXObjectGetCosObj (PDEXObject xObject,  
CosObj* cosObjP);
```

Description

Gets a Cos object corresponding to a PDEXObject.

Parameters

xObject	The PDEXObject whose Cos object is obtained.
cosObjP	(Filled by the method) Cos object for xObject.

Return Value

None

Known Exceptions

peErrWrongPDEObjectType

Header File

PERCalls.h

Related Methods

PDEXObjectCreate

Product

base, pro, pdf

Availability

Available if PI_PDFEDIT_READ_VERSION (in PIRequir.h) is set to 0x00040000 or higher.



PDSysEncoding

PDSysEncodingCreateFromBaseName

```
PDSysEncoding PDSysEncodingCreateFromBaseName  
(ASAtom baseEncName, const char** diffEnc);
```

Description

Create an encoding object from base name.

Parameters

baseEncName	The base encoding. See Section 5.5.5 in the PDF Reference .
diffEnc	Array of 256 const char* describing the differences from the encoding specified by baseEncName . May be NULL .

Return Value

An object of type **PDSysEncoding**.

Known Exceptions

[genErrBadParm](#)

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDSysEncodingCreateFromCMapName

```
PDSysEncoding PDSysEncodingCreateFromCMapName  
(ASAtom cmapName) ;
```

Description

Create an encoding object from a PDF **CMap** name.

Parameters

cmapName	The CMap name.

Return Value

An object of type **PDSysEncoding**.

Known Exceptions

[genErrBadParm](#)

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDSEncodingCreateFromCodePage

```
PDSEncoding PDSEncodingCreateFromCodePage
(ASInt32 codePage, ASInt16 wMode);
```

Description

Create an encoding object from a code page.

Parameters

codePage	The code page character-mapping construct. See Code Page Values .
wMode	0 for horizontal writing, 1 for vertical writiing.

Return Value

An object of type PDSEncoding.

Known Exceptions

[genErrBadParm](#)

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if `PI_PDFEDIT_WRITE_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

PDSysEncodingGetWMode

```
ASInt16 PDSysEncodingGetWMode (PDSysEncoding sysEnc);
```

Description

Returns writing mode. 0 for horizontal writing and 1 for vertical writing.

Parameters

sysEnc	An object of type PDSysEncoding .

Return Value

0 for horizontal writing and 1 for vertical writing.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDSysEncodingIsIdentity

```
ASBool PDSysEncodingIsIdentity (PDSysEncoding sysEnc);
```

Description

Returns **true** for Identity-H or Identity-V encoding; **false** otherwise.

Parameters

sysEnc	An object of type PDSysEncoding .

Return Value

See above.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDSysEncodingIsMultiByte

```
ASBool PDSysEncodingIsMultiByte (PDSysEncoding sysEnc);
```

Description

Returns **true** for CMap encoding; **false** otherwise.

Parameters

sysEnc	An object of type PDSysEncoding .

Return Value

See above.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PERCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDSysFont

PDEmbedSysFontForPDEFont

```
void PDEmbedSysFontForPDEFont (PDEFont font, ASInt32 flags,
CosDoc cosDoc);
```

Description

If there is a font on the system that matches this **PDEFont**, embed the full font, regardless of whether it was subsetted or not embedded at all in the first place. This will not work for CID fonts, because they must be subsetted.

The matching is based on the **PDSysFontMatchFlags**.

Only the font object itself is modified—no content streams are changed.

NOTE: This method does not change the reference count of the font.

Parameters

font	A PDEFont object returned from one of the PDEFontCreate methods.
flags	Flags from PDSysFontMatchFlags that determine matches.
cosDoc	Currently unused.

Known Exceptions

Raises **peErrFontToEmbedNotOnSys** if there is no system font that matches this **PDEFont**.

Raises **genErrBadParm** if the **PDEFont** is a CID font.

peErrCantCreateFontSubset

peErrCantGetAttrs

peErrCantGetWidths

Header File

PSFCalls.h

Related Methods

PDEFontCreateFromSysFont

PDFindSysFontForPDEFont

Product

base, pro, pdf

**Availability**

Available if **PI_PDSYSFONT_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDEnumSysFonts

```
void PDEnumSysFonts (PDSysFontEnumProc enumProc,
void* clientData);
```

Description

Enumerates all of the system fonts with a user-supplied procedure.

The **PDSysFont** must be acquired during the enumeration if the font is needed beyond the **enumProc**.

Developers should *not* assume that the **enumProc** will get called. If no system fonts are found (for example, if the **PSRESOURCEPATH** environment variable is not set on UNIX platforms), **enumProc** is never called, and **PDEnumSysFonts** does not raise an exception.

NOTE: The font names that are returned from the methods **PDEnumSysFonts** and **PDSysFontsGetAttrs** are different in 5.0 (compared to 4.05). The differences are shown in the table below.

Acrobat 4.05		Acrobat 5.0	
Name	PSname	Name	PSname
MS-Mincho	NULL	MSMincho	MS-Mincho
MS-Gothic	NULL	MSGothic	MS-Gothic
MS-PMincho	NULL	MSPMincho	MS-PMincho
MS-PGothic	NULL	MSPGothic	MS-PGothic
MS-UIGothic	NULL	MSUIGothic	MS-UIGothic

Parameters

enumProc	User-supplied callback to call once for each system font. Enumeration continues until all fonts have been enumerated, or until enumProc returns false .
clientData	Pointer to user-supplied data to pass to enumProc each time it is called.

Return Value

None

Header File

PSFCalls.h

**Related Methods**[PDFindSysFont](#)[PDFindSysFontForPDEFont](#)**Product**

base, pro, pdf

Availability

Available if `PI_PDSYSFONT_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.



PDFindSysFont

```
PDFSysFont PDFindSysFont (PDEFontAttrsP attrs,  
ASInt32 attrSize, ASInt32 flags);
```

Description

Finds a system font that matches the requested attributes.

The method gets the **PDFSysFont** rather than acquires it, so do *not* call **PDFRelease** on the returned **PDFSysFont** when done with it.

Parameters

attrs	Pointer to a PDEFontAttrs structure with the attributes of the font you are searching for.
attrSize	Size of the attrs buffer, in bytes.
flags	Flags from PDFSysFontMatchFlags .

Return Value

The desired system font.

Header File

PSFCalls.h

Related Methods

[PDEnumSysFonts](#)
[PDFindSysFontForPDEFont](#)
[PDFindSysFontEx](#)

Product

base, pro, pdf

Availability

Available if **PI_PDSYSFONT_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDFindSysFontEx

```
PDFont PDFindSysFontEx (PDEFontAttrsP attrs,
ASInt32 attrsSize, ASInt32 flags, ASFixed* mmDesignVector,
ASInt32* designVecLength);
```

Description

Finds a system font that matches the requested attributes.

If the requested font is a multiple master font instance, the base font is returned, and the specified design vector is decoded and returned in **mmDesignVector**.

The method gets the **PDFont** rather than acquires it, so do *not* call **PDFRelease** on the returned **PDFont** when done with it.

Parameters

attrs	Pointer to a PDEFontAttrs structure with the attributes of the font you are searching for.
attrsSize	Size of the attrs buffer, in bytes.
flags	Flags from PDFontMatchFlags .
mmDesignVector	<i>(Filled by the method)</i> If the requested font is a multiple master font instance, the specified design vector is decoded and returned in mmDesignVector .
designVecLength	<i>(Filled by the method)</i> Pass the length of mmDesignVector . This parameter also returns the number of elements filled in mmDesignVector (maximum = 4).

Return Value

The desired system font.

Header File

PSFCalls.h

Related Methods

[PDEnumSysFonts](#)
[PDFindSysFont](#)
[PDFindSysFontForPDEFont](#)

Product

base, pro, pdf

**Availability**

Available if **PI_PDSYSFONT_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDFindSysFontForPDEFont

```
PDSysFont PDFindSysFontForPDEFont (PDEFont font,  
ASInt32 flags);
```

Description

Find a system font that matches the requested **PDEFont**.

The method gets the **PDSysFont** rather than acquires it, so do *not* call **PDERelease** on the returned **PDSysFont** when done with it.

Parameters

font	A PDEFont whose matching system font is found.
flags	Bit field comprised of PDSysFontMatchFlags values. • kPDSysFontMatchNameAndCharSet • kPDSysFontMatchFontType • PDSysFontMatchFlags Passing zero matches font by name only.

Return Value

The system font corresponding to **font**.

Known Exceptions

```
peErrCantGetAttrs  
genErrBadParm  
genErrResourceLoadFailed
```

Header File

```
PSFCalls.h
```

Related Methods

```
PDFindSysFont  
PDEFontGetSysFont
```

Product

base, pro, pdf

Availability

Available if **PI_PDSYSFONT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDSysFontAcquirePlatformData

```
PDSysFontPlatDataP PDSysFontAcquirePlatformData
(PDSysFont sysFont);
```

Description

Acquires platform-specific data for use by user interface code. Must be released when finished by [PDSysFontReleasePlatformData](#).

Parameters

sysFont	A PDSysFont object referencing a system font returned by either PDFindSysFont or PDFindSysFontForPDEFont .

Return Value

Pointer to a platform-dependent structure **PDSysFontPlatData** containing information relating to a system font. Returns **NULL** if out of memory.

Header File

PSFCalls.h

Related Methods

[PDSysFontReleasePlatformData](#)

Product

base, pro, pdf

Availability

Available if **PI_PDSYSFONT_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSysFontGetAttrs

```
void PDSysFontGetAttrs (PDSysFont sysFont,
PDEFontAttrsP attrsP, ASInt32 attrsSize);
```

Description

Gets the attributes of a system font.

The attributes will be returned in the buffer pointed to by **attrsP**.

No more than **attrsSize** bytes will be written to the buffer.

This call can be expensive to execute, as it may involve parsing the font in order to determine attributes.

NOTE: The font names that are returned from the methods **PDEnumSysFonts** and **PDSysFontsGetAttrs** are different in 5.0 (compared to 4.05). The differences are shown in the table below.

Acrobat 4.05		Acrobat 5.0	
Name	PSname	Name	PSname
MS-Mincho	NULL	MSMincho	MS-Mincho
MS-Gothic	NULL	MSGothic	MS-Gothic
MS-PMincho	NULL	MSPMincho	MS-PMincho
MS-PGothic	NULL	MSPGothic	MS-PGothic
MS-UIGothic	NULL	MSUIGothic	MS-UIGothic

Parameters

sysFont	A PDSysFont object referencing a system font whose attributes are obtained.
attrsP	(Filled by the method) Pointer to a PDEFontAttrs structure with the attributes of a system font.
attrsSize	Size of the attrsP buffer, in bytes.

Return Value

None

Known Exceptions

peErrCantGetAttrs

**Header File**

PSFCalls.h

Related Methods

[PDSysFontGetEncoding](#)

[PDSysFontGetInfo](#)

[PDSysFontGetName](#)

[PDSysFontGetType0Widths](#)

Product

base, pro, pdf

Availability

Available if **PI_PDSYSFONT_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDSysFontGetCIDSystemInfo

```
void PDSysFontGetCIDSystemInfo (PDSysFont sysFont,  
ASAtom* registry, ASAtom* ordering, ASInt32* supplement);
```

Description

Derives the registry, ordering, and supplement information of a multi-byte system font. This information can be used to create a **PDEFont** from a system font. For more information on CID fonts, see **PDFontGetCIDSystemInfo**.

Parameters

sysFont	A PDSysFont object referencing a multibyte system font.
registry	(Filled by the method) The ASAtom representing the CIDFont’s Registry information, as in “Adobe”.
ordering	(Filled by the method) The ASAtom representing the CIDFont’s Ordering information, for example, “Japan1”.
supplement	(Filled by the method) The SystemSupplement field from the CIDFont.

Return Value

None

Header File

PSFCalls.h

Related Methods

- [PDFontGetCIDSystemInfo](#)
- [PDFontGetCIDSystemSupplement](#)

Product

base, pro, pdf

Availability

Available if **PI_PDSYSFONT_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSysFontGetCreateFlags

```
ASInt32 PDSysFontGetCreateFlags (PDSysFont sysFont ,
PDSysEncoding sysEnc) ;
```

Description

This function returns a **createFlags** that can be passed to [PDEFontCreateFromSysFontAndEncoding](#). If the combination of **sysFont** and **sysEnc** is not allowed, -1 is returned.

Parameters

sysFont	An object of type PDSysFont .
sysEnc	An object of type PDSysEncoding .

Return Value

See above.

Known Exceptions

[peErrWrongPDEObjectType](#)

Header File

PEWCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDFEDIT_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDSysFontGetEncoding

```
ASUns8** PDSysFontGetEncoding (PDSysFont sysFont,
ASAtom* encodingNameP);
```

Description

Gets the encoding of a single-byte encoded system font.

Parameters

sysFont	A PDSysFont object referencing a system font whose encoding is obtained.
encodingNameP	<i>(Filled by the method)</i> An encoding name if the standard encoding is used. For Macintosh, this is MacRomanEncoding ; for Windows, it is WinAnsiEncoding .

Return Value

If the return value is non-**NULL**, it is a pointer to an encoding array of 256 C strings. Each entry in the array either contains a glyph name or **NULL**; if **NULL**, the corresponding entry uses the font's built in encoding value.

The returned encoding must be freed via a call to **ASfree**.

NOTE: This encoding array is returned only on Macintosh platforms; on all others the function returns **NULL**.

Header File

PSFCalls.h

Related Methods

[PDSysFontAcquirePlatformData](#)
[PDSysFontGetInfo](#)
[PDSysFontGetName](#)
[PDSysFontGetType0Widths](#)

Product

base, pro, pdf

Availability

Available if **PI_PDSYSFONT_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSysFontGetInfo

```
void PDSysFontGetInfo (PDSysFont sysFont, PDEFontInfoP infoP,
ASInt32 infoSize);
```

Description

Gets high-level information about a system font.

Parameters

sysFont	A PDSysFont object referencing a system font whose information is obtained.
infoP	(Filled by the method) Pointer to PDEFontInfoP structure to fill with font information for sysFont. No more than infoSize bytes are written to this buffer.
infoSize	Size of the infoP buffer, in bytes.

Return Value

None

Header File

PSFCalls.h

Related Methods

- PDSysFontAcquirePlatformData
- PDSysFontGetEncoding
- PDSysFontGetName
- PDSysFontGetType0Widths

Product

base, pro, pdf

Availability

Available if PI_PDSYSFONT_VERSION (in PIRequir.h) is set to 0x00040000 or higher.

PDSysFontGetName

```
ASAtom PDSysFontGetName (PDSysFont sysFont);
```

Description

Gets the PostScript or TrueType styled name for a system font.

Parameters

sysFont	A PDSysFont object referencing a system font whose name is obtained.

Return Value

The **ASAtom** for the system font's name.

Header File

PSFCalls.h

Related Methods

[PDSysFontAcquirePlatformData](#)
[PDSysFontGetEncoding](#)
[PDSysFontGetInfo](#)
[PDSysFontGetType0Widths](#)

Product

base, pro, pdf

Availability

Available if **PI_PDSYSFONT_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDSysFontGetType0Widths

```
void PDSysFontGetType0Widths (PDSysFont sysFont,
ASAtom ordering, ASBool* hasDW, ASInt32* dw, CosObj* w,
ASBool* hasdw2, ASInt32* dw2, CosObj* w2);
```

Description

Gets width information from a Type 0 system font. This information can be used to create a **PDEFont** from a system font.

NOTE: In general, you are discouraged from using this method. Instead use **PDEFontCreateFromSysFontAndEncoding** followed by **PDEFontCreateWidthsNow** to create the **W** entry in a font.

Parameters

sysFont	A PDSysFont object referencing a multibyte system font.
ordering	ASAtom representing the CIDFont's Ordering information. Used to get a CMap object for sysFont .
hasdw	(Filled by the method) true if sysFont has a valid dw value; false otherwise.
dw	(Filled by the method) Default width for glyphs in a CIDFont. Currently, always 1000 . See Section 5.6 on CIDFontType 0 in the PDF Reference for more information.
w	(Filled by the method) A Cos array of a set of lists that define the widths for the glyphs in the CIDFont. Each list can specify individual widths for consecutive CIDs, or one width for a range of CIDs. See Section 5.6.3 on character widths in CIDFonts in the PDF Reference for information on the format of this array.
hasdw2	(Filled by the method) true if sysFont has a valid dw2 value. Default is false .
dw2	(Filled by the method) The default metrics for writing mode 1. This entry is an array of two ASInt32 numbers: the y component of the position vector and the y component of the displacement vector for writing mode 1. The x component of the position vector is always half the width of the character. The x component of the displacement vector is always 0. The default value is [880 - 1000]. For information on writing mode 1, see Section 5.6.3 on vertical writing in the PDF Reference .

w2	(Filled by the method) A Cos array defining the metrics for vertical writing. Its format is similar to the format of the array in <i>w</i> . It defines the x and y components of the position vector, and the y component of the displacement vector. The x component of the displacement vector is always 0. See Section 5.6.3 on character widths in CIDFonts in the PDF Reference for information on the format of this array.

Return Value

None

Header File

PSFCalls.h

Related Methods

[PDSysFontGetWidths](#)
[PDSysFontGetWidthsEx](#)
[PDFontGetWidths](#)

Product

base, pro, pdf

Availability

Available if **PI_PDSYSFONT_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDSysFontGetWidths

```
void PDSysFontGetWidths (PDSysFont sysFont, ASInt16* widthsP);
```

Description

Gets the widths of a single byte encoded system font.

Parameters

sysFont	A PDSysFont object referencing a system font whose widths are obtained.
widthsP	(Filled by the method) Pointer to widths array. widthsP must have room for 256 entries.

Return Value

None

Known Exceptions

peErrCantGetWidths

Header File

PSFCalls.h

Related Methods

PDSysFontGetType0Widths
PDSysFontGetWidthsEx
PDFontGetWidths

Product

base, pro, pdf

Availability

Available if **PI_PDSYSFONT_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSysFontGetWidthsEx

```
void PDSysFontGetWidthsEx (PDSysFont sysFont,
ASInt16* widthsP, ASFixed* mmDesignVector);
```

Description

Gets the widths of a single byte encoded system font.

Parameters

sysFont	A PDSysFont object referencing a system font whose widths are obtained.
widthsP	<i>(Filled by the method)</i> Pointer to widths array. widthsP must have room for 256 entries.
mmDesignVector	If sysFont is a multiple master font, points to the design vector, whose length must equal the number of design axes of sysFont .

Return Value

None

Known Exceptions

[peErrCantGetWidths](#)

Header File

PSFCalls.h

Related Methods

[PDSysFontGetType0Widths](#)
[PDSysFontGetWidths](#)
[PDFontGetWidths](#)

Product

base, pro, pdf

Availability

Available if **PI_PDSYSFONT_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDSysFontReleasePlatformData

```
void PDSysFontReleasePlatformData  
(PDSysFontPlatDataP platDataP);
```

Description

Releases platform-specific data for the specified **PDSysFont**.

Parameters

platDataP	A pointer to a PDSysFontPlatDataP structure containing platform-specific data.

Return Value

None

Header File

PSFCalls.h

Related Methods

[PDSysFontAcquirePlatformData](#)

Product

base, pro, pdf

Availability

Available if **PI_PDSYSFONT_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDSEdit Methods

NOTE: Methods in this layer that write are not available in Adobe Reader.

[PDSAttrObj](#)

[PDSClassMap](#)

[PDSElement](#)

[PDSMC](#)

[PDSOBJR](#)

[PDSRoleMap](#)

[PDSTreeRoot](#)

PDSAttrObj

PDSAttrObjCreate

```
void PDSAttrObjCreate (PDDoc pdDoc, ASAtom owner,  
ASBool indirect, PDSAttrObj* attrObj);
```

Description

Creates a new attribute object with the specified owner.

Parameters

pdDoc	Document in which the attribute object is created.
owner	Owner of the new attribute object.
indirect	If true , creates the attribute object as an indirect Cos object and sets pdDoc 's PDDocNeedsSave flag (see PDDocFlags). If false , creates the attribute object as a direct object.
attrObj	<i>(Filled by the method)</i> The newly created attribute object.

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

[PDSAttrObjCreateFromStream](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDSAttrObjCreateFromStream

```
void PDSAttrObjCreateFromStream (ASAtom owner,  
CosObj cosStreamObj, PDSAttrObj* attrObj);
```

Description

Creates an attribute object with the specified owner from the specified Cos stream.

Parameters

owner	Owner of the new attribute object.
cosStreamObj	The Cos stream containing the data with which to create the attribute. The dictionary of this stream is modified.
attrObj	<i>(Filled by the method)</i> Pointer to the newly created attribute object. This actually points to cosStreamObj . May be NULL .

Return Value

None

Known Exceptions

Among others, raises [pdsErrWrongTypeParameter](#) if **cosStreamObj** is not a Cos stream.

Header File

PDSWriteCalls.h

Related Methods

[PDSAttrObjCreate](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSAttrObjGetCosObj

```
CosObj PDSAttrObjGetCosObj (PDSAttrObj attrObj);
```

Description

Gets the Cos object corresponding to the specified attribute object. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

attrObj	The attribute object whose Cos object is obtained.

Return Value

The dictionary Cos object for the attribute object.

Header File

`PDSReadCalls.h`

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDSAttrObjGetOwner

```
ASAtom PDSAttrObjGetOwner (PDSAttrObj attrObj);
```

Description

Gets the value of the key (**Owner**) in the specified attribute object.

Parameters

attrObj	The attribute object whose owner is obtained.

Return Value

The **ASAtom** for the owner's name.

Known Exceptions

Various

Header File

PDSReadCalls.h

Related Methods

[PDSAttrObjCreate](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSClassMap

PDSClassMapAddAttrObj

```
void PDSClassMapAddAttrObj (PDSClassMap classMap,
                             ASAtom classAtom, PDSAttrObj attrObj);
```

Description

Adds the specified attribute object to the specified **PDSClassMap** for the given class name. If the attribute object is already present, it is not added a second time.

Parameters

classMap	The PDSClassMap to which the specified attribute object is added.
classAtom	The ASAtom representing the class name.
attrObj	Attribute object to add to the class in classAtom .

Return Value

None

Known Exceptions

Raises **pdsErrBadPDF** if an error is found in the PDF file.

Header File

PDSWriteCalls.h

Related Methods

[PDSClassMapGetAttrObj](#)
[PDSClassMapRemoveAttrObj](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDSClassMapGetAttrObj

```
void PDSClassMapGetAttrObj (PDSClassMap classMap,
    ASAtom classAtom, ASInt32 index, PDSAttrObj* attrObj);
```

Description

Gets the attribute object associated with the specified class name at an index in the class. If there is only one object and **index** is zero, that object is retrieved.

Parameters

classMap	The PDSClassMap .
classAtom	The ASAtom of a class name for which an associated attribute objects is found.
index	Index of the desired attribute object in the class.
attrObj	<i>(Filled by the method)</i> Attribute object at index . Set to CosNull if there is no attribute object at the specified location.

Return Value

None

Known Exceptions

Various

Header File

PDSReadCalls.h

Related Methods

[PDSClassMapAddAttrObj](#)
[PDSClassMapGetNumAttrObjs](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSClassMapGetNumAttrObjs

```
ASInt32 PDSClassMapGetNumAttrObjs (PDSClassMap classMap,
ASAtom classAtom);
```

Description

Gets the number of attribute objects associated with a class name.

Parameters

classMap	The PDSClassMap .
classAtom	The ASAtom of a class name for which the number of associated attribute objects is found.

Return Value

Number of attribute objects associated with the class in **classAtom**.

Known Exceptions

Various

Header File

PDSReadCalls.h

Related Methods

[PDSClassMapGetAttrObj](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSClassMapRemoveAttrObj

```
void PDSClassMapRemoveAttrObj (PDSClassMap classMap,
ASAtom classAtom, PDSAttrObj attrObj);
```

Description

Removes the specified attribute object from the specified **PDSClassMap**. If **classAtom** is **ASAtomNull**, removes all occurrences of **attrObj** in the entire **classMap**.

Parameters

classMap	The PDSClassMap from which the specified attribute object is removed.
classAtom	The ASAtom of a class name for which the associated attribute object is found.
attrObj	Attribute object to remove from classMap .

Return Value

None

Known Exceptions

Raises **pdsErrBadPDF** if an error is found in the PDF file.

Header File

PDSWriteCalls.h

Related Methods

[PDSClassMapAddAttrObj](#)
[PDSClassMapRemoveClass](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSClclassMapRemoveClass

```
void PDSClclassMapRemoveClass (PDSClclassMap classMap,  
ASAtom classAtom);
```

Description

Removes the specified class from the specified **PDSClclassMap**, if it exists.

Parameters

classMap	The PDSClclassMap from which a class is removed.
classAtom	The ASAtom representing the class to remove from classMap .

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

[PDSClclassMapRemoveAttrObj](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDSElement

PDSElementAddAttrObj

```
void PDSElementAddAttrObj (PDSElement element,  
    PDSAttrObj attrObj);
```

Description

Associates the specified attribute object with an element at the element's current revision value.

Parameters

element	Element with which attrObj is associated.
attrObj	Attribute object to associate with element .

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

[PDSElementGetAttrObj](#)
[PDSElementRemoveAttrObj](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDSElementAddClass

```
void PDSElementAddClass (PDSElement element,
                          ASAtom classAtom);
```

Description

Adds a class name to the element’s list of classes to which it belongs at the element’s current revision value.

Parameters

element	Element to which a class is added.
classAtom	The ASAtom representing the class to add to element . If classAtom is already present among element ’s classes, it will not be added again.

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

- PDSElementGetClass
- PDSElementGetNumClasses
- PDSElementRemoveAllClasses
- PDSElementRemoveClass

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDSElementClearID

```
void PDSElementClearID (PDSElement element);
```

Description

Removes an element's ID, if it exists.

Parameters

element	Element whose ID is removed.

Return Value

None

Known Exceptions

Raises [pdsErrWrongTypeParameter](#) if **element** is not a valid **PDSElement**.

Raises [pdsErrBadPDF](#) if an error is found in the PDF file.

Header File

PDSWriteCalls.h

Related Methods

[PDSElementGetID](#)

[PDSElementSetID](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementCreate

```
void PDSElementCreate (PDDoc pdDoc, PDSElement* element);
```

Description

Creates a new (but empty) **PDSElement**.

Parameters

pdDoc	The PDDoc in which the PDSElement is created.
element	<i>(Filled by the method)</i> The newly created PDSElement .

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementGetActualText

```
ASInt32 PDSElementGetActualText (PDSElement element,
ASUns8 *buffer) ;
```

Description

Gets the actual text associated with the specified **PDSElement**. Returns the number of bytes in the text or 0 if the element has no actual text or has an empty string.

To check for the existence of alternate text, check for a non-zero return value. To get the needed size of buffer, call this method with a **NULL** buffer.

NOTE: Due to implementation issues, make the buffer one byte larger than the required size. Code will not null-terminate the string correctly in the case of Unicode strings.

Parameters

element	The structural element whose actual text is sought.
buffer	If not NULL , buffer contains the element's actual text. The string is null-terminated (but not correctly for Unicode). This is not a C-style string, so normal string handling functions may not work; the buffer may contain a Unicode string.

Return Value

An **ASInt32** representing the number of bytes in the text or 0 if the element has no actual text.

Header File

PDSReadCalls.h

Related Methods

[PDSElementSetActualText](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDSElementGetAlt

ASInt32 PDSElementGetAlt (PDSElement element, ASUns8* buffer);

Description

Gets the alternate text associated with an element.

Can first be called with a **NULL** buffer to find the size, so that **buffer** can then be appropriately sized.

NOTE: The Alt text can be legally defined as an empty string. To differentiate between an Alt text string of zero length and no Alt text being defined, call **PDSElementHasAlt** first.

NOTE: Due to implementation issues, make the buffer one byte larger than the required size. Code will not null-terminate the string correctly in the case of Unicode strings.

Parameters

element	The element whose alternate text is obtained.
buffer	<i>(Filled by the method)</i> A buffer into which the alternate text is placed. May be NULL , if the method is called only to find the length of the element's alternate text. If not NULL , buffer contains the element's actual text. The string is null-terminated (but not correctly for Unicode). This is not a C-style string, so normal string handling functions may not work; the buffer may contain a Unicode string.

Return Value

Number of bytes in **element**'s alternate text.

Known Exceptions

Raises **pdsErrWrongTypeParameter** if **element** is not a valid **PDSElement**.

Header File

PDSReadCalls.h

Related Methods

PDSElementSetAlt
PDSElementHasAlt

Product

reader, base, pro, pdf

**Availability**

Available if **PI_PDS_READ_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDSElementGetAttrObj

```
ASInt32 PDSElementGetAttrObj (PDSElement element,
ASInt32 index, PDSAttrObj* attrObj);
```

Description

Gets the attribute object at a specified array index in the specified element.

If there is only one attribute object (that is, there is no array of attributes), and **index** is zero, that attribute object is obtained.

Parameters

element	The element whose attribute is obtained.
index	Index of the attribute object to obtain.
attrObj	<i>(Filled by the method)</i> Attribute object at index .

Return Value

Revision number of **element** at time of last association.

Known Exceptions

[pdsErrRequiredMissing](#)

Raises [pdsErrWrongTypeParameter](#) if **element** is not a valid **PDSElement**.

Header File

PDSReadCalls.h

Related Methods

[PDSElementAddAttrObj](#)
[PDSElementGetNumAttrObjs](#)
[PDSElementRemoveAttrObj](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementGetClass

```
ASInt32 PDSElementGetClass (PDSElement element, ASInt32 index,
ASAtom* classAtom) ;
```

Description

Gets the class name at an array index in the specified element.
If there is only one attribute object (that is, there is no array), and **index** is zero, that class name is obtained.

Parameters

element	The element whose class is obtained.
index	Index of the class to obtain.
classAtom	(Filled by the method) The ASAtom describing the class.

Return Value

Revision number of **element** at time of last association.

Known Exceptions

[pdsErrRequiredMissing](#)
Raises [pdsErrWrongTypeParameter](#) if **element** is not a valid **PDSElement**.

Header File

PDSReadCalls.h

Related Methods

[PDSElementAddClass](#)
[PDSElementGetNumClasses](#)
[PDSElementRemoveAllClasses](#)
[PDSElementRemoveClass](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementGetCosObj

```
CosObj PDSElementGetCosObj (PDSElement element);
```

Description

Gets the Cos object corresponding to the specified element object. This method does not copy the object, but is instead the logical equivalent of a type cast.

Parameters

element	The element object whose Cos object is obtained.

Return Value

The dictionary Cos object for the element object.

Header File

PDSReadCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDSElementGetFirstPage

```
CosObj PDSElementGetFirstPage (PDSElement element,
ASAtom* firstKidType, CosObj* firstCosObjKidOnAPage,
PDEContainer* firstMCKidOnAPage);
```

Description

Gets the Cos object for the page of the first kid of the element.

NOTE: The order in which the returned page is first is the order of kids, not the order of pages. That is, the first descendant with page content determines which page is returned.

Parameters

element	Element whose kid's first page is found.
firstKidType	<i>(Filled by the method)</i> Pointer to an ASAtom for the name that appears as the Type entry of the actual first kid of element . Possible values are the values that PDSElementGetKid can return. Pass NULL to inhibit setting firstKidType .
firstCosObjKidOnAPage	<i>(Filled by the method)</i> The kid whose content determined that the page returned was the first page with content—if that kid is a CosObj . Pass NULL to inhibit setting firstCosObjKidOnAPage .
firstMCKidOnAPage	<i>(Filled by the method)</i> The kid whose content determined that the page returned was the first page with content—if that kid is marked content that is <i>not</i> a CosObj . Pass NULL to inhibit setting firstMCKidOnAPage .

Return Value

The **CosObj** of the page found, **CosObjNull** if the element has no page content.

Known Exceptions

Various

Header File

PDSReadCalls.h

Related Methods

[PDSElementGetKid](#)

Example

```
cosPage = PDSElementGetFirstPage(pdsElement, NULL, NULL, NULL);
```

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDSElementGetID

ASInt32 PDSElementGetID (PDSElement element, ASUns8* idBuf);

Description

Gets the ID of an element or CosObjNull if there is no ID set.

Parameters

element	Element whose ID is obtained.
idBuf	(Filled by the method) Pointer to the buffer containing the element's ID.

Return Value

The number of bytes in the ID, or zero if the element has no ID.

Known Exceptions

Raises pdsErrWrongTypeParameter if element is not a valid PDSElement.

Raises pdsErrBadPDF if an error is found in the PDF file.

Header File

PDSReadCalls.h

Related Methods

- PDSElementClearID
- PDSElementSetID
- PDSTreeRootGetElementFromID

Product

reader, base, pro, pdf

Availability

Available if PI_PDS_READ_VERSION (in PIRequir.h) is set to 0x00040000 or higher.

PDSElementGetKid

```
ASAtom PDSElementGetKid (PDSElement element, ASInt32 index,
CosObj* cosObjKid, void** pointerKid, CosObj* cosPage);
```

Description

Gets the kid at an array index in the specified element.

A PDF structural element—unlike the structure tree root—can have *several different kinds* of children: marked content, another element, or an entire PDF object. The parameter in which the kid is placed depends on the type of kid. If the kid is a structural element or an object reference, **PDSElementGetKid** places the result in **cosObjKid**; if the kid is page content, it is placed in **pointerKid**.

Any or all of **cosObjKid**, **pointerKid**, and **cosPage** can be **NULL** to get the kid's type without setting that parameter.

NOTE: When the kid is an **MC**, it is actually a pointer of the type **PDEContainer**. As with all PDFEdit objects, you must be careful to manage the reference count of the object by calling **PDEAcquire** and **PDERelease**. **PDSElementGetKid** does not call **PDEAcquire** for you.

NOTE: This method cannot access marked content inside a Form XObject.

Parameters

element	Element whose specified kid is found.
index	Index of the kid to obtain.
cosObjKid	<i>(Filled by the method)</i> The CosObj of the specified kid—if that kid is a PDSElement or an OBJR . If cosObjKid is NULL , it is not filled in, but the type of the kid is returned regardless. NOTE: This CosObj can be treated as a PDSElement or a PDSObjR. Use the return type to decide which to use.
pointerKid	<i>(Filled by the method)</i> Pointer to the kid at index —if that kid is an MC . If pointerKid is NULL , it is not filled in, but the type of the kid is returned regardless.
cosPage	<i>(Filled by the method)</i> Pointer to the CosObj of the page containing the kid. If cosPage is NULL , it is not filled in, but the type of the kid is returned regardless.

Return Value

The **ASAtom** representing the kid's **Type** value: **StructElem**, **MC** or **OBJR**. **MCR** is never returned.



Known Exceptions

Raises [pdsErrWrongTypeParameter](#) if `element` is not a valid `PDSElement`.

Raises [pdsErrBadPDF](#) if an error is found in the PDF file.

Header File

`PDSReadCalls.h`

Related Methods

[PDSElementGetFirstPage](#)

[PDSElementGetKidEx](#)

[PDSElementGetKidWithMCInfo](#)

[PDSElementGetNumKids](#)

[PDSElementInsertKid](#)

Product

reader, base, pro, pdf

Availability

Available if `PI_PDS_READ_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.

PDSElementGetKidEx

```
ASAtom PDSElementGetKidEx (PDSElement element, ASInt32 index,
CosObj* cosObjKid, ASInt32* mcid, void** pointerKid,
CosObj* cosPage);
```

Description

Functions identically to **PDSElementGetKid**, but for children that are marked contents can return the **mcid** as well as or instead of the actual object.

NOTE: This method cannot access marked content inside a Form XObject.

Parameters

element	The PDSElement containing the kid that is retrieved.
index	The index of the kid.
cosObjKid	(Filled in by method) The kid being accessed (depending on the kid's type) or NULL .
mcid	(Filled in by method) The kid's mcid or NULL .
pointerKid	(Filled in by method) Pointer to the kid or NULL .
cosPage	(Filled in by method) The CosObj of the page containing the kid or NULL .

Return Value

An **ASAtom** representing the **Type** value of the kid. See above.

Known Exceptions

Raises **pdsErrWrongTypeParameter** if **element** is not a valid **PDSElement**.
 Raises **pdsErrBadPDF** if an error is found in the PDF file.

Header File

PDSReadCalls.h

Related Methods

[PDSElementGetKid](#)
[PDSElementGetKidWithMCInfo](#)

Product

reader, base, pro, pdf

**Availability**

Available if **PI_PDS_READ_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDSElementGetKidWithMCInfo

```
ASAtom PDSElementGetKidWithMCInfo (PDSElement element,
ASInt32 index, CosObj* cosObjKid, PDSMCInfo* mcidInfo,
void** pointerKid, CosObj* cosPage);
```

Description

Functions identically to **PDSElementGetKidEx**, but returns additional information about marked content kids that are in streams other than the page content streams.

Parameters

element	The PDSElement containing the kid that is retrieved.
index	The index of the kid.
cosObjKid	<i>(Filled in by method)</i> The kid being accessed (depending on the kid's type) or NULL .
mcidInfo	<i>(Filled in by method)</i> The kid's information object or NULL .
pointerKid	<i>(Filled in by method)</i> Pointer to the kid or NULL .
cosPage	<i>(Filled in by method)</i> The CosObj of the page containing the kid or NULL .

Return Value

An **ASAtom** representing the **Type** value of the kid.

Known Exceptions

Raises **pdsErrWrongTypeParameter** if **element** is not a valid **PDSElement**.
 Raises **pdsErrBadPDF** if an error is found in the PDF file.

Header File

PDSReadCalls.h

Related Methods

[PDSElementGetKid](#)
[PDSElementGetKidEx](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDSElementGetLanguage

```
ASInt32 PDSElementGetLanguage (PDSElement element,
ASUns8 *buffer) ;
```

Description

Gets the language associated with the specified **PDSElement**.
Returns the number of bytes in the language string or 0 if the element has no language or has an empty string.
To check for the existence of expansion text, call **PDSElementHasLanguage**. To get the needed buffer size, call this method with a **NULL** buffer.

NOTE: Due to implementation issues, make the buffer one byte larger than the required size. Code will not null-terminate the string correctly in the case of Unicode strings.

Parameters

element	The structural element whose expansion text is sought.
buffer	<i>(Filled by the method)</i> A buffer containing the element's expansion text or NULL . See PDSElementSetLanguage for format and languages. If not NULL , buffer contains the element's expansion text. The string is null-terminated (but not correctly for Unicode). This is not a C-style string, so normal string handling functions may not work; the buffer may contain a Unicode string.

Return Value

An **ASInt32** representing the number of bytes in the language string.

Header File

PDSReadCalls.h

Related Methods

[PDSElementSetLanguage](#)
[PDSElementHasLanguage](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDSElementGetNumAttrObjs

ASInt32 PDSElementGetNumAttrObjs (**PDSElement** element);

Description

Gets the number of attribute objects directly attached to the specified element.

Parameters

element	The element whose number of attributes is obtained.

Return Value

Number of attribute objects directly attached to **element**.

Known Exceptions

Raises **pdsErrWrongTypeParameter** if **element** is not a valid **PDSElement**.

Header File

PDSReadCalls.h

Related Methods

PDSElementGetAttrObj

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementGetNumClasses

```
ASInt32 PDSElementGetNumClasses (PDSElement element);
```

Description

Gets the number of classes to which the specified element belongs.

Parameters

element	The element whose number of classes is obtained.

Return Value

Number of classes to which **element** belongs.

Known Exceptions

Raises [pdsErrWrongTypeParameter](#) if **element** is not a valid **PDSElement**.

Header File

PDSReadCalls.h

Related Methods

[PDSElementGetClass](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementGetNumKids

```
ASInt32 PDSElementGetNumKids (PDSElement element);
```

Description

Gets the number of kids of the specified element.

Parameters

element	Element whose number of kids is obtained.

Return Value

Number of direct kids of **element**.

Known Exceptions

Raises [pdsErrWrongTypeParameter](#) if **element** is not a valid **PDSElement**.

Header File

PDSReadCalls.h

Related Methods

[PDSElementGetKid](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementGetParent

```
void PDSElementGetParent (PDSElement element,  
    PDSElement* parent, ASBool* parentIsTreeRoot);
```

Description

Gets the immediate ancestor element of the specified element in the tree.

If the element's **parent** is another element, **parent** is set to that parent and **parentIsTreeRoot** is set to **false**. If the element's **parent** is the structure tree root, **parent** is set to **CosNull** and **parentIsTreeRoot** is set to **true**. If **parentIsTreeRoot** is **NULL**, it is not set.

Parameters

element	The element whose parent is obtained.
parent	(Filled by the method) The element 's parent.
parentIsTreeRoot	(Filled by the method) The element 's parent is the structure tree root.

Return Value

None

Known Exceptions

Raises **pdsErrWrongTypeParameter** if **element** is not a valid **PDSElement**.

Header File

PDSReadCalls.h

Related Methods

- PDSElementGetKid
- PDSElementGetStructTreeRoot
- PDSMCGetInfo
- PDSOBJGetParent

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementGetRevision

```
ASInt32 PDSElementGetRevision (PDSElement element);
```

Description

Gets the revision number of an element.

Parameters

element	The element whose revision is obtained.

Return Value

Revision number of **element**.

Known Exceptions

Raises [pdsErrWrongTypeParameter](#) if **element** is not a valid **PDSElement**.

Header File

PDSReadCalls.h

Related Methods

[PDSElementIncrementRevision](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementGetStructTreeRoot

```
ASBool PDSElementGetStructTreeRoot (PDSElement element,  
PDSTreeRoot* treeRoot);
```

Description

Gets the structure tree root of the document containing **element**.

Parameters

element	Element whose title is obtained.
treeRoot	<i>(Filled by the method)</i> The structure tree root.

Return Value

true if the document has a structure tree root, **false** otherwise. If there is a structure tree root, sets **treeRoot** to be the structure tree root.

Known Exceptions

Raises **pdsErrWrongTypeParameter** if **element** is not a valid **PDSElement**.

Header File

PDSReadCalls.h

Related Methods

[PDSTreeRootGetKid](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementGetTitle

```
ASInt32 PDSElementGetTitle (PDSElement element,
ASUns8* buffer);
```

Description

Gets the title of the specified element, returning the number of bytes in the title.

Can first be called with a **NULL** buffer to find the title size, so that **buffer** can be appropriately sized as one greater than the title's length.

NOTE: Due to implementation issues, make the buffer one byte larger than the required size.

Parameters

element	Element whose title is obtained.
buffer	(Filled by the method) A buffer into which the title text is placed. May be NULL , in which case the number of bytes in the title is returned.

Return Value

Number of bytes in **element**'s title, or zero if **element** has no title.

Known Exceptions

Raises [pdsErrWrongTypeParameter](#) if **element** is not a valid **PDSElement**.

Header File

PDSReadCalls.h

Related Methods

[PDSElementSetTitle](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDSElementGetType

```
ASAtom PDSElementGetType (PDSElement element);
```

Description

Gets the element's structural element type. The type corresponds to the **Subtype** key in the structure element dictionary.

PDSElementGetType gets the value of the **Subtype** key—not the **Type** key—in the structure element dictionary. All **PDSElements** have a **Type** value of **StructElem**.

Parameters

element	The element whose structural element type is obtained.

Return Value

The **ASAtom** representing **element**'s type.

Known Exceptions

Raises **pdsErrWrongTypeParameter** if **element** is not a valid **PDSElement**.

Header File

PDSReadCalls.h

Related Methods

PDSElementSetType

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementHasActualText

```
ASBool PDSElementHasActualText (PDSElement element);
```

Description

Tests whether **ActualText** is defined for a given **PDSElement**.

Parameters

element	The PDSElement being tested.

Return Value

true if text exists (including the empty string); **false** otherwise.

Header File

PDSReadCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDSElementHasAlt

```
ASBool PDSElementHasAlt (PDSElement element);
```

Description

Tests whether **Alt** text is defined for a given **PDSElement**.

Parameters

element	The PDSElement being tested.

Return Value

true if text exists (including the empty string); **false** otherwise.

Header File

PDSReadCalls.h

Related Methods

[PDSElementGetAlt](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDSElementHasLanguage

```
ASBool PDSElementHasLanguage (PDSElement element);
```

Description

Tests whether a language string is defined for a given **PDSElement**.

Parameters

element	The PDSElement being tested.

Return Value

true if text exists (including the empty string); **false** otherwise.

Header File

PDSReadCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.



PDSElementIncrementRevision

```
void PDSElementIncrementRevision (PDSElement element);
```

Description

Increments an element's revision count by one.

Parameters

element	Element whose revision count is incremented.

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDSElementInsertKid

```
void PDSElementInsertKid (PDSElement element, PDSElement kid,
ASInt32 insertAfter);
```

Description

Inserts the specified kid **PDSElement** object into the specified element after position **insertAfter**.

Parameters

element	Element in which the specified kid is inserted.
kid	The kid to insert.
insertAfter	Position after which the kid is inserted. If element currently has no kids, insertAfter is ignored.

Return Value

None

Known Exceptions

[pdsErrWrongTypeParameter](#)

Header File

PDSWriteCalls.h

Related Methods

[PDSElementGetFirstPage](#)
[PDSElementGetKid](#)
[PDSElementInsertMCAsKid](#)
[PDSElementInsertOBJAsKid](#)
[PDSElementInsertStmMCAsKid](#)
[PDSElementRemoveKid](#)
[PDSElementReplaceKid](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementInsertMCAsKid

```
void PDSElementInsertMCAsKid (PDSElement element,  
CosObj cosPage, PDSMC mc, ASInt32 insertAfter);
```

Description

Inserts a reference to the specified **PDSMC** (marked content) in the specified element after position **insertAfter**.

This method automatically creates **MCR** objects if needed.

Parameters

element	Element in which the reference is inserted.
cosPage	The CosObj for the page containing the reference to insert.
mc	The marked content to insert.
insertAfter	Position after which the reference is inserted. If element currently has no kids, insertAfter is ignored.

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

[PDSElementInsertKid](#)
[PDSElementInsertOBJAsKid](#)
[PDSElementInsertStmMCAsKid](#)
[PDSElementReplaceKidMC](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementInsertOBJAsKid

```
void PDSElementInsertOBJAsKid (PDSElement element,  
    CosObj cosPage, CosObj obj, ASInt32 insertAfter);
```

Description

Inserts a reference to the specified PDF object as a kid into the specified element.

Parameters

element	Element in which the reference is inserted.
cosPage	The CosObj for the page containing the reference to insert.
obj	The CosObj to insert.
insertAfter	Position after which the reference is inserted in element. If element currently has no kids, insertAfter is ignored.

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

[PDSElementReplaceKidOBJ](#)

Product

base, pro, pdf

Availability

Available if `PI_PDS_WRITE_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.

PDSElementInsertStmMCAsKid

```
void PDSElementInsertStmMCAsKid (PDSElement element,
    CosObj cosPage, CosObj containingStm, CosObj stmOwner,
    PDSMC mc, ASInt32 insertAfter);
```

Description

Inserts a marked content sequence from a non-page-content stream as a kid of the specified element.

Parameters

element	Element in which the reference is inserted.
cosPage	The CosObj for the page containing the reference to insert.
cosStream	The stream containing the content given by mc .
streamOwner	The PDF object owning the stream given in cosStream ; for example, the annotation to which an appearance stream belongs. Can be CosNull if owner is not important.
mc	The marked content to insert.
insertAfter	Position after which the reference is inserted. If element currently has no kids, insertAfter is ignored.

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

[PDSElementInsertKid](#)
[PDSElementInsertMCAsKid](#)
[PDSElementInsertOBJAsKid](#)
[PDSElementReplaceKidMC](#)

Product

base, pro, pdf



PDSElementInsertStmMCAsKid

Availability

Available if **PI_PDS_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.



PDSElementRemoveAllAttrObjs

```
void PDSElementRemoveAllAttrObjs (PDSElement element);
```

Description

Removes *all* attribute objects directly associated with the specified element.

Parameters

element	Element whose attributes are removed.

Return Value

None

Known Exceptions

Raises [pdsErrWrongTypeParameter](#) if **element** is not a valid **PDSElement**.

Header File

PDSWriteCalls.h

Related Methods

[PDSElementRemoveAttrObj](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to 0x00040000 or higher.

PDSElementRemoveAllClasses

```
void PDSElementRemoveAllClasses (PDSElement element);
```

Description

Removes *all* classes from the specified element.

Parameters

element	Element whose classes are removed.

Return Value

None

Known Exceptions

Raises [pdsErrWrongTypeParameter](#) if **element** is not a valid **PDSElement**.

Header File

PDSWriteCalls.h

Related Methods

[PDSElementAddClass](#)
[PDSElementGetClass](#)
[PDSElementGetNumClasses](#)
[PDSElementRemoveClass](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementRemoveAttrObj

```
void PDSElementRemoveAttrObj (PDSElement element,
                               PDSAttrObj attrObj);
```

Description

Removes the specified attribute object from an element. If **element** does not have an **attrObj** attribute, this method does nothing.

NOTE: Calling **PDSElementRemoveAttrObj** while iterating over the attribute objects of an element will change the relationship between attribute object indices and attribute objects. Although it is possible to track this change in indices in a single loop, it is more straightforward to accumulate a list of attribute objects to remove during one pass over the attribute objects and to carry out the actual removals during a subsequent iteration over the accumulated list.

Parameters

element	Element whose attribute is removed.
attrObj	Attribute object to remove.

Return Value

None

Known Exceptions

Raises **pdsErrWrongTypeParameter** if **element** is not a valid **PDSElement** or **attrObj** is not a valid attribute object.

Header File

PDSWriteCalls.h

Related Methods

[PDSElementAddAttrObj](#)
[PDSElementGetAttrObj](#)
[PDSElementRemoveAllAttrObjs](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementRemoveClass

```
void PDSElementRemoveClass (PDSElement element,
                              ASAtom classAtom);
```

Description

Removes the specified class name from the element’s list of classes to which it belongs.

NOTE: Calling PDSElementRemoveClass while iterating over the classes of an element will change the relationship between class indices and classes. Although it is possible to track this change in indices in a single loop, it is more straightforward to accumulate a list of classes to remove during one pass over the classes and to carry out the actual removals during a subsequent iteration over the accumulated list.

Parameters

element	Element from which the specified class is removed.
classAtom	The ASAtom representing the class to remove.

Return Value

None

Known Exceptions

Raises pdsErrWrongTypeParameter if element is not a valid PDSElement.

Header File

PDSWriteCalls.h

Related Methods

- PDSElementAddClass
- PDSElementGetClass
- PDSElementGetNumClasses
- PDSElementRemoveAllClasses

Product

base, pro, pdf

Availability

Available if PI_PDS_WRITE_VERSION (in PIRequir.h) is set to 0x00040000 or higher.

PDSElementRemoveKid

```
void PDSElementRemoveKid (PDSElement element, CosObj kid);
```

Description

Removes the specified kid from an element.

NOTE: Approved method of removing OBJ kids is [PDSElementRemoveKidOBJ](#).

Parameters

element	Element whose kid is removed.
kid	Kid to remove.

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

[PDSElementGetKid](#)
[PDSElementInsertKid](#)
[PDSElementRemoveKidMC](#)
[PDSElementRemoveKidOBJ](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDSElementRemoveKidMC

```
void PDSElementRemoveKidMC (PDSElement element,
                             CosObj cosPage, PDSMC mc);
```

Description

Removes the specified PDSMC (marked content) from an element’s kids, if it has any.
After calling this method, use PDPageSetPDEContent to commit any changes that have been made to the page contents.

Parameters

element	Element whose reference is removed.
cosPage	The CosObj for the page containing the reference to remove.
mc	The marked content to remove.

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

- PDSElementInsertMCAsKid
- PDSElementReplaceKidMC
- PDSElementRemoveKid

Product

base, pro, pdf

Availability

Available if PI_PDS_WRITE_VERSION (in PIRequir.h) is set to 0x00040000 or higher.



PDSElementRemoveKidOBJ

```
void PDSElementRemoveKidOBJ (PDSElement element, CosObj kid);
```

Description

Removes an OBJ from among the kids of a given element. Does nothing if the given OBJ isn't a kid of the given element.

Parameters

element	Element whose kid is having an OBJ removed.
kid	kid whose OBJ is removed.

Return Value

None

Known Exceptions

Various

Notifications

None

Header File

PDSWriteCalls.h

Related Methods

[PDSElementInsertMCAsKid](#)
[PDSElementReplaceKidMC](#)
[PDSElementRemoveKid](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDSElementReplaceKid

```
void PDSElementReplaceKid (PDSElement element, CosObj oldKid,  
CosObj newKid);
```

Description

Replaces the specified kid in the specified element.

NOTE: Approved method of replacing OBJ kids is [PDSElementReplaceKidOBJ](#).

Parameters

element	Element whose kid is replaced.
oldKid	Kid to replace.
newKid	Kid that is replacing oldKid .

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

[PDSElementInsertKid](#)
[PDSElementGetFirstPage](#)
[PDSElementGetKid](#)
[PDSElementRemoveKid](#)
[PDSElementReplaceKidMC](#)
[PDSElementReplaceKidOBJ](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in [PIRequir.h](#)) is set to **0x00040000** or higher.

PDSElementReplaceKidMC

```
void PDSElementReplaceKidMC (PDSElement element,
CosObj oldCosPage, PDSMC oldMC, CosObj newCosPage,
PDSMC newMC) ;
```

Description

Replaces the specified **PDSMC** (on **oldCosPage**) with a new **PDSMC** (on **newCosPage**) in the specified element.

Parameters

element	Element whose reference is replaced.
oldCosPage	The CosObj for the page holding the reference to replace.
oldMC	The marked content to replace.
newCosPage	The CosObj for the page holding the reference that is replacing oldMC .
newMC	The marked content that is replacing oldMC .

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

[PDSElementInsertMCAsKid](#)
[PDSElementRemoveKidMC](#)
[PDSElementReplaceKid](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementReplaceKidOBJ

```
void PDSElementReplaceKidOBJ (PDSElement element,  
    CosObj oldObj, CosObj newObj, CosObj newPage);
```

Description

Replaces `oldObj` with `newObj` on the specified page in the specified `element`.

Parameters

<code>element</code>	Element whose object is replaced.
<code>oldObj</code>	Object to replace.
<code>newObj</code>	Object that is replacing <code>oldObj</code> .
<code>newPage</code>	The <code>CosObj</code> for the page holding the reference that is replacing <code>oldObj</code> .

Return Value

None

Known Exceptions

Various

Header File

`PDSWriteCalls.h`

Related Methods

- [PDSElementInsertOBJAsKid](#)
- [PDSElementReplaceKid](#)
- [PDSElementReplaceKidMC](#)

Product

base, pro, pdf

Availability

Available if `PI_PDS_WRITE_VERSION` (in `PIRequir.h`) is set to `0x00040000` or higher.

PDSElementSetActualText

```
void PDSElementSetActualText (PDSElement element,
    const ASUns8 *buffer, ASInt32 nBytes);
```

Description

Sets the actual text representation of the specified **PDSElement**'s contents to buffer (from 0 to **nBytes**)

Parameters

element	The PDSElement whose contents are being set to buffer .
buffer	The buffer to which the PDSElement 's contents are being set.
nBytes	The number of bytes in the text representation.

Return Value

None

Header File

PDSWriteCalls.h

Related Methods

[PDSElementGetActualText](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

PDSElementSetAlt

```
void PDSElementSetAlt (PDSElement element,  
    const ASUns8* buffer, ASInt32 nBytes);
```

Description

Sets the alternate text representation of an element's contents.

Parameters

element	Element whose alternate text representation is set.
buffer	Pointer to a buffer containing a string to be made the element's alternate text representation.
nBytes	Number of bytes in buffer to use as element's new alternate text representation. May be zero. Sets an Alt string even if the buffer length is zero, but such an Alt string looks like no Alt string according to PDSElementGetAlt .

Return Value

None

Known Exceptions

Raises [pdsErrWrongTypeParameter](#) if **element** is not a valid **PDSElement**.

Header File

PDSWriteCalls.h

Related Methods

[PDSElementGetAlt](#)
[PDSElementHasAlt](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementSetID

```
void PDSElementSetID (PDSElement element,
    const ASUns8* buffer, ASInt32 nBytes);
```

Description

Sets the ID of an element to the given Cos string.

Parameters

element	Element whose ID is set.
buffer	Pointer to a buffer containing a string to be made the element's ID.
nBytes	Number of bytes in buffer to use as element 's new ID. May be zero. Sets an ID even if the buffer length is zero, but such an ID looks like no ID according to PDSElementGetID .

Return Value

None

Known Exceptions

Raises [ErrSysPDSEdit](#) if another element already has **id** as its ID.

Raises [pdsErrWrongTypeParameter](#) if **element** is not a valid **PDSElement**.

Header File

PDSWriteCalls.h

Related Methods

[PDSElementGetID](#)

[PDSElementClearID](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementSetLanguage

```
void PDSElementSetLanguage (PDSElement element,
    const ASUns8 *buffer, ASInt32 nBytes);
```

Description

Sets the language field associated with the **PDSElement** to **buffer**'s contents (from 0 to **nBytes**).

Parameters

element	The PDSElement whose language field is set to buffer .
buffer	Pointer to a buffer containing a string to be made the element's language field. The empty string indicates that the language is unknown. String should be in format <IETF RFC-1766-language-code>. NOTE: ISO 639 language codes can be found at: http://lcweb.loc.gov/standards/iso639-2 NOTE: IANA registered language codes can be found at: http://www.isi.edu/in-notes/iana/assignments/languages NOTE: The IETF Standard for Language Element Values (RFC 1766) can be found at: http://www.ietf.org/rfc/rfc1766.txt?number=1766
nBytes	Size of buffer. May be zero. Sets the language even if the buffer length is zero, but such a language setting looks like no language according to PDSElementGetLanguage .

Return Value

None

Header File

PDSWriteCalls.h

Related Methods

[PDSElementGetLanguage](#)
[PDSElementHasLanguage](#)

Product

base, pro, pdf



Availability

Available if **PI_PDS_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDSElementSetTitle

```
void PDSElementSetTitle (PDSElement element,  
    const ASUns8* buffer, ASInt32 nBytes);
```

Description

Sets an element's title.

Parameters

element	Element whose title is set.
buffer	Pointer to a buffer containing a string to be made the element's title.
nBytes	Number of bytes in buffer to use as element 's new title. May be zero. Sets a title even if the buffer length is zero, but such a title looks like no title according to PDSElementGetTitle .

Return Value

None

Known Exceptions

Raises [pdsErrWrongTypeParameter](#) if **element** is not a valid **PDSElement**.

Header File

PDSWriteCalls.h

Related Methods

[PDSElementGetTitle](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSElementSetType

```
void PDSElementSetType (PDSElement element, ASAtom type);
```

Description

Sets an element’s type value to the specified type. The type corresponds to the **Subtype** key in the structure element dictionary.

PDSElementSetType sets the value of the **Subtype** key—not the **Type** key—in the structure element dictionary. All **PDSElements** have a **Type** value of **StructElem**.

Parameters

element	Element whose type is set.
type	The ASAtom representing the element’s type.

Return Value

None

Known Exceptions

Raises **pdsErrWrongTypeParameter** if **element** is not a valid **PDSElement**.

Header File

PDSWriteCalls.h

Related Methods

[PDSElementGetType](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSMC

PDSMCGetInfo

```
void PDSMCGetInfo (CosObj containingObj, PDSMC mc,  
PDSMCInfoP info);
```

Description

Gets information about how the specified marked content is contained in its parent.

NOTE: This method cannot access marked content inside a Form XObject.

Parameters

containingObj	The CosObj containing the MC whose information is obtained. For marked content on a page, this is the Cos object representing the page. For marked content elsewhere, this is the stream in which the marked content resides.
mc	The marked content whose information is obtained.
info	<i>(Filled by the method)</i> A pointer to a structure that the method fills with information about the containingObj .

Return Value

None

Known Exceptions

Raises **pdsErrBadPDF** if an error is found in the PDF file. Will also raise the error if the **PDSMC** passed to it is not in the structure tree.

Header File

PDSReadCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.

PDSMCGetParent

```
void PDSMCGetParent (CosObj containingObj, PDSMC mc,
PDSElement* parent);
```

Description

Gets the parent element of the specified marked content.

Parameters

containingObj	The CosObj containing the MC whose parent is obtained. For marked content on a page, this is the Cos object representing the page. For marked content elsewhere, this is the stream in which the marked content resides.
mc	The marked content whose parent is obtained.
parent	<i>(Filled by the method)</i> Parent element of containingObj .

Return Value

None

Known Exceptions

Raises [pdsErrBadPDF](#) if an error is found in the PDF file. Will also raise the error if the [PDSMC](#) passed to it is not in the structure tree.

Header File

PDSReadCalls.h

Related Methods

[PDSElementGetParent](#)
[PDSElementInsertMCAsKid](#)
[PDSElementRemoveKidMC](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in [PIRequir.h](#)) is set to **0x00040000** or higher.

PDSMCGetPDEContainer

```
PDEContainer PDSMCGetPDEContainer (PDSMC mc);
```

Description

Gets the PDE container object for the specified marked content.

Parameters

mc	The marked content whose container is obtained.

Return Value

The PDE container object.

Header File

`PDSReadCalls.h`

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

PDSMCIDGetParent

```
ASBool PDSMCIDGetParent (ASInt32 mcid, CosObj containingObj,
PDSElement* parent);
```

Description

Gets the parent element of the specified marked content, referred to by its containing object and marked-content identifier.

Parameters

mcid	The identifier (MCID) of the marked content whose parent is obtained.
containingObj	The CosObj containing the marked content whose parent is obtained. For marked content on a page, this is the Cos object representing the page. For marked content elsewhere, this is the stream in which the marked content resides.
parent	(Filled by the method) Parent element of containingObj.

Return Value

true if the parent is successfully obtained, false otherwise.

Known Exceptions

Raises pdsErrrBadPDF if an error is found in the PDF file. Will also raise the error if the PDSMC passed to it is not in the structure tree.

Header File

PDSReadCalls.h

Related Methods

```
PDSMCGetParent
PDSElementGetParent
```

Product

reader, base, pro, pdf

Availability

Available if PI_PDS_READ_VERSION (in PIRequir.h) is set to 0x00060000 or higher.

PDSOBJR

PDSOBJGetParent

```
void PDSOBJGetParent (CosObj obj, PDSElement* parent);
```

Description

Gets the parent element of the specified PDF object.

Parameters

obj	PDF object whose parent element is obtained. Must be referred to via an OBJR from some element (that is, it has a struct parent key), otherwise undefined.
parent	<i>(Filled by the method)</i> Parent element of obj .

Return Value

None

Known Exceptions

Various

Header File

PDSReadCalls.h

Related Methods

[PDSElementGetParent](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSRoleMap

PDSRoleMapCopy

```
void PDSRoleMapCopy (PDSRoleMap srcRoleMap,  
PDS_TreeRoot dstTreeRoot, PDSRoleMap* dstRoleMap);
```

Description

Makes a copy of a **PDSRoleMap**, making it the **PDSRoleMap** of the specified **StructTreeRoot**.

Parameters

srcRoleMap	The PDSRoleMap to copy.
dstTreeRoot	The structure tree root in which to place srcRoleMap .
dstRoleMap	(Filled by the method) If not NULL , points to the new, copied PDSRoleMap .

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

[PDS_TreeRootGetRoleMap](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSRoleMapDoesMap

```
ASBool PDSRoleMapDoesMap (PDSRoleMap roleMap, ASAtom src,
ASAtom dst) ;
```

Description

Determines whether the specified **PDSRoleMap** provides any mapping path for two given element types.

Parameters

roleMap	The PDSRoleMap .
src	The ASAtom for an element type whose mapping is tested.
dst	The ASAtom for an element type. NOTE: This may be a standard element type.

Return Value

true if an mapping path was found, **false** otherwise.

Known Exceptions

Raises **pdsErrBadPDF** if an error is found in the PDF file.

Header File

PDSReadCalls.h

Related Methods

[PDSRoleMapMap](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSRoleMapGetDirectMap

```
ASAtom PDSRoleMapGetDirectMap (PDSRoleMap roleMap,
ASAtom type);
```

Description

Gets the type, if any, directly mapped in the specified **PDSRoleMap** for the given element type.

Parameters

roleMap	The PDSRoleMap .
type	The ASAtom for an element type whose mapping is found.

Return Value

The **ASAtom** for the equivalent type specified in **roleMap**, or **ASAtomNull** if **type** has no mapping in **roleMap**.

Known Exceptions

Raises **pdsErrBadPDF** if an error is found in the PDF file.

Header File

PDSReadCalls.h

Related Methods

[PDSRoleMapDoesMap](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSRoleMapMap

```
void PDSRoleMapMap (PDSRoleMap roleMap, ASAtom src,  
ASAtom dst);
```

Description

Maps an element type **src** to another element type **dst** in the specified **PDSRoleMap**.

Parameters

roleMap	The PDSRoleMap in which to create a new mapping.
src	Element type to map to dst .
dst	Element type that src maps onto. NOTE: This may be a standard element type, such as P .

Return Value

None

Known Exceptions

Raises [ErrSysPDSEdit](#) if **src** is already mapped.

Header File

PDSWriteCalls.h

Related Methods

[PDSRoleMapDoesMap](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSRoleMapUnMapDst

```
void PDSRoleMapUnMapDst (PDSRoleMap roleMap, ASAtom dst);
```

Description

Makes the specified element type have no mapping.

Parameters

roleMap	The PDSRoleMap in which to un-map all element types that map onto the dst element type.
dst	Element type to which all mappings are removed. All element types that map to the dst element type are unmapped.

Return Value

None

Known Exceptions

[pdsErrWrongTypeParameter](#)

Header File

PDSWriteCalls.h

Related Methods

[PDSRoleMapUnMapSrc](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSRoleMapUnMapSrc

```
void PDSRoleMapUnMapSrc (PDSRoleMap roleMap, ASAtom src,
ASBool fixupOthers);
```

Description

Makes the specified element type have no mapping.

Parameters

roleMap	The PDSRoleMap in which to un-map the src element type.
src	Element type whose mapping is removed.
fixupOthers	If true , any element type that was directly mapped to src is mapped to whatever src previously mapped to. If false , PDSRoleMapUnMapSrc only un-maps src .

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

[PDSRoleMapUnMapDst](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.



PDSTreeRoot

PDSTreeRootCreateClassMap

```
void PDSTreeRootCreateClassMap (PDSTreeRoot treeRoot,  
    PDSClassMap* classMap);
```

Description

Creates a **PDSClassMap** in the specified tree root.
Any previously existing **PDSClassMap** is unlinked.

Parameters

treeRoot	The structure tree root in which to create a PDSClassMap .
classMap	<i>(Filled by the method)</i> The newly created PDSClassMap .

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

[PDSTreeRootGetClassMap](#)
[PDSTreeRootRemoveClassMap](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSTreeRootCreateRoleMap

```
void PDSTreeRootCreateRoleMap (PDSTreeRoot treeRoot,  
    PDSRoleMap* roleMap);
```

Description

Creates and sets the **PDSRoleMap** of the specified **StructTreeRoot** element. Any previously existing **PDSRoleMap** is unlinked.

Parameters

treeRoot	The structure tree root in which to create a PDSRoleMap .
roleMap	<i>(Filled by the method)</i> The newly created PDSRoleMap .

Return Value

None

Known Exceptions

Various

Header File

PDSTreeRoot.h

Related Methods

[PDSTreeRootGetRoleMap](#)
[PDSTreeRootRemoveRoleMap](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSTreeRootGetClassMap

```
ASBool PDSTreeRootGetClassMap (PDSTreeRoot treeRoot,
PDSClassMap* classMap);
```

Description

Gets the **PDSClassMap** object for the specified structure tree root.

Parameters

treeRoot	The structure tree root whose PDSClassMap is obtained.
classMap	<i>(Filled by the method)</i> Pointer to a location in which to return the class map, if one exists. Set to CosNull if there is no class map. If a NULL pointer is passed, no retrieval will take place.

Return Value

true if there is a class map, **false** otherwise.

Known Exceptions

Various

Header File

PDSReadCalls.h

Related Methods

[PDSTreeRootCreateClassMap](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSTreeRootGetElementFromID

```
ASBool PDSTreeRootGetElementFromID (PDSTreeRoot treeRoot,
const char* id, ASInt32 numChars, PDSElement* element);
```

Description

Gets the element associated with the given ID, if any.

Parameters

treeRoot	The structure tree root in which to search for id .
id	Pointer to a buffer containing the ID to search for.
numChars	Number of characters in id .
element	<i>(Filled by the method)</i> The element corresponding to id . Undefined if no element has the specified id .

Return Value

true if an element for **id** found, or **false** with **element** undefined if the tree root contains no **IDTree** value.

Known Exceptions

Raises [pdsErrWrongTypeParameter](#) if **id** is **NULL** or **numChars** is zero or less.

Raises [pdsErrWrongTypeEntry](#) if the **IDTree** value in **treeRoot** is not a dictionary.

Header File

PDSReadCalls.h

Related Methods

[PDSElementGetID](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSTreeRootGetKid

```
void PDSTreeRootGetKid (PDSTreeRoot treeRoot, ASInt32 index,
    PDSElement* kid);
```

Description

Gets the kid at an array index in the specified structure tree root.

Parameters

treeRoot	The structure tree root whose kid is obtained.
index	Index of the kid to obtain.
kid	(Filled by the method) Pointer to the kid at index.

Return Value

None

Known Exceptions

Raises [pdsErrBadPDF](#) if an error is found in the PDF file.

Header File

PDSReadCalls.h

Related Methods

[PDSTreeRootGetNumKids](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSTreeRootGetNumKids

```
ASInt32 PDSTreeRootGetNumKids (PDSTreeRoot treeRoot);
```

Description

Gets the number of kids of the structure tree root.

Parameters

treeRoot	The structure tree root whose number of kids is obtained.

Return Value

The number of kids of the structure tree root.

Known Exceptions

Various

Header File

PDSReadCalls.h

Related Methods

[PDSTreeRootGetKid](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDSTreeRootGetRoleMap

```
ASBool PDSTreeRootGetRoleMap (PDSTreeRoot treeRoot,
                               PDSRoleMap* roleMap);
```

Description

Gets the **PDSRoleMap** object for the specified structure tree root.

Parameters

treeRoot	The structure tree root whose PDSRoleMap is obtained.
roleMap	<i>(Filled by the method)</i> Pointer to a location in which to return the role map, if one exists. Set to CosNull if there is no role map. If a NULL pointer is passed, no retrieval will take place.

Return Value

true if there is a role map, **false** otherwise.

Known Exceptions

Various

Header File

PDSReadCalls.h

Related Methods

[PDSTreeRootCreateRoleMap](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_PDS_READ_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSTreeRootInsertKid

```
void PDSTreeRootInsertKid (PDSTreeRoot treeRoot,  
    PDSElement kid, ASInt32 insertAfter);
```

Description

Inserts the specified kid element after the given position as a kid of the specified structure tree root.

Parameters

treeRoot	The structure tree root in which a kid is inserted.
kid	The kid to insert.
insertAfter	Position after which the kid is inserted. If element currently has no kids, insertAfter is ignored.

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

[PDSTreeRootRemoveKid](#)
[PDSTreeRootReplaceKid](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSTreeRootRemoveClassMap

```
void PDSTreeRootRemoveClassMap (PDSTreeRoot treeRoot);
```

Description

Removes the **PDSCClassMap** of the specified structure tree root element. Does nothing if one does not exist.

Parameters

treeRoot	The structure tree root whose PDSCClassMap is removed.

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

[PDSTreeRootCreateClassMap](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSTreeRootRemoveKid

```
void PDSTreeRootRemoveKid (PDSTreeRoot treeRoot,
    PDSElement kid);
```

Description

Removes the specified kid element from the specified structure tree root.

Parameters

treeRoot	The structure tree root whose kid is removed.
kid	The kid to remove.

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

[PDSTreeRootInsertKid](#)
[PDSTreeRootReplaceKid](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

PDSTreeRootRemoveRoleMap

```
void PDSTreeRootRemoveRoleMap (PDSTreeRoot treeRoot);
```

Description

Removes the **PDSRoleMap** of the specified structure tree root element. Does nothing if one does not exist.

Parameters

treeRoot	The structure tree root whose PDSRoleMap is removed.

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

[PDSTreeRootCreateRoleMap](#)

[PDSTreeRootGetRoleMap](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSTreeRootReplaceKid

```
void PDSTreeRootReplaceKid (PDSTreeRoot treeRoot,
    PDSElement oldKid, PDSElement newKid);
```

Description

Replaces structural element **oldKid** with element **newKid** as a kid of **treeRoot**.

Parameters

treeRoot	The structure tree root whose kid is replaced.
oldKid	The kid to replace.
newKid	The kid that is replacing oldKid

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Related Methods

[PDSTreeRootInsertKid](#)
[PDSTreeRootRemoveKid](#)

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00040000** or higher.

PDSTreeRootReplaceStreamRef

```
void PDSTreeRootReplaceStreamRef (PDSTreeRoot treeRoot,  
    CosObj oldStream, CosObj newStream);
```

Description

Updates the stream entries (**Stm**) in marked content reference dictionaries to reference a new Cos stream object. Replaces references to the old stream with references to the new stream.

Parameters

treeRoot	The structure tree root in which stream references are updated.
oldStream	The stream reference to replace.
newStream	The stream reference that is replacing oldStream .

Return Value

None

Known Exceptions

Various

Header File

PDSWriteCalls.h

Product

base, pro, pdf

Availability

Available if **PI_PDS_WRITE_VERSION** (in **PIRequir.h**) is set to **0x00060000** or higher.



PDSTreeRootReplaceStreamRef



Macintosh-specific Methods

AVAppDidOrWillSwitchForDialog

ASBool AVAppDidOrWillSwitchForDialog (void);

Description

This call is useful when Acrobat is running embedded in a browser and needs to present a dialog for user input. However, before presenting the dialog, Acrobat needs to pull itself to the front to be the front application. This call enables it to do that.

Parameters

None

Return Value

true if Acrobat either is running in the background and will pull itself to the front or has already pulled itself to be the front application; **false** otherwise.

Header File

AVCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_ACROVIEW_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

AVAppEnumSystemFonts

```
void AVAppEnumSystemFonts (ASInt32 flags,  
AVSystemFontEnumProc enumProc, void* clientData);
```

Description

Enumerates a list of fonts that are installed in the system. These are the fonts that are available for use by Acrobat. This method does *not* enumerate fonts that have been extracted or fauxed by Acrobat.

Parameters

flags	For future expansion. Must be zero.
enumProc	Client provided callback to call for each system font.
clientData	Client data for enumProc .

Return Value

None

Header File

MacCalls.h

Related Methods

[PDEnumSysFonts](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_MACINTOSH_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.

AVAppHandleAppleEvent

```
void AVAppHandleAppleEvent (AppleEvent* appleEvent,
DescType eventClass, DescType eventId, AppleEvent* replyEvent,
OSErr* err);
```

Description

Handles Apple events that are sent to the Acrobat viewer. Clients that wish to handle their own Apple events do so by using **HFTReplaceEntry** to replace this method. If your replacement method does not wish to handle an Apple event it receives, it must pass the Apple event along to the Acrobat viewer by using the **CALL_REPLACED_PROC** macro.

For further information, see “Application-Defined Routines” in the “Responding to Apple Events” chapter of *Inside Macintosh: Interapplication Communication*.

If a client wishes to get/replace any of the four required Apple events, it must use Macintosh toolbox calls such as **AEGetEventHandler** and **AESetEventHandler**.

Clients can also support AppleScript; the Acrobat viewer contains an ‘scsz’ resource, and clients can handle the **GetAETE** event to append information about scriptable events they provide.

Parameters

appleEvent	The Apple event that was sent to the Acrobat viewer.
eventClass	The event class of appleEvent .
eventId	The event ID of appleEvent .
replyEvent	An event to send back to the application that sent appleEvent . Add parameters to this event, if appropriate.
err	A value indicating whether the Apple event was handled successfully. Return the value noErr (defined by Apple, not in the Acrobat SDK) if the Apple event was handled successfully. See <i>Inside Macintosh: Interapplication Communication</i> . err is the value returned by the function MyEventHandler in that section. In the Acrobat core API, it is returned as a parameter rather than as a function’s return value.

Return Value

None

Header File

MacCalls.h



Product

reader, base, pro, pdf

Availability

Available if `PI_MACINTOSH_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.



AVRectToRect

```
void AVRectToRect (const AVRect* avr, Rect* rect);
```

Description

Converts an **AVRect** into a QuickDraw **rect**.

Parameters

avr	Pointer to the AVRect to convert to a Rect .
rect	<i>(Filled by the method)</i> The Rect corresponding to avr .

Return Value

None

Known Exceptions

Numerous

Notifications

Numerous

Header File

MacCalls.h

Related Methods

[RectToAVRect](#)

Product

reader, base, pro, pdf

Availability

Available if **PI_MACINTOSH_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



AVWindowGetCursorAtPoint

```
AVCursor AVWindowGetCursorAtPoint (AVWindow win, ASInt32 xHit,
ASInt32 yHit);
```

Description

Queries the **AVWindow** for the appropriate cursor for display at the given point (in the **AVWindow**'s device coordinate space).

Parameters

win	The AVWindow to query.
xHit	The x-coordinate of a point in win .
yHit	The y-coordinate of a point in win .

Return Value

The **AVCursor** appropriate for the specified point in the window.

Known Exceptions

Various, depending on the method called.

Header File

AVCalls.h

Product

reader, base, pro, pdf

Availability

Available if **PI_MACINTOSH_VERSION** (in **PIRequir.h**) is set to **0x00020002** or higher.



RectToAVRect

```
void RectToAVRect (const Rect* rect, AVRect* avr);
```

Description

Converts a Quickdraw **Rect** into an **AVRect**.

Parameters

rect	Pointer to a Rect to convert to an AVRect .
avr	<i>(Filled by the method)</i> Pointer to the AVRect corresponding to rect .

Return Value

None

Header File

MacCalls.h

Related Methods

[AVRectToRect](#)

Product

reader, base, pro

Availability

Available if **PI_MACINTOSH_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

UNIX-specific Methods

UnixAppAddModifierCallback

```
void UnixAppAddModifierCallback (XtCallbackProc callback,  
                                XtPointer closure);
```

Description

Registers a callback to call when a KeyEvent that specifies a modifier key is dispatched. The modifier state can be queried with [AVSysGetModifiers](#) or [XQueryPointer](#).

The callback will be called even though the state of the modifiers may not have actually changed. For example, if both the left and right control keys are depressed, then releasing one of them causes the callbacks to be called, even though the state has not changed.

Parameters

callback	The procedure to call. Must be declared in the form: <pre>static void UnixPageViewModifierCallback (Widget widget, XtPointer clientData, XtPointer callData)</pre> where callData is an eventPtr , and widget is the Acrobat viewer's shell widget, as returned by UnixAppGetAppShellWidget .
closure	User-supplied data to pass (as clientData) to proc each time it is called.

Return Value

None

Known Exceptions

[genErrNoMemory](#)

Header File

UnixCalls.h

Related Methods

[UnixAppRemoveModifierCallback](#)

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to 0x00020000 or higher.

UnixAppClipboardGetItemId

```
long UnixAppClipboardGetItemId (Display** displayPtr,  
                               Window* windowPtr);
```

Description

Gets the `item_id`, display, and window needed to call `XmClipboardCopy` during a Copy or Cut operation of a custom selection server (see [AVDocSelectionServer](#)).
The Acrobat viewer will have already started the Motif Clipboard Copy or Cut operation by calling `XmClipboardStartCopy` before calling the selection server's [AVDocSelectionCopyProc](#) or [AVDocSelectionCutProc](#) function. When the Copy or Cut function returns, the Acrobat viewer calls `XmClipboardEndCopy`.

Parameters

<code>displayPtr</code>	(Filled by the method) The clipboard's display.
<code>windowPtr</code>	(Filled by the method) The clipboard's window.

Return Value

The `item_id`

Header File

UnixCalls.h

Product

reader, base, pro

Availability

Available if `PI_UNIX_VERSION` (in `PIRequir.h`) is set to `0x00020000` or higher.



UnixAppDispatchEvent

```
ASBool UnixAppDispatchEvent (XEvent* eventPtr);
```

Description

A wrapper function for **XtDispatchEvent**.

The Acrobat viewer needs to look at every event before it is dispatched by **libXt**. If a client needs to dispatch events, it must dispatch them by either calling **UnixAppDispatchEvent** or **UnixAppProcessEvent**.

Parameters

eventPtr	The event.

Return Value

true if the event was dispatched, **false** if no handler was found to dispatch the event to.

Header File

UnixCalls.h

Related Methods

[UnixAppProcessEvent](#)

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

UnixAppGetAppShellWidget

```
Widget UnixAppGetAppShellWidget (void);
```

Description

Gets the application shell widget created when the Acrobat viewer called **XtAppInitialize**. The widget can be used to get the display, screen, **XtAppContext**, and so forth. Each client should create its own application shell widget, which will be the parent widget for all dialogs the client creates.

Parameters

None

Return Value

The Acrobat viewer's application shell widget.

Header File

UnixCalls.h

Related Methods

[UnixAppAddModifierCallback](#)

Example

```
#define PLUG_IN_CLASS "Foo"

static String fooFallbackResources[] = {
    "Foo*XmText.background:green",
    "Foo*shadowThickness:5",
    NULL
};

static Widget fooAppShellWidget;

FooInit()
{
    Widget appShellWidget = UnixAppGetAppShellWidget();

    ...

    UnixAppLoadPlugInAppDefaults(PLUG_IN_CLASS, fooFallbackResources);

    fooAppShellWidget = XtVaAppCreateShell(
        XtName(appShellWidget), PLUG_IN_CLASS,
        applicationShellWidgetClass, XtDisplay(appShellWidget),
        XmNmappedWhenManaged, False, XmNx,
        WidthOfScreen(XtScreen(appShellWidget)) / 2, XmNy,
```

```
HeightOfScreen(XtScreen(appShellWidget)) / 2, XmNwidth, 1,XmNheight, 1,  
NULL);  
  
XtRealizeWidget(fooAppShellWidget);  
  
...  
}
```

Note, setting the x, y, width and height will cause all dialogs to be centered on the screen. Also, setting **XmMappedWhenManaged** to **false** makes it invisible (that is, unmapped).

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

UnixAppGetPlugInFilename

```
char* UnixAppGetPlugInFilename (ASExtension thePI);
```

Description

Gets the client's filename. The directory can be used to find auxiliary files for that client.

The Acrobat viewer searches all the directories specified in the **systemPlugInPath** and **userPlugInPath** resources to find clients.

Parameters

thePI	The gExtensionID extension registering the client.

Return Value

The client's filename. The string returned must not be altered or freed.

Header File

UnixCalls.h

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

UnixAppLoadPlugInAppDefaults

```
void UnixAppLoadPlugInAppDefaults (String className,
    String* fallbackResources);
```

Description

Loads the system and user application defaults for a client into the screen resource database.

The screen resource database is shared by all clients, and the Acrobat viewer, so name space is important.

Application default filenames are generated from the **XFILESEARCHPATH** and **XUSERFILESEARCHPATH** environment variables, and the **className** parameter.

Parameters

className	Should be the same as the class name used to create the client's application shell widget.
fallbackResources	NULL -terminated array of resource specification strings to use if the system application default file is not found. See XtAppInitialize for more information. fallbackResources should always start with the class name of the client so they do not interfere with other clients, or Acrobat's resources. That is why it is suggested to create an application shell widget for each client, with a unique class name.

Return Value

None

Known Exceptions

[genErrNoMemory](#)

Header File

UnixCalls.h

Related Methods

[UnixSysPrefInit](#)

[UnixSysPrefUpdate](#)

Example

```

#define PLUG_IN_CLASS "Foo"

static String fooFallbackResources[] = {
    "Foo*XmText.background:green",
    "Foo*shadowThickness:5",
    NULL
};

static Widget fooAppShellWidget;

FooInit()
{
    Widget appShellWidget = UnixAppGetAppShellWidget();

    ...

    UnixAppLoadPlugInAppDefaults(PLUG_IN_CLASS, fooFallbackResources);

    fooAppShellWidget = XtVaAppCreateShell(
        XtName(appShellWidget), PLUG_IN_CLASS,
        applicationShellWidgetClass, XtDisplay(appShellWidget),
        XmNmappedWhenManaged, False, XmNx,
        WidthOfScreen(XtScreen(appShellWidget)) / 2, XmNy,
        HeightOfScreen(XtScreen(appShellWidget)) / 2, XmNwidth, 1, XmNheight, 1,
        NULL);

    XtRealizeWidget(fooAppShellWidget);

    ...
}

```

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

UnixAppProcessEvent

```
void UnixAppProcessEvent (XtAppContext appContext,
                          XtInputMask mask);
```

Description

A wrapper function for **XtAppProcessEvent**.
The Acrobat viewer needs to look at every event before it is dispatched by **libXt**. If a client needs to dispatch events, it must dispatch them by either calling [UnixAppDispatchEvent](#) or **UnixAppProcessEvent**.

Parameters

appContext	The application context that identifies the application; same as the parameter passed to XtAppProcessEvent .
mask	A mask specifying which events to handle; same as the parameter passed to XtAppProcessEvent .

Return Value

None

Header File

UnixCalls.h

Related Methods

[UnixAppDispatchEvent](#)

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

UnixAppRemoveModifierCallback

```
void UnixAppRemoveModifierCallback (XtCallbackProc callback,
                                   XtPointer closure);
```

Description

Removes a callback added by [UnixAppAddModifierCallback](#). Both **closure** and **callback** must match the values passed in the call to [UnixAppAddModifierCallback](#).

Parameters

callback	The callback to remove.
closure	User-supplied data that was passed in the call to UnixAppAddModifierCallback .

Return Value

None

Header File

UnixCalls.h

Related Methods

[UnixAppAddModifierCallback](#)

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

UnixAppWaitForWm

```
void UnixAppWaitForWm (Widget shellWidget);
```

Description

Dispatches events until the shell is mapped or the time out specified by the shell's **XtNwaitForWm** and **XmNwmTimeout** has expired.

This method can be used to wait for the window manager to map the shell window when bringing up a dialog.

Parameters

shellWidget	Must be a ShellWidgetClass widget, and it must be realized.

Return Value

None

Header File

UnixCalls.h

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

UnixMachinePortGetOffscreenPixmap

```
Boolean UnixMachinePortGetOffscreenPixmap (Widget widget,
      Pixmap *pixmapPtr, int *xOffsetPtr, int *yOffsetPtr);
```

Description

Must be used in conjunction with [AVPageViewAcquireMachinePort](#). If offscreen drawing is in effect, this method returns **true** and clients should draw directly onto the display using XtWindow to obtain the Window associated with **widget**.

When Acrobat is updating a region within the document window, the values of **xOffsetPtr** and **yOffsetPtr** define the offset from the origin of the window to the origin of the pixmap.

Parameters

widget	The Widget returned from the call to AVPageViewAcquireMachinePort .
pixmapPtr	<i>(Filled by the method)</i> The pixmap that the client should draw onto.
xOffsetPtr	<i>(Filled by the method)</i> The X coordinate of the origin offset.
yOffsetPtr	<i>(Filled by the method)</i> The Y coordinate of the origin offset.

Return Value

true if the offscreen drawing is in effect, **false** otherwise.

Known Exceptions

None.

Header File

UnixCalls.h

Product

reader, base, pro

Availability

First available with Acrobat 5. See the release notes for details.

UnixSysGetConfigName

```
char* UnixSysGetConfigName (void);
```

Description

Gets the Acrobat viewer's configuration name. The name is one of the following:

- SunOS — **sparcsun**
- Solaris — **sparcsolaris**
- HP-UX — **hppahpux**

The Acrobat viewer's launch shell script uses **uname** to set the configuration name in the environment variable **ACRO_CONFIG**.

Auxiliary files can be found by concatenating the installation directory with the configuration name sub-directory:

```
installation_dir/config_name
```

Parameters

None

Return Value

The Acrobat viewer's configuration name.

Header File

```
UnixCalls.h
```

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

UnixSysGetCursor

```
Cursor UnixSysGetCursor (Widget widget, char* resourceName,
    char* defaultName, char* defaultBits,
    char* defaultMaskBits, unsigned int defaultWidth,
    unsigned int defaultHeight, int defaultHotX,
    int defaultHotY);
```

Description

A convenience method for getting internationalized cursors. It sets up a **XtResource** specification with the **resourceName** and **defaultName** parameters, then calls **XtGetSubresources** with the name **cursor** and class **Cursor**.

The cursor name returned by **XtGetSubresources** is used to find the cursor bitmap by calling **XmGetPixmapByDepth**, which first checks the Motif image cache. If not found in the image cache, then **XmGetPixmapByDepth** searches for an **xbm** file. If the cursor name is not an absolute filename (that is, does not start with '/'), then **XmGetPixmapByDepth** uses the environment variable **XBMLANGPATH** to find it.

If a bitmap is found, then **UnixSysGetCursor** appends **Mask** to the bitmap name to find the cursor mask bitmap, otherwise it uses the **defaultBits** and **defaultMaskBits** to create a cursor.

This method does not cache its results; a new cursor is created each time it is called.

Parameters

widget	The widget to pass to XtGetSubresources .
resourceName	Used to create an XtResource specification to locate the cursor.
defaultName	Used to create an XtResource specification to locate the cursor.
defaultBits	Bitmap used for the cursor if the requested cursor cannot be found. It is passed to XCreateBitmapFromData .
defaultMaskBits	Bitmap used as the cursor mask if the requested cursor cannot be found. It is passed to XCreateBitmapFromData .
defaultWidth	Cursor width if the requested cursor cannot be found. It is passed to XCreateBitmapFromData .
defaultHeight	Cursor width if the requested cursor cannot be found. It is passed to XCreateBitmapFromData .
defaultHotX	The x-coordinate of the cursor's hot spot if the requested cursor cannot be found. It is passed to XCreatePixmapCursor .



defaultHotY	The y-coordinate of the cursor's hot spot if the requested cursor cannot be found. It is passed to XCreatePixmapCursor .

Return Value

The requested cursor.

Known Exceptions

[genErrNoMemory](#)

Header File

UnixCalls.h

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



UnixSysGetCwd

```
char* UnixSysGetCwd (void);
```

Description

Gets the current working directory. This method tries to eliminate automounter **tmp** directories and symbol links. Using **stat**, it checks if the environment variable **PWD** specifies the same directory returned by **getcwd**.

Parameters

None

Return Value

The current working directory.

Known Exceptions

[genErrNoMemory](#)

Header File

UnixCalls.h

Related Methods

[UnixSysGetHomeDirectory](#)

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

UnixSysGetHomeDirectory

```
char* UnixSysGetHomeDirectory (void);
```

Description

Gets the home directory of the user running the Acrobat viewer. If the **HOME** environment variable is set, its value is returned. Otherwise the method looks in the **passwd** database.

Parameters

None

Return Value

The user's home directory. The string returned by this method must not be altered or freed.

Header File

UnixCalls.h

Related Methods

[UnixSysGetCwd](#)

[UnixSysGetHostname](#)

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

UnixSysGetHostname

```
char* UnixSysGetHostname (void);
```

Description

Gets the host name.

Parameters

None

Return Value

The host name.

Header File

UnixCalls.h

Related Methods

[UnixSysGetHomeDirectory](#)

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

UnixSysGetIcon

```
Pixmap UnixSysGetIcon (Widget widget, char* resourceName,
                      char* defaultName, char* defaultBits,
                      unsigned int defaultWidth, unsigned int defaultHeight);
```

Description

A convenience method for getting internationalized icons, which are bitmaps (pixel maps with depth 1). Use [UnixSysGetPixmap](#) for pixel maps with depth greater than 1.

This method sets up an **XtResource** specification with the **resourceName** and **defaultName** parameters, then calls **XtGetSubresources** with the name **icon** and class **Icon**.

The icon name returned by **XtGetSubresources** is used to find the icon by calling **XmGetPixmapByDepth**, which first checks the Motif image cache. If not found in the image cache, then **XmGetPixmapByDepth** searches for an **xbm** file. If the cursor name is not an absolute filename (that is, does not start with '/'), then **XmGetPixmapByDepth** uses the environment variable **XBMLANGPATH** to find it.

If an icon is not found, **defaultBits** is used to create an icon.

This method does not cache its results, a new icon will be created each time it is called.

Parameters

widget	Widget, as passed to XtGetSubresources .
resourceName	Used to create an XtResource specification for the cursor.
defaultName	Used to create an XtResource specification for the cursor.
defaultBits	Bitmap used for the icon if the requested icon is not found. Passed to XCreateBitmapFromData .
defaultWidth	Width used for the default icon if the requested icon is not found. Passed to XCreateBitmapFromData .
defaultHeight	Height used for the default icon if the requested icon is not found. Passed to XCreateBitmapFromData .

Return Value

The requested icon.

Header File

UnixCalls.h

Related Methods

[UnixSysGetPixmap](#)

**Product**

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

UnixSysGetInstallDirectory

```
char* UnixSysGetInstallDirectory (void);
```

Description

Gets the directory in which the Acrobat viewer is installed. The launch shell script sets the installation directory in the environment variable **ACRO_INSTALL_DIR**.

Auxiliary files can be found by concatenating the installation directory with the configuration name sub-directory:

```
installation_dir/config_name
```

Parameters

None

Return Value

The Acrobat viewer's installation directory.

Header File

```
UnixCalls.h
```

Related Methods

[UnixSysGetHomeDirectory](#)

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

UnixSysGetPixmap

```
Pixmap UnixSysGetPixmap (Widget widget, char* resourceName,
char* defaultFilename, char** defaultXpmData,
int defaultXpmDataCount, unsigned int* widthPtr,
unsigned int* heightPtr);
```

Description

A convenience method for getting internationalized pixel maps of depth greater than 1. This method is intended for displaying graphics in an About box. The pixel map will be created with the same depth as the widget. Use [UnixSysGetIcon](#) for bitmaps (that is, pixel maps with depth 1).

This method sets up an **XtResource** specification with the **resourceName** and **defaultFilename** parameters, then calls **XtGetSubresources** with the name **pixmap** and class **Pixmap**.

If the pixmap filename returned by **XtGetSubresources** exists and is a valid **xbm** or **xpm** file, then a pixmap will be created. Otherwise **defaultXpmData** is used.

If a gray scale visual (or the application is in grayscale mode because it could not allocate enough colors, see color cube resources for more info), then the color map key **m** will be used in the **xpm** file (or data), otherwise **c** will be used.

XLookupColor is used to translate the colors specified in the **xpm** color map. Because the color map is a valuable shared resource, and **UnixSysGetPixmap** was only intended for the graphics in an About box, **XAllocColor** is not called. Instead the closest color in the Acrobat viewer's color cube is used. Therefore, it is suggested that:

- All xpm files (and data) have both **c** and **m** color map keys, for both color and grayscale modes.
- Specify all colors in hexadecimal form (#rgb) to avoid problems with the Xserver's RGB database.
- Specify only the 8 fully saturated colors (black, white, red, green, ...) #000, #fff, #f00, #0f0, #00f, #ff0, #0ff, #f0f, since only these colors (which are the corners of the color cube) can be guaranteed.

This method does not cache its results, a new pixmap will be created each time it is called.

Parameters

widget	The widget, passed to XtGetSubresources .
resourceName	Used to create an XtResource specification for the pixmap.
defaultFilename	Name of a file containing a default pixmap to use if the requested pixmap is not found. Used to create an XtResource specification for the pixmap.

defaultXpmData	A default pixmap used if the requested pixmap cannot be found.
defaultXpmCount	The number of strings in the defaultXpmData array, typically XtNumber (defaultXpmData) .
widthPtr	Width of the image. Used both if the requested pixmap is found, and for the default pixmap, if the requested pixmap is not found.
heightPtr	Height of the image. Used both if the requested pixmap is found, and for the default pixmap, if the requested pixmap is not found.

Return Value

The requested pixmap.

Header File

UnixCalls.h

Related Methods

[UnixSysGetIcon](#)

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

UnixSysGetString

```
char* UnixSysGetString (Widget widget, char* resourceName,  
                        char* defaultString);
```

Description

A convenience method for getting internationalized strings. It sets up an **XtResource** specification with the **resourceName** and **defaultString** parameters, then calls **XtGetSubresources** with the name **string** and class **String**.

This method does not cache its results.

Parameters

widget	The widget, passed to XtGetSubresources .
resourceName	Used to create an XtResource specification to get the string.
defaultString	Default string, used if the requested string is not found.

Return Value

The requested string, or the default string if the requested string cannot be located.

Header File

UnixCalls.h

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.



UnixSysGetTempFileDirectory

```
char* UnixSysGetTempFileDirectory (void);
```

Description

Gets the temporary file directory specified by the user. The default is /tmp.

Parameters

None

Return Value

The temporary file directory.

Header File

UnixCalls.h

Related Methods

[UnixSysGetHomeDirectory](#)

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

UnixSysPrefInit

```
void UnixSysPrefInit (Widget widget, String name,
    String class, char* prefFilename,
    XtResourceList resources, Cardinal numResources,
    XtPointer base, XrmDatabase* dbPtr,
    void** resourceDataPtr);
```

Description

Reads preferences from the specified file into a resource manager database. The file format is the same as an application defaults file.

The database is then used to initialize the structure pointed to by base, which is described by resources. See **XtGetApplicationResources** for more information on the format of the resources parameter.

UnixSysPrefInit uses string resource converters to convert the string data in the preference file to the appropriate data type, as described in the resources parameter. See **XtConvertAndStore** for a list of standard converters, which are automatically registered. For other data types described in the resources parameter, a converter needs to register with **XtSetTypeConverter** prior to calling **UnixSysPrefInit**.

Furthermore, a reverse string converter needs to register prior to calling **UnixSysPrefUpdate**. The Acrobat viewer has already registered reverse string converters for the following general data types: **XtRBoolean**, **XtRBool**, **XtRDimension**, **XtRFloat**, **XtRInt**, **XtRPosition**, **XtRShort**, **XtRUnsignedChar**.

Also the following Acrobat viewer data types have both forward and reverse string converters:

```
#define XtRBoolean16 "Boolean16" /* boolean */ #define XtRAVZoomType
"AVZoomType"
#define XtRFixed "ASFixed"
#define XtRPDColorValue
"PDColorValue"
#define XtRPDPageMode "PDPageMode"
```

Finally, Motif provides the function **XmRepTypeRegister**, which will create a string converter for a list of strings to an unsigned char. Also, **XmRepTypeAddReverse** will create the reverse string converter.

Parameters

widget	The client's application shell. This value is used for calling XtConvertAndStore .
name	The name used to create the client's application shell. This value is used to match items in the database.

class	The class used to create the client's application shell. This value is used to match items in the database.
prefFilename	The file from which preferences are read. If non- NULL , prefFileName must be a resource file, and its contents are merged into the database pointed to by dbPtr .
resources	List of resources to read from the file.
numResources	Number of resources to read from the file (that is, the number of resources specified in the resources parameter).
base	All string data put into the structure pointed to by base must not be altered or freed. It is suggested that a copy of all string data be made, as shown in the Example.
dbPtr	Pointer to a resource manager database. The database is created from the specified items in the preferences file. Clients typically do not need access to the resource manager database created from the preference file, and can pass NULL for this parameter. If dbPtr is not NULL, it must point to an XrmDatabase variable which is initialized to NULL, or is a valid resource manager database with which the preference file will be combined.
resourceDataPtr	Pointer to a data buffer. Must be a valid pointer. If non- NULL , all resources specified in the resources parameter or contained in the file specified by prefFilename are copied into this buffer.

Return Value

None

Header File

UnixCalls.h

Related Methods[UnixSysPrefUpdate](#)

Example

```

#define PLUG_IN_CLASS "Foo"
#define PREF_FILE ".foorc"

static Widget fooAppShellWidget;

typedef struct {
    Boolean paletteOnLeft;
    String greetingString;
    int grossNationalDebt;
} FooDataRec;

static XtResource fooResources[] = {
    { "paletteOnLeft", "PaletteOnLeft",
      XtRBoolean, sizeof(Boolean),
      XtOffsetOf(FooDataRec, paletteOnLeft),
      XtRImmediate, True },
    { "greetingString", "GreetingString",
      XtRString, sizeof(String),
      XtOffsetOf(FooDataRec, greetingString),
      XtRImmediate, "Hello" },
    { "grossNationalDebt", "GrossNationalDebt", XtRInt, sizeof(int),
      XtOffsetOf(FooDataRec, grossNationalDebt), XtRImmediate, MAX_INT },
};

static void* fooResourceData;
static String fooPrefFilename;
static String fooFallbackResources[] = {
    "Foo*XmText.background:green",
    "Foo*shadowThickness:5",
    NULL
};

FooDataRec fooDataRec;

FooInit()
{
    Widget appShellWidget = UnixAppGetAppShellWidget(); String appName =
    XtName(appShellWidget); String homeDir = UnixSysGetHomeDirectory();
    ...
    UnixAppLoadPlugInAppDefaults(PLUG_IN_CLASS, fooFallbackResources);

    fooAppShellWidget = XtVaAppCreateShell(
        appName, PLUG_IN_CLASS,
        applicationShellWidgetClass, XtDisplay(appShellWidget),
        XmNmappedWhenManaged, False,
        XmNx, WidthOfScreen(XtScreen(appShellWidget)) / 2, XmNy,
        HeightOfScreen(XtScreen(appShellWidget)) / 2, XmNwidth, 1,
        XmNheight, 1,
        NULL);

```

```

XtRealizeWidget(fooAppShellWidget);

fooPrefFilename = malloc(strlen(homeDir) + 1 + strlen(PREF_FILE) + 1);
strcpy(fooPrefFilename, homeDir);
strcat(fooPrefFilename, "/");
strcat(fooPrefFilename, PREF_FILE);

UnixSysPrefInit(fooAppShellWidget, appName, PLUG_IN_CLASS,
fooPrefFilename, fooResources, XtNumber(fooResources), &fooDataRec,
NULL, &fooResourceData);

if (fooDataRec.greetingString != NULL)
{
String tmpString = malloc(strlen(fooDataRec.greetingString) + 1);

strcpy(tmpString, fooDataRec.greetingString); fooDataRec.greetingString
= tmpString;
}

...
}
FooExit()
{
...

UnixSysPrefUpdate(fooAppShellWidget, fooPrefFilename, &fooDataRec,
fooResources, XtNumber(fooResources), fooResourceData);

...
}

```

Product

reader, base, pro

Availability

Available if **PI_UNIX_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

UnixSysPrefUpdate

```
void UnixSysPrefUpdate (Widget widget, char* prefFilename,
                        XtPointer base, XtResourceList resources,
                        Cardinal numResources, void* resourceData);
```

Description

Updates a preference file read in by [UnixSysPrefInit](#). All the remaining parameters are the same as those passed to UnixSysPrefInit.

Data in the preference file are strings. **UnixSysPrefUpdate** uses reverse string converters (for example, int to string) to update the file. Therefore, reverse string converters need to register for all data types listed in resources prior to calling **UnixSysPrefUpdate**. The Acrobat viewer has already registered reverse string converters for the following general data types: **XtRBoolean**, **XtRBool**, **XtRDimension**, **XtRFloat**, **XtRInt**, **XtRPosition**, **XtRShort**, **XtRUnsignedChar**.

and for the following Acrobat viewer data types: **XtRBoolean16**, **XtRAVZoomType**, **XtRFixed**, **XtRPDColorValue**, **XtRPDPageMode**.

Parameters

widget	The client's application shell. This value is used for calling XtConvertAndStore.
prefFileName	The filename containing the preferences.
base	Base.
resources	Resources.
numResources	Number of resources.
resourceData	Must be the data returned by UnixSysPrefInit .

Return Value

None

Header File

UnixCalls.h

Related Methods

[UnixSysPrefInit](#)

Product

reader, base, pro

**Availability**

Available if **PI_UNIX_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



Windows-specific Methods

WinAppEnableIdleTimer

```
ASBool WinAppEnableIdleTimer (ASBool enable);
```

Description

(Windows only) Allows a client to turn the **AVAppIdle** timer on and off, which is needed when a client calls another process and thus blocks Acrobat for an extended period of time.

Parameters

enable	true to turn the timer on, false to turn it off.

Return Value

The previous state of the **AVAppIdle** timer.

Header File

WinCalls.h

Product

reader, base, pro

Availability

Available if **PI_WIN_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

WinAppGetModalParent

```
HWND WinAppGetModalParent (AVDoc doc);
```

Description

(Windows only) Gets appropriate parent for any modal dialogs created by a client. This method is only useful if there is an **AVDoc**; it cannot be used, for example, to put up a modal dialog while a file is being opened.

In circumstances where there is no **AVDoc**, use the **gHWND** provided in `PIMAIN.C`. Although this does not give perfect results in some cases, there is no real alternative. (For example, if a file is opened in an external application's window, the dialog is not hidden if the external application is hidden.)

Parameters

doc	The AVDoc for a PDF file, if the dialog is acting on an PDF document, which is generally the case. The AVDoc must be provided so that for external documents, the viewer can parent the dialog off the external application—instead of the viewer.

Return Value

HWND for modal dialogs' parent.

Header File

`WinCalls.h`

Related Methods

[AVAppBeginModal](#)

[AVAppEndModal](#)

Product

reader, base, pro

Availability

Available if **PI_WIN_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



WinAppGetPalette

```
HPALETTE WinAppGetPalette (void);
```

Description

(Windows only) Gets the application's color palette in the case where the system is running in 256 color mode or less. Used when you want to set and realize a palette in an external window before drawing to it.

Do *not* release this palette handle—it may be in use by other plug-in's.

Parameters

None

Return Value

The application's color palette. **NULL** if the system is running direct colors (15/16/24/32-bit) or no palette is used.

Header File

WinCalls.h

Product

reader, base, pro

Availability

Available if **PI_WIN_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.

WinAppGetPrinterHDC

```
HDC WinAppGetPrinterHDC (void);
```

Description

(Windows only) Gets the device context for a printer, which is the **HDC** used to print a document.

Used if you need to modify the device context Acrobat creates when printing to a non-PostScript printer. You should register for the notification [PDDocWillPrintPage](#) and acquire the printer DC for the page you wish to modify.

Parameters

None

Return Value

The printer device context.

Header File

WinCalls.h

Product

reader, base, pro

Availability

Available if **PI_WIN_VERSION** (in `PIRequir.h`) is set to **0x00040000** or higher.

WinAppRegisterInterface

```
BOOL WinAppRegisterInterface (COMServer);
```

Description

(Windows only) Register a COM interface.

Parameters

COMServer	Pointer to COMServerRec .

Return Value

true if the interface was registered with Acrobat; **false** otherwise.

Header File

WinCalls.h

Product

reader, base, pro

Availability

Available if **PI_WIN_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

WinAppRegisterModelessDialog

```
void WinAppRegisterModelessDialog (HWND dialog);
```

Description

(Windows only) Registers mode-less dialogs with the viewer so that the dialog gets the correct messages.

Parameters

dialog	HWND for the dialog to register.

Return Value

None

Header File

WinCalls.h

Related Methods

[WinAppUnRegisterModelessDialog](#)

Product

reader, base, pro

Availability

Available if **PI_WIN_VERSION** (in **PIRequir.h**) is set to **0x00020000** or higher.

WinAppUnRegisterModelessDialog

```
void WinAppUnRegisterModelessDialog (HWND dialog);
```

Description

(Windows only) Un-registers mode-less dialogs with the viewer.

Parameters

dialog	HWND for the dialog to un-register.

Return Value

None

Header File

WinCalls.h

Related Methods

[WinAppRegisterModelessDialog](#)

Product

reader, base, pro

Availability

Available if **PI_WIN_VERSION** (in `PIRequir.h`) is set to **0x00020000** or higher.



Callbacks

ActivateProcType

```
ACCB1 void ACCB2 ActivateProcType (AVTool tool,
ASBool persistent);
```

Description

Callback for [AVTool](#). Called when this tool has become the active tool. It is legal to call [AVAppGetActiveTool](#) from within this method or from [DeactivateProcType](#); it will return this tool object. Call [AVAppGetLastActiveTool](#) to get the formerly active tool object (its [DeactivateProcType](#) method will already have been called).

Parameters

tool	The tool to activate.
persistent	true if it should remain active through arbitrarily many “operations” (whatever that means for a particular tool), rather than performing a single action (“one shot”) and then restoring the previously active tool, false otherwise.

Return Value

None

Header File

AVExpT.h

Related Callbacks

[DeactivateProcType](#)

AdjustCursorProcType

```
ACCB1 ASBool ACCB2 AdjustCursorProcType (AVTool tool,
AVPageView pageView, ASInt16 x, ASInt16 y);
```

Description

(Optional) Callback for [AVTool](#). This callback controls the cursor shape when the tool is the active tool.

If omitted, the cursor specified in the tool's **cursorID** field is used.

Parameters

tool	The currently active tool.
pageView	The page view in which the cursor is currently located.
x	The x-coordinate of the current cursor location.
y	The y-coordinate of the current cursor location.

Return Value

true if you handled the cursor shape (that is, do not allow underlying layers to handle it), **false** if you did (that is, allow underlying layers to handle it).

Header File

AVExpT.h

Related Methods

[AVAppRegisterTool](#)

ASCabEnumProc

```
ACCB1 ASBool ACCB2 ASCabEnumProc (ASCab theCab,
const char* theKey, ASCabValueType itsType, void* clientData);
```

Description

Used when enumerating the values inside a cabinet.

Parameters

theCab	The cabinet being enumerated.
theKey	The key name of an entry in the cabinet.
itsType	The type of the value associated with theKey .
clientData	User-supplied data that was passed in to ASCabEnum .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

ASExtraExpT.h

Related Methods

[ASCabEnum](#)

ASCabPointerDestroyProc

```
ACCB1 void ACCB2 ASCabPointerDestroyProc (void* ptr);
```

Description

A de-allocation callback that can be associated with a pointer in an [ASCab](#). When the reference count of the pointer falls to zero, this callback is called to free the resources associated with the object the pointer references.

Parameters

ptr	The value stored in an ASCab .

Return Value

None

Header File

ASExtraExpT.h

Related Methods

[ASCabPutPointer](#)

[ASCabGetPointerDestroyProc](#)

ASCallbackCreateProto

[ASCallback](#) AScallbackCreateProto(funcType, proc)

Description

Macro that creates a callback for the specified procedure.

Performs compile-time type-checking of the procedure if the [DEBUG](#) macro is nonzero.

NOTE: Should not be used for notifications and replacements. Use [ASCallbackCreateReplacement](#) and [ASCallbackCreateNotification](#) instead (these macros call [ASCallbackCreateProto](#)).

Parameters

funcType	The type of function being declared. This is used to allow compile-time type checking against the appropriate function prototype.
proc	A pointer to the user-supplied procedure for which a callback is created.

Return Value

A pointer to [ASCallback](#). Needs to be saved so that it can be destroyed after use with the method [ASCallbackDestroy](#).

Header File

CorCalls.h

Related Macros

[ASCallbackCreateReplacement](#)
[ASCallbackCreateNotification](#)
[ACCB1](#)
[ACCB2](#)
[DEBUG](#)

Related Methods

[ASCallbackCreate](#)
[ASCallbackCreateNotification](#)

Example

```
ASCallback pFuncType = AScallbackCreateProto(FuncType, pMyFunction);

/* Use the pointer, then destroy */

ASCallbackDestroy(pFuncType)
```



ASCancelProc

```
ASBool ASCancelProc (void* clientData);
```

Description

This callback replaces [CancelProc](#).

Callback to check for canceling operations. An **ASCancelProc** is typically passed to some method that takes a long time to complete. At frequent intervals, the method calls the **ASCancelProc**. If it returns **true**, then the method cancels its operation; if **false**, it continues.

Parameters

clientData	User-supplied data that was passed to the method that uses the ASCancelProc .

Return Value

true if the processing is to be canceled, **false** otherwise.

Header File

ASExpT.h

Related Callbacks

[PDFLPrintCancelProc](#) (Only available with PDF Library SDK)

Related Methods

[AVAppGetCancelProc](#)
[PDDocCreateThumbs](#)
[PDDocInsertPages](#)

ASConstCabEnumProc

```
ACCB1 ASBool ACCB2 ASConstCabEnumProc (ASConstCab theCab,
const char* theKey, ASCabValueType itsType, void* clientData);
```

Description

Used when enumerating the values inside a constant cabinet. The callback procedure must not add, delete, or modify items in **theCab** during the enumeration.

Parameters

theCab	The cabinet being enumerated.
theKey	The key name of an entry in the cabinet.
itsType	The type of the value associated with theKey .
clientData	User-supplied data that was passed in to ASCabEnum .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

```
ASExtraExpT.h
```

Related Methods

```
ASConstCabEnum
```

ASCryptStmFCloseProc

```
ACCB1 ASInt32 ACCB2 ASCryptStmFCloseProc (ASCryptStm stm);
```

Description

Callback for [ASCryptStm](#). Closes a security stream.

Parameters

stm	The security stream to be closed.

Return Value

0 when successful, otherwise a non-zero error code.

Header File

`ASExpT.h`

Related Callbacks

[ASCryptStmFFlushProc](#)
[ASCryptStmFilBufProc](#)
[ASCryptStmFlsBufProc](#)
[ASCryptStmFPutEOFProc](#)
[ASCryptStmFResetProc](#)
[ASCryptStmUnGetcProc](#)

ASCryptStmFFlushProc

```
ACCB1 ASInt32 ACCB2 ASCryptStmFFlushProc (ASCryptStm stm);
```

Description

Callback for [ASCryptStm](#). Flushes a dirty buffer if necessary.

Parameters

stm	The security stream to be flushed.

Return Value

0 when successful, otherwise a non-zero error code.

Header File

`ASExpT.h`

Related Callbacks

[ASCryptStmFCloseProc](#)
[ASCryptStmFilBufProc](#)
[ASCryptStmFlsBufProc](#)
[ASCryptStmFPutEOFProc](#)
[ASCryptStmFResetProc](#)
[ASCryptStmUnGetcProc](#)

ASCryptStmFilBufProc

```
ACCB1 ASInt32 ACCB2 ASCryptStmFilBufProc (ASCryptStm pstm);
```

Description

Callback for [ASCryptStm](#). Called by `getc` when buffer is empty. Called only during decryption (that is, reading from the stream, not writing).

Parameters

<code>pstm</code>	The security stream to fill.

Return Value

0 when successful, otherwise a non-zero error code.

Header File

`ASExpT.h`

Related Callbacks

[ASCryptStmFCloseProc](#)
[ASCryptStmFFlushProc](#)
[ASCryptStmFlsBufProc](#)
[ASCryptStmFPutEOFProc](#)
[ASCryptStmFResetProc](#)
[ASCryptStmUnGetcProc](#)

ASCryptStmFlsBufProc

```
ACCB1 ASInt32 ACCB2 ASCryptStmFlsBufProc (ASInt32 ch,  

ASCryptStm stm);
```

Description

Callback for [ASCryptStm](#). Called by `putc` when buffer is full. Called only during encryption (that is, writing to the stream, not reading).

Parameters

ch	The character being put to the full stream.
stm	The security stream that is full.

Return Value

0 when successful, otherwise a non-zero error code.

Header File

`ASExpT.h`

Related Callbacks

[ASCryptStmFCloseProc](#)
[ASCryptStmFFlushProc](#)
[ASCryptStmFilBufProc](#)
[ASCryptStmFPutEOFProc](#)
[ASCryptStmFResetProc](#)
[ASCryptStmUnGetcProc](#)

ASCryptStmFPutEOFProc

```
ACCB1 ASInt32 ACCB2 ASCryptStmFPutEOFProc (ASCryptStm stm);
```

Description

Callback for [ASCryptStm](#). Puts an end-of-file marker to a security stream.

Parameters

stm	The security stream to receive the EOF.

Return Value

0 when successful, otherwise a non-zero error code.

Header File

`ASExpT.h`

Related Callbacks

[ASCryptStmFCloseProc](#)
[ASCryptStmFFlushProc](#)
[ASCryptStmFilBufProc](#)
[ASCryptStmFlsBufProc](#)
[ASCryptStmFResetProc](#)
[ASCryptStmUnGetcProc](#)

ASCryptStmFResetProc

```
ACCB1 ASInt32 ACCB2 ASCryptStmFResetProc (ASCryptStm stm) ;
```

Description

Callback for [ASCryptStm](#). Resets a security stream, discarding any buffered data. Called only during encryption (that is, writing to the stream, not reading).

Parameters

stm	The security stream to be reset.

Return Value

0 when successful, otherwise a non-zero error code.

Header File

`ASExpT.h`

Related Callbacks

[ASCryptStmFCloseProc](#)
[ASCryptStmFFlushProc](#)
[ASCryptStmFilBufProc](#)
[ASCryptStmFlsBufProc](#)
[ASCryptStmFPutEOFProc](#)
[ASCryptStmUnGetcProc](#)

ASCryptStmUnGetcProc

```
ACCB1 ASInt32 ACCB2 ASCryptStmUnGetcProc (ASInt32 ch,
ASCryptStm stm) ;
```

Description

Callback for [ASCryptStm](#). Goes back one character in the input stream, undoing a character get operation. Called only during decryption (that is, reading from the stream, not writing).

Parameters

ch	The character being put to the stream.
stm	The security stream to which the character is put.

Return Value

0 when successful, otherwise a non-zero error code.

Header File

```
ASExpT.h
```

Related Callbacks

- [ASCryptStmFCloseProc](#)
- [ASCryptStmFFlushProc](#)
- [ASCryptStmFilBufProc](#)
- [ASCryptStmFlsBufProc](#)
- [ASCryptStmFPutEOFProc](#)
- [ASCryptStmFResetProc](#)

ASExtensionEnumProc

```
ACCB1 ASBool ACCB2 ASExtensionEnumProc (ASExtension extension,
void* clientData);
```

Description

Enumeration function for [ASEnumExtensions](#).

Parameters

extension	The ASExtension for a client.
clientData	User-supplied data that was passed in the call to ASEnumExtensions .

Return Value

If **false**, enumeration halts and the **ASExtension** on which enumeration halted is returned. If **true**, enumeration continues.

Header File

CorExpT.h

Related Methods

[ASEnumExtensions](#)

ASFileCompletionProc

```
ACCB1 void ACCB2 ASFileCompletionProc (ASFile aFile,
const char* p, ASTFilePos fileOffsetRequested,
ASTArraySize countRequested, ASTArraySize nBytesRead,
ASErrorCode error, void* compProcClientData);
```

Description

Called when an asynchronous read or write request has completed.

Parameters

aFile	The ASFile for which data is read or written.
p	Pointer to the buffer provided by the client. Contains nBytesRead of data if error is zero.
fileOffsetRequested	File offset requested by the client.
countRequested	Number of bytes requested by the client.
nBytesRead	Number of bytes actually read or written.
error	Error condition if nonzero; see AcroErr.h .
compProcClientData	Client data parameter provided by client.

Return Value

None

Header File

ASExpT.h

Related Callbacks

[ASFileSysAsyncAbortProc](#)
[ASFileSysAsyncReadProc](#)
[ASFileSysAsyncWriteProc](#)
[ASFileSysYieldProc](#)

ASFileSysAcquireFileSysPathProc

```
ACCB1 ASPathName ACCB2 ASFileSysAcquireFileSysPathProc  
(ASPathName pathName, ASFileSys newFileSys);
```

Description

Callback for [ASFileSysRec](#). Used for non-local file systems. Returns an [ASPathName](#) on the new [ASFileSys](#) that refers to an image (possibly cached) of the remote file. Because of the possibility of cache flushing, you must hold a copy of the remote file's [ASPathName](#) for the duration of use of the local file.

NOTE: Do not remove the local file copy, since the default file system does not know about the linkage to the remote file. Removing this temporary file is left to the file system.

NOTE: The [ASPathName](#) returned should be released with the [ASFileSysReleasePath](#) method when it is no longer needed.

Parameters

pathName	The ASPathName for which an equivalent in newFileSys is obtained.
newFileSys	The file system in which an equivalent of pathName is obtained.

Return Value

The [ASPathName](#) (in **newFileSys**) for the specified file. Returns **NULL** if one can not be made.

Header File

ASExpT.h

Related Callbacks

[ASFileSysCreatePathNameProc](#)

ASFileSysAcquirePlatformPathProc

```
ACCB1 ASInt32 ACCB2 ASFileSysAcquirePlatformPathProc  
(ASPathName path, ASAtom platformPathType,  
ASPlatformPath *platformPath);
```

Description

Callback for [ASFileSysRec](#). Acquires a platform-specific file-system representation of the specified path, according to the specified type, wrapped in an allocated [ASPlatformPath](#) object. Use the [ASPlatformPath*](#) calls to get the actual platform object.

Parameters

pathName	The ASPathName in the current file system.
platformPathType	The platform path type, one of the following constants: FSSpec FSRef FSRefWithCFStringRefRec CFURLRefRec
platformPath	<i>(Filled by the method)</i> The new platform path object.

Return Value

0 if the operation was successful, otherwise returns a nonzero platform-dependent error code.

Header File

[ASExpT.h](#)

Related Callbacks

[ASFileSysAcquireFileSysPathProc](#)

Related Methods

[ASFileSysAcquirePlatformPath](#)

ASFileSysAsyncAbortProc

```
ACCB1 void ACCB2 ASFileSysAsyncAbortProc (ASMDFile file);
```

Description

Callback for [ASFileSysRec](#). Aborts all uncompleted asynchronous I/O requests for the specified file. This callback can be called at any time.

This callback calls each outstanding [ASIORequest](#)'s [ASIODoneProc](#) to be called with the `totalBytes` = 0 and error = -1.

Parameters

file	The file for which all uncompleted asynchronous read/write requests are aborted.

Return Value

None

Header File

ASExpT.h

Related Callbacks

[ASFileSysAsyncReadProc](#)
[ASFileSysAsyncWriteProc](#)
[ASFileSysYieldProc](#)

ASFileSysAsyncReadProc

```
ACCB1 ASInt32 ACCB2 ASFileSysAsyncReadProc (ASIORequest req);
```

Description

Callback for [ASFileSysRec](#). Asynchronously reads the specified data, returning immediately after the request has been queued. The [ASFileSys](#) must call the [ASIODoneProc](#) (if one was provided) when the specified data has been read.

This callback is similar to the [ASFileSysMReadRequestProc](#), except that this callback contains a caller-provided [ASIODoneProc](#), and can only be used for a single byte range.

Parameters

req	Data structure specifying the data to read. Contains information about the request including the ASMDFile , the file offset, the buffer for the request, the number of bytes in the request and an ASIODoneProc and clientData for the ASIODoneProc . If the ASIODoneProc in req is non-NULL, and there is an error queueing the read request, the ASIODoneProc must <i>not</i> be called.

Return Value

0 if the request was successfully queued, otherwise returns a nonzero platform-dependent error code.

Header File

[ASExpT.h](#)

Related Callbacks

[ASFileSysAsyncAbortProc](#)
[ASFileSysAsyncWriteProc](#)
[ASFileSysYieldProc](#)

ASFileSysAsyncWriteProc

```
ACCB1 ASInt32 ACCB2 ASFileSysAsyncWriteProc (ASIORequest req);
```

Description

Callback for [ASFileSysRec](#). Asynchronously writes the specified data, returning immediately after the request has been queued. The [ASFileSys](#) must call the [ASIODoneProc](#) (if one was provided) when the specified data has been written.

Parameters

req Data structure specifying the data to write. Contains information about the request including the [ASMDFile](#), the file offset, the buffer for the request, the number of bytes in the request and an [ASIODoneProc](#) and **clientData** for the [ASIODoneProc](#).

If the [ASIODoneProc](#) in **req** is non-**NULL**, and there is an error queueing the write request, the [ASIODoneProc](#) must *not* be called.

Return Value

0 if the request was successfully queued, otherwise returns a nonzero platform-dependent error code.

Header File

`ASExpT.h`

Related Callbacks

[ASFileSysAsyncAbortProc](#)

[ASFileSysAsyncReadProc](#)

[ASFileSysYieldProc](#)

ASFileSysCanPerformOpOnItemProc

```
ACCB1 ASInt32 ACCB2 ASFileSysCanPerformOpOnItemProc  
(ASPathName pathName, const char *op);
```

Description

Callback for [ASFileSysRec](#). Tests whether a specified operation can be performed on the file—that is, whether a handler is defined for that operation in [ASFileSysPerformOpOnItemProc](#).

Parameters

<code>pathName</code>	The ASPathName of the file.
<code>op</code>	The name of the operation to test.

Return Value

0 if the operation was successful, otherwise returns a nonzero platform-dependent error code.

Header File

`ASExpT.h`

Related Callbacks

[ASFileSysPerformOpOnItemProc](#)

Related Methods

[ASFileSysCanPerformOpOnItem](#)

ASFileSysCanSetEofProc

```
ACCB1 ASBool ACCB2 ASFileSysCanSetEofProc (ASMDFile f,
ASFilePos pos);
```

Description

Callback for [ASFileSysRec](#). Determines whether **ASFileSys** can set EOF to a new offset for the specified file.

Parameters

f	The file.
pos	The new EOF offset.

Return Value

true if EOF can be set for the file.

Header File

`ASExpT.h`

Related Callbacks

[ASFileSysSetEofProc](#)

Related Methods

[ASFileGetEOF](#)

[ASFileSetEOF](#)

ASFileSysClearOutstandingMReadsProc

```
ACCB1 void ACCB2 ASFileSysClearOutstandingMReadsProc  
(ASMDFile file);
```

Description

Callback for [ASFileSysRec](#). Used to advise a file system that the previous range of bytes requested to read are not needed, so that it may drop the read requests. The file system can continue pushing the bytes if it cannot stop the reads.

Parameters

file	The file that was being read.

Return Value

None

Header File

[ASExpT.h](#)

Related Callbacks

[ASFileSysMReadRequestProc](#)

Related Methods

[ASFileRead](#)

[ASFileHasOutstandingMReads](#)



ASFileSysCloseProc

```
ACCB1 ASInt32 ACCB2 ASFileSysCloseProc (ASMDFile f);
```

Description

Callback for [ASFileSysRec](#). This callback is responsible for closing the specified file. It is called by [ASFileClose](#).

Parameters

f	The file to close.

Return Value

0 if the request was successful, otherwise returns a nonzero platform-dependent error code.

Header File

[ASExpT.h](#)

Related Callbacks

[ASFileSysOpenProc](#)

Related Methods

[ASFileClose](#)

ASFileSysCopyPathNameProc

```
ACCB1 ASPathName ACCB2 ASFileSysCopyPathNameProc  
(ASPathName pathName);
```

Description

Callback for [ASFileSysRec](#). Copies a pathname (not the underlying file). It is called by [ASFileSysCopyPath](#).

Copying a pathname does not result in any file-level operations, and is unaffected by whether there is or is not an open file for the pathname.

NOTE: The [ASPathName](#) returned should be released by the [ASFileSysReleasePath](#) method when it is no longer needed.

Parameters

pathName	The pathname to copy.

Return Value

New copy of the pathname.

Header File

`ASExpT.h`

Related Methods

[ASFileSysCopyPath](#)

ASFileSysCreateFolderProc

```
ACCB1 ASInt32 ACCB2 ASFileSysCreateFolderProc  
(ASPathName path);
```

Description

Callback for [ASFileSysRec](#). Called to create an empty folder at the specified path.

Parameters

path	The path of the folder to create.

Return Value

0 to denote success, some error code otherwise.

Header File

`ASExpT.h`

Related Callbacks

[ASFileSysRemoveFolderProc](#)

Related Methods

[ASFileSysCreateFolder](#)

ASFileSysCreatePathNameProc

```
ACCB1 ASPathName ACCB2 ASFileSysCreatePathNameProc  
(ASAtom pathSpecType, const void* pathSpec,  
const void* clientData);
```

Description

Callback for [ASFileSysRec](#). Creates an **ASPathName** based on the input type and [PDFFileSpec](#). Each **ASFileSys** implementation must publish the input types that it accepts. For example, the Macintosh **ASFileSys** may accept the type **FSSpecPtr**, and the MS-DOS **ASFileSys** may only accept types of **Cstring**.

NOTE: The [ASPathName](#) returned should be released by the [ASFileSysReleasePath](#) method when it is no longer needed.

Parameters

specType	The type of the input PDFFileSpec .
fileSpec	The file specification from which to create an ASPathName .
clientData	Reserved for future use.

Return Value

The newly created pathname.

Header File

[ASExpT.h](#)

Related Methods

[ASFileSysCreatePathName](#)

ASFileSysDestroyFolderIteratorProc

```
ACCB1 void ACCB2 ASFileSysDestroyFolderIteratorProc  
(ASFolderIterator folderIter);
```

Description

Callback for [ASFileSysRec](#). Called to release the resources associated with `folderIter`.

Parameters

folderIter	An ASFolderIterator object returned from a previous call to ASFileSysFirstFolderItem .

Return Value

None

Header File

`ASExpT.h`

Related Callbacks

[ASFileSysFirstFolderItemProc](#)
[ASFileSysNextFolderItemProc](#)

Related Methods

[ASFileSysFirstFolderItem](#)
[ASFileSysNextFolderItem](#)
[ASFileSysDestroyFolderIterator](#)

ASFileSysDiPathFromPathProc

```
ACCB1 char* ACCB2 ASFileSysDiPathFromPathProc  
(ASPathName path, ASPathName relativeToThisPath);
```

Description

Callback for [ASFileSysRec](#). Converts a pathname to a device-independent pathname. It is called by [ASFileSysDIPathFromPath](#).

NOTE: The memory for the `char*` returned should be freed with the [ASfree](#) method when it is no longer needed.

Parameters

path	The pathname to convert to a device-independent pathname.
relativeToThisPath	The path relative to which the device-independent pathname is specified. Pass NULL if the device-independent pathname (for example, <code>c:\dir1\dir2\dir3\myfile.pdf</code>) is an absolute pathname instead of a relative pathname (for example, <code>../../dir3/myfile.pdf</code>).

Return Value

The device-independent pathname.

Header File

`ASExpT.h`

Related Methods

[ASFileSysDIPathFromPath](#)

ASFileSysDIPathFromPathExProc

```
ACCB1 ASErrorCode ACCB2 ASFileSysDIPathFromPathExProc
(ASPathName path, ASPathName relativeToThisPath,
ASText diPathText);
```

Description

Callback for [ASFileSysRec](#). Converts a pathname to a device-independent pathname, returned as an [ASText](#) object. It is called by [ASFileSysDIPathFromPathEx](#).

Parameters

path	The pathname to convert to a device-independent pathname.
relativeToThisPath	The path relative to which the device-independent pathname is specified. Pass NULL if the device-independent pathname (for example, c:\dir1\dir2\dir3\myfile.pdf) is an absolute pathname instead of a relative pathname (for example, ../../dir3/myfile.pdf).
di_PathText	<i>(Filled by the method)</i> The ASText object to contain the device-independent path. Must be allocated and freed by the client.

Return Value

0 if the operation was successful, otherwise returns a nonzero platform-dependent error code.

Header File

[ASExpT.h](#)

Related Callbacks

[ASFileSysPathFromDIPathExProc](#)

Related Methods

[ASFileSysDIPathFromPathEx](#)

[ASFileSysPathFromDIPathEx](#)

ASFileSysDisplayASTextFromPathProc

```
ACCB1 ASErrorCode ACCB2 ASFileSysDisplayASTextFromPathProc  
(ASPathName path, ASText displayText);
```

Description

Callback for [ASFileSysRec](#). Called to obtain a representation of a path that can be displayed by the user.

Parameters

path	The ASPathName in question.
displayText	<i>(Filled by method)</i> The text object containing the display representation of the path.

Return Value

0 if the operation was successful, otherwise returns a nonzero platform-dependent error code.

Header File

[ASExpT.h](#)

Related Methods

[ASFileSysDisplayASTextFromPath](#)

ASFileSysDisplayStringFromPathProc

```
ACCB1 char* ACCB2 ASFileSysDisplayStringFromPathProc  
(ASPathName path);
```

Description

Callback for [ASFileSysRec](#). Called to obtain a representation of a path that can be displayed by the user.

Parameters

path	The ASPathName in question.

Return Value

The display string or **NULL** if some error occurred. Allocated memory must be free-able with [ASfree](#).

Header File

ASExpT.h

Related Methods

[ASFileSysDisplayStringFromPath](#)

ASFileSysDisposePathNameProc

```
ACCB1 void ACCB2 ASFileSysDisposePathNameProc  
(ASPathName pathName);
```

Description

Callback for [ASFileSysRec](#). It is called by [ASFileSysReleasePath](#).

This callback frees any memory occupied by a pathname. It does not result in any file-level operations.

Parameters

pathName	The pathname to release.

Return Value

None

Header File

ASExpT.h

Related Methods

[ASFileSysReleasePath](#)

ASFileSysFirstFolderItemProc

```
ACCB1 ASFolderIterator ACCB2 ASFileSysFirstFolderItemProc  
(ASPathName folderPath, ASFileSysItemProps props,  
ASPathName* itemPath);
```

Description

Callback for [ASFileSysRec](#). Begins the process of iterating through the contents of a folder.

Parameters

folderPath	Path to the folder to be iterated through.
props	<i>(Filled by the callback)</i> Properties structure describing the object.
itemPath	<i>(Filled by the callback)</i> A newly-allocated ASPathName associated with the object (which the client must free). Contains an absolute path.

Return Value

ASFolderIterator object used for iterating through subsequent items. If there are no items in the folder this procedure returns **NULL**.

Header File

ASExpT.h

Related Callbacks

[ASFileSysNextFolderItemProc](#)
[ASFileSysDestroyFolderIteratorProc](#)

Related Methods

[ASFileSysFirstFolderItem](#)
[ASFileSysNextFolderItem](#)
[ASFileSysDestroyFolderIterator](#)



ASFileSysFlushProc

```
ACCB1 ASInt32 ACCB2 ASFileSysFlushProc (ASMDFile f);
```

Description

Callback for [ASFileSysRec](#). Flushes data for the specified file. It is called by [ASFileFlush](#).

Parameters

f	The file to flush.

Return Value

0 if the request was successful, otherwise returns a nonzero platform-dependent error code.

Header File

[ASExpT.h](#)

Related Callbacks

[ASFileSysFlushVolumeProc](#)

Related Methods

[ASFileFlush](#)

ASFileSysFlushVolumeProc

```
ACCB1 ASInt32 ACCB2 ASFileSysFlushVolumeProc  
(ASPathName pathName);
```

Description

Callback for [ASFileSysRec](#). Flushes the volume on which the specified file resides. This ensures that any data written to the system for the volume containing **pathName** is flushed out to the physical volume (equivalent to the Macintosh **FlushVol**, or to the UNIX **sync**). Call this after you're finished writing a complete "transaction" to force a commit.

This callback is not called directly from any client API method, but is used internally by the Acrobat viewer.

Parameters

pathName	The pathname for the file whose volume is flushed.

Return Value

0 if the request was successful, otherwise returns a nonzero platform-dependent error code.

Header File

ASExpT.h

Related Callbacks

[ASFileSysFlushProc](#)

ASFileSysGetEofProc

```
ACCB1 ASInt32 ACCB2 ASFileSysGetEofProc (ASMDFile f,
ASInt32* pos) ;
```

Description

Callback for [ASFileSysRec](#). Gets a file’s current logical size. Called by [ASFileGetEOF](#).

Parameters

f	The file whose logical size is obtained.
pos	<i>(Filled by the callback)</i> The file’s logical size, in bytes.

Return Value

0 if the request was successful, otherwise returns a nonzero platform-dependent error code.

Header File

```
ASExpT.h
```

Related Methods

```
ASFileGetEOF
ASFileSetEOF
```



ASFileSysGetFileFlags

```
ACCB1 ASFlagBits ACCB2 ASFileSysGetFileFlags (ASMDFile file);
```

Description

Callback for [ASFileSysRec](#). Gets the flags for the specified file.

Parameters

file	The file whose flags are obtained.

Return Value

Bit field using [ASFile Flags](#).

Header File

ASExpT.h

ASFileSysGetFileSysNameProc

```
ACCB1 ASAtom ACCB2 ASFileSysGetFileSysNameProc (void);
```

Description

Callback for [ASFileSysRec](#). Gets this file system's name.

This callback is not called directly by any method in the client API, but is used internally by the Acrobat viewer.

Parameters

None

Return Value

The [ASAtom](#) containing the name of this file system.

Header File

`ASExpT.h`

Related Methods

[ASFileRegisterFileSys](#)

ASFileSysGetItemPropsProc

```
ACCB1 ASInt32 ACCB2 ASFileSysGetItemPropsProc  
(ASPathName path, ASFileSysItemProps props);
```

Description

Callback for [ASFileSysRec](#). Called to retrieve a full description of the file system object associated with **path**.

Parameters

path	The ASPathName associated with the object.
props	<i>(Filled by the callback)</i> Properties structure describing the object.

Return Value

0 to denote success, some error code otherwise.

Header File

ASExpT.h

Related Methods

[ASFileSysGetItemProps](#)

ASFileSysGetItemPropsAsCabProc

```
ACCB1 ASInt32 ACCB2 ASFileSysGetItemPropsAsCabProc
(ASPathName pathName, ASCab theCab);
```

Description

Callback for **ASFileSysRec**. Gets a full description of the file system object associated with **pathName**, returning the item properties in the **ASCab** format.

If the **ASCab** has no keys on entry, every property known is filled in. If it is not empty, only properties corresponding to keys in the **ASCab** are filled in. Keys that do not map to a property of the object are removed. The **ASCab** has the following potential entries:

```
ASBool isThere;
ASInt32 type;
ASBool isHidden;
ASBool isReadOnly;
char * creationDate; // PDF style date string
char * modDate; // PDF style date string
ASUns32 fileSizeHigh;
ASUns32 fileSizeLow;
ASInt32 folderSize;
ASUns32 creatorCode;
ASUns32 typeCode;
ASUns32 versionMajor;
ASUns32 versionMinor;
ASBool isCheckedOut;
ASBool isPublished;
```

Parameters

pathName	The ASPathName associated with the object.
theCab	<i>(Filled by the method)</i> Properties describing the object, in cabinet format.

Return Value

0 if no error was encountered; otherwise an error code is returned. If an error code is returned, **theCab** is not filled with valid values. If the path name does not point to an object on the file system, returns **asFileErrFNF** and a valid **ASCab** with **isThere** set to **false**.

Header File

ASExpT.h

Related Methods

ASFileSysGetItemPropsAsCab

ASFileSysGetNameProc

```
ACCB1 ASErrorCode ACCB2 ASFileSysGetNameProc
(ASPathName pathName, char* name, ASArraySize maxLength);
```

Description

Callback for [ASFileSysRec](#). Returns a character string containing the filename for the specified [ASPathName](#). The character string contains only the filename; it is not a complete pathname.

This callback is not called directly from any client API method. It is used internally by the Acrobat viewer.

NOTE: Superseded by [ASFileSysGetNameAsASTextProc](#) in Acrobat 6.0.

Parameters

pathName	The ASPathName for which the filename is returned.
name	<i>(Filled by the callback)</i> Character string containing the filename for pathName .
maxLength	Maximum number of characters that name can hold.

Return Value

0 if the request was successful, otherwise returns a nonzero platform-dependent error code.

Header File

ASExpT.h

Related Callbacks

[ASFileSysGetFileSysNameProc](#)
[ASFileSysGetNameAsASTextProc](#)

ASFileSysGetNameAsASTextProc

```
ACCB1 ASErrorCode ACCB2 ASFileSysGetNameProc  
(ASPathName pathName, ASText name);
```

Description

Callback for [ASFileSysRec](#). Gets the filename for the specified [ASPathName](#) as an [ASText](#) object.

NOTE: Supersedes [ASFileSysGetNameProc](#) for Acrobat 6.0.

Parameters

pathName	The ASPathName for which the filename is returned.
name	<i>(Filled by the callback)</i> ASText object for the filename for pathName .

Return Value

0 if the request was successful, otherwise returns a nonzero platform-dependent error code.

Header File

[ASExpT.h](#)

Related Callbacks

[ASFileSysGetFileSysNameProc](#)
[ASFileSysGetNameProc](#)



ASFileSysGetParentProc

```
ACCB1 ASPathName ACCB2 ASFileSysGetParentProc  
(ASPathName path);
```

Description

Callback for [ASFileSysRec](#). Called to obtain the parent of the input path.

Parameters

path	The ASPathName in question.

Return Value

The parent path, or **NULL** if path is a root directory. It is the client's responsibility to free the returned **ASPathName**.

Header File

ASExpT.h

ASFileSysGetPosProc

```
ACCB1 ASInt32 ACCB2 ASFileSysGetPosProc (ASMDFile f,
ASInt32* pos) ;
```

Description

Callback for [ASFileSysRec](#). Gets the current position for the specified file. Called by [ASFileGetPos](#).

Parameters

f	The file whose current position is obtained.
pos	<i>(Must by filled by the callback)</i> The current position.

Return Value

0 if the request was successful, otherwise returns a nonzero platform-dependent error code.

Header File

```
ASExpT.h
```

Related Callbacks

```
ASFileSysSetPosProc
```

Related Methods

```
ASFileGetPos
ASFileSetPos
```

ASFileSysGetStatusProc

```
ACCB1 ASFlagBits ACCB2 ASFileSysGetStatusProc (ASMDFile file);
```

Description

Callback for [ASFileSysRec](#). Gets the status of the specified file. This callback is used for asynchronous I/O. For example, it can indicate that an underlying file connection has been closed.

Parameters

file	The file whose status is obtained.

Return Value

The file's status. Must be one of the [ASFileStatus Flags](#) values.

Header File

`ASExpT.h`

Related Methods

[ASFileRead](#)

ASFileSysGetStorageFreeSpaceProc

```
ACCB1 ASInt32 ACCB2 ASFileSysGetStorageFreeSpaceProc  
(ASPathName pathName);
```

Description

Callback for [ASFileSysRec](#). Gets the amount of free space on the volume containing the specified **ASPathName**.

Parameters

pathName	The ASPathName for a file on the volume whose free space is obtained.

Return Value

Free space, in bytes. Because the free space is returned as an [ASInt32](#), it is limited to 4 GB.

Header File

ASExpT.h

ASFileSysGetTempPathNameProc

```
ACCB1 ASPathName ACCB2 ASFileSysGetTempPathNameProc  
(ASPathName pathName);
```

Description

Callback for [ASFileSysRec](#). Returns an unique pathname suitable for use in creating temporary files.

NOTE: The [ASPathName](#) returned should be released by the [ASFileSysReleasePath](#) method when it is no longer needed.

Parameters

pathName	If pathName is non- NULL , the temporary file must be stored such that a rename of pathName will succeed (for example, on the same volume if renames across volumes are illegal on this file system).

Return Value

Pathname for a temporary file.

Header File

`ASExpT.h`

Related Callbacks

[ASFileSysCopyPathNameProc](#)

ASFileSysGetTypeAndCreatorProc

```
ACCB1 void ACCB2 ASFileSysGetTypeAndCreatorProc
(ASPathName path, unsigned long* type,
 unsigned long* creator);
```

Description

Callback for [ASFileSysRec](#). Gets the file type and creator on the file. Currently only implemented on Mac.

Does not raise.

Parameters

path	Path to file.
type	<i>(Filled by the callback)</i> The file type.
creator	<i>(Filled by the callback)</i> The file creator.

Return Value

None

Header File

[ASExpT.h](#)

Related Callbacks

[ASFileSysSetTypeAndCreatorProc](#)

Related Methods

[ASFileSysGetTypeAndCreator](#)

[ASFileSysSetTypeAndCreator](#)

ASFileSysIsSameFileProc

```
ACCB1 ASBool ACCB2 ASFileSysIsSameFileProc (ASMDFile f,  
ASPathName pathName, ASPathName newPathName);
```

Description

Callback for **ASFileSysRec**. Tests whether two files are the same.

Parameters

f	The ASFile of first file to compare (in many implementations, it may be unnecessary to use this information, pathName and newPathName provide sufficient information).
pathName	The ASPathName of first file to compare.
newPathName	The ASPathName of second file to compare.

Return Value

0 if the files are different, nonzero if they are the same.

Header File

ASExpT.h

ASFileSysMReadRequestProc

```
ACCB1 ASInt32 ACCB2 ASFileSysMReadRequestProc (ASMDFile file,  
ASFile aFile, ASTFilePos* blockPairs,  
ASTArraySize nBlockPairs);
```

Description

Callback for [ASFileSysRec](#). Queues asynchronous requests for one or more byte ranges that the caller (usually the Acrobat viewer or Library) will need in the near future. This callback is important for slow file systems, such as the Web, to improve overall performance by allowing the file system to begin retrieving bytes before they are actually needed, while the Acrobat software continues processing as much as it can with the data that has already been downloaded.

This callback does not actually read the data, but merely queues the requests, starts the asynchronous code that reads the data, and returns. The asynchronous code that reads the data must use [ASFilePushData](#) to push the data from each byte range to the Acrobat software as soon as the data is ready.

This callback is similar to the [ASFileSysAsyncReadProc](#), except that this callback contains a caller-provided [ASIODoneProc](#), and can only be used for a single byte range.

Parameters

file	The file whose data is read.
aFile	The corresponding ASFile , for use with ASFilePushData .
blockPairs	An array of file offsets and byte lengths.
nBlockPairs	The number of block pairs in blockPairs .

Return Value

0 if the request was successfully queued, otherwise returns a nonzero platform-dependent error code.

Header File

`ASExpT.h`

Related Callbacks

[ASFileSysAsyncReadProc](#)
[ASFileSysClearOutstandingMReadsProc](#)

Related Methods

[ASFileRead](#)
[ASFileHasOutstandingMReads](#)

ASFileSysNextFolderItemProc

```
ACCB1 ASBool ACCB2 ASFileSysNextFolderItemProc  
(ASFolderIterator folderIter, ASFileSysItemProps props,  
ASPathName* itemPath);
```

Description

Callback for [ASFileSysRec](#). Called to continue the iteration process associated with the [ASFolderIterator](#) object. Both **itemPath** and **props** are optional and can be **NULL** if the caller is not interested in that information.

Parameters

folderIter	An ASFolderIterator object returned from a previous call to ASFileSysFirstFolderItemProc .
props	<i>(Filled by the callback)</i> Properties structure describing the object.
itemPath	<i>(Filled by the callback)</i> A newly-allocated ASPathName associated with the object (which the client must free). Contains an absolute path.

Return Value

false if no other objects were found, **true** otherwise.

Header File

[ASExpT.h](#)

Related Callbacks

[ASFileSysFirstFolderItemProc](#)
[ASFileSysDestroyFolderIteratorProc](#)

Related Methods

[ASFileSysFirstFolderItem](#)
[ASFileSysNextFolderItem](#)
[ASFileSysDestroyFolderIterator](#)

ASFileSysOpenProc

```
ACCB1 ASInt32 ACCB2 ASFileSysOpenProc (ASPathName pathName,  
ASUns16 mode, ASMDFile\* fP);
```

Description

Callback for [ASFileSysRec](#). Opens the specified file. It is called by [ASFileSysOpenFile](#) and [ASFileReopen](#).

Parameters

pathName	The pathname for the file to open.
mode	The mode in which the file is opened. Must be an OR of the ASFileMode .
fP	<i>(Filled by the callback)</i> The newly-opened file.

Return Value

0 if the request was successful, otherwise returns a nonzero platform-dependent error code.

Header File

ASExpT.h

Related Callbacks

[ASFileSysCloseProc](#)

Related Methods

[ASFileSysOpenFile](#)
[ASFileReopen](#)
[ASFileClose](#)

ASFileSysPathFromDIPathExProc

```
ACCB1 ASPathName ACCB2 ASFileSysPathFromDIPathExProc
(ASText diPath, ASPathName relativeToThisPath);
```

Description

Callback for [ASFileSysRec](#). Converts a device-independent pathname from an [ASText](#) object to an [ASPathName](#). It is called by [ASFileSysPathFromDIPathEx](#).

NOTE: The [ASPathName](#) returned should be released by the [ASFileSysReleasePath](#) method when it is no longer needed.

Parameters

diPath	Device-independent pathname to convert to an ASPathName , as an ASText object.
relativeToThisPath	If diPath is an absolute pathname, NULL . If diPath is a relative pathname, the pathname relative to which it is specified.

Return Value

The [ASPathName](#) corresponding to the specified device-independent pathname.

Header File

[ASExpT.h](#)

Related Callbacks

[ASFileSysDIPathFromPathExProc](#)

Related Methods

[ASFileSysPathFromDIPathEx](#)

[ASFileSysDIPathFromPathEx](#)

ASFileSysPathFromDIPathProc

```
ACCB1 ASPathName ACCB2 ASFileSysPathFromDIPathProc  
(char* diPath, ASPathName relativeToThisPath);
```

Description

Callback for [ASFileSysRec](#). Converts a device-independent pathname to an [ASPathName](#). It is called by [ASFileSysPathFromDIPath](#).

NOTE: The [ASPathName](#) returned should be released by the [ASFileSysReleasePath](#) method when it is no longer needed.

Parameters

diPath	Device-independent pathname to convert to an ASPathName .
relativeToThisPath	If diPath is an absolute pathname, NULL . If diPath is a relative pathname, the pathname relative to which it is specified.

Return Value

The [ASPathName](#) corresponding to the specified device-independent pathname.

Header File

`ASExpT.h`

Related Methods

[ASFileSysPathFromDIPath](#)
[ASFileSysDIPathFromPath](#)

ASFileSysPerformOpOnItemProc

```
ACCB1 ASInt32 ACCB2 ASFileSysPerformOpOnItemProc  
(ASPathName pathName, const char *op,, ASCab params);
```

Description

Callback for [ASFileSysRec](#). Performs the specified operation on a particular file.

Parameters

pathName	The ASPathName of the file.
op	The name of the operation to perform (a filesystem-defined string).
params	An ASCab object containing parameters to pass to the operation.

Return Value

0 if the operation was successful, otherwise returns a nonzero platform-dependent error code.

Header File

[ASExpT.h](#)

Related Callbacks

[ASFileSysCanPerformOpOnItemProc](#)

Related Methods

[ASFileSysPerformOpOnItem](#)

ASFileSysRangeArrivedProc

```
ACCB1 void ACCB2 ASFileSysRangeArrivedProc (ASInt32 start,  
ASInt32 length, void* clientData);
```

Description

Callback for [ASFileSysRec](#). Called when a byte range has arrived during a file load operation.

Parameters

start	The offset from the beginning of the file to the start of the byte range.
length	The number of bytes in the byte range.
clientData	A pointer to data to pass to the callback.

Return Value

None

Header File

ASExpT.h

ASFileSysReleasePlatformPathProc

```
ACCB1 ASInt32 ACCB2 ASFileSysReleasePlatformPathProc  
(ASPathName pathName);
```

Description

Callback for [ASFileSysRec](#). Releases the specified platform path object when the client is done with it.

Parameters

pathName	The ASPathName of the file.

Return Value

0 if the operation was successful, otherwise returns a nonzero platform-dependent error code.

Header File

`ASExpT.h`

Related Callbacks

[ASFileSysAcquirePlatformPathProc](#)

Related Methods

[ASFileSysReleasePlatformPath](#)

ASFileSysReadProc

```
ACCB1 ASSize_t ACCB2 ASFileSysReadProc (void* ptr,
ASSize_t size, ASSize_t count, ASMDFile f, ASInt32* pError);
```

Description

Callback for [ASFileSysRec](#). Reads data from the specified file. It is called by [ASFileRead](#).

Parameters

ptr	<i>(Filled by the callback)</i> The data read from the file.
size	The size of each item to read.
count	The number of items to read.
f	The file from which data is read.
pError	<i>(Filled by the callback)</i> Error code. This value is filled only if an error occurred. In that case, it should contain an error code.

Return Value

The number of bytes read, or 0 if there was an error.

Header File

ASExpT.h

Related Callbacks

[ASFileSysWriteProc](#)

Related Methods

[ASFileRead](#)

ASFileSysRemoveFolderProc

```
ACCB1 ASInt32 ACCB2 ASFileSysRemoveFolderProc  
(ASPathName path);
```

Description

Callback for [ASFileSysRec](#). Called to delete the folder at the specified path.

Parameters

path	The path of the folder to remove.

Return Value

0 to denote success, some error code otherwise

Header File

`ASExpT.h`

Related Callbacks

[ASFileSysCreateFolderProc](#)

Related Methods

[ASFileSysRemoveFolder](#)

ASFileSysRemoveProc

```
ACCB1 ASInt32 ACCB2 ASFileSysRemoveProc (ASPathName pathName);
```

Description

Callback for [ASFileSysRec](#). Deletes a file. It is called by [ASFileSysRemoveFile](#).

Parameters

pathName	The file to delete.

Return Value

0 if the request was successful, otherwise returns a nonzero platform-dependent error code.

Header File

`ASExpT.h`

Related Methods

[ASFileSysRemoveFile](#)

ASFileSysRenameProc

```
ACCB1 ASInt32 ACCB2 ASFileSysRenameProc (ASMDFile* f,  
ASPathName oldPath, ASPathName newPath);
```

Description

Callback for [ASFileSysRec](#). Renames a file. It is not called directly by any method in the client API, but is used internally by the Acrobat viewer.

Parameters

f	The file to rename.
oldPath	The file's old pathname.
newPath	The file's new pathname.

Return Value

0 if the request was successful, otherwise returns a nonzero platform-dependent error code.

Header File

`ASExpT.h`

Related Callbacks

[ASFileSysGetNameProc](#)

ASFileSysReopenProc

```
ACCB1 ASMDFile ACCB2 ASFileSysReopenProc (ASMDFile f,
ASFileMode newMode, ASErrorCode *error);
```

Description

Callback for [ASFileSysRec](#). Reopens a file in the specified mode. [ASFileReopen](#) calls this method if it is present. If this method is not present, or if it returns **NULL** and **error** is 0, [ASFileReopen](#) does a close followed by an open. If **error** is nonzero, [ASFileReopen](#) ignores the return value and fails with that error.

On success, the old file should not need to be closed. On failure, the old file should remain unchanged.

Parameters

f	The file to reopen.
newMode	The mode for the new session.
error	The error code for the operation. 0 if the operation was successful, otherwise returns a nonzero error code. The error is platform- and file-system specific.

Return Value

The newly reopened file or **NULL**.

Header File

ASExpT.h

Related Methods

[ASFileReopen](#)
[ASFileSysOpenFile](#)

ASFileSysSetEofProc

```
ACCB1 ASInt32 ACCB2 ASFileSysSetEofProc (ASMDFile f,
ASInt32 pos) ;
```

Description

Callback for [ASFileSysRec](#). Increases or decreases the logical size of a file. It is called by [ASFileSetEOF](#).

Parameters

f	The file to expand/shrink.
pos	The desired size, in bytes.

Return Value

0 if the request was successful, otherwise returns a nonzero platform-dependent error code.

Header File

```
ASExpT.h
```

Related Callbacks

```
ASFileSysCanSetEofProc
```

Related Methods

```
ASFileGetEOF
ASFileSetEOF
```


ASFileSysSetPosProc

```
ACCB1 ASInt32 ACCB2 ASFileSysSetPosProc (ASMDFile f,  
ASInt32 pos) ;
```

Description

Callback for [ASFileSysRec](#). Sets the current position in a file (that is, the point from which data will next be read). It is called by [ASFileSetPos](#).

Parameters

f	The file in which the position is set.
pos	The desired new position (specified in bytes from the beginning of the file).

Return Value

0 if the request was successful, otherwise returns a nonzero platform-dependent error code.

Header File

`ASExpT.h`

Related Methods

[ASFileGetPos](#)
[ASFileSetPos](#)

ASFileSysSetTypeAndCreatorProc

```
ACCB1 void ACCB2 ASFileSysSetTypeAndCreatorProc
(ASPathName path, unsigned long* type,
 unsigned long* creator);
```

Description

Callback for [ASFileSysRec](#). Sets the file type and creator on the file. Currently only implemented on Macintosh. Does not raise.

Parameters

path	Path to the file.
type	File type.
creator	File creator.

Return Value

None

Header File

[ASExpT.h](#)

Related Callbacks

[ASFileSysGetTypeAndCreatorProc](#)

Related Methods

[ASFileSysGetTypeAndCreator](#)
[ASFileSysSetTypeAndCreator](#)

ASFileSysURLFromPathProc

```
ACCB1 char* ACCB2 ASFileSysURLFromPathProc (ASPathName path);
```

Description

Callback for [ASFileSysRec](#). Called to obtain the URL associated with the given [ASPathName](#).

Parameters

path	The ASPathName in question.

Return Value

The URL or **NULL** if it cannot be determined. Allocated memory must be freeable with [ASfree](#).

Header File

[ASExpT.h](#)

Related Methods

[ASFileSysURLFromPath](#)

ASFileSysWriteProc

```
ACCB1 ASSize_t ACCB2 ASFileSysWriteProc (void* ptr,
ASSize_t size, ASSize_t count, ASMDFile f, ASInt32* pError);
```

Description

Callback for [ASFileSysRec](#). Writes data to the specified file.

Parameters

ptr	Buffer containing data to write.
size	The size of each item to write.
count	The number of items to write.
f	The file into which data is written.
pError	<i>(Filled by the callback)</i> Error.

Return Value

The number of bytes written. Returns 0 if there was an error.

Header File

ASExpT.h

Related Callbacks

[ASFileSysReadProc](#)

Related Methods

[ASFileWrite](#)



ASFileSysYieldProc

```
ACCB1 ASInt32 ACCB2 ASFileSysYieldProc (ASMDFile file);
```

Description

Callback for [ASFileSysRec](#). Yields on the asynchronous I/O requests for the specified file to allow other processes a chance to process events that may be required for a file read to complete. An **ASFileSys** should implement a yield mechanism to complement asynchronous read/write requests.

In Windows, this could be a normal **PeekMessage** based yield.

In UNIX, it could mean using **select** on a file descriptor.

Parameters

file	The file whose asynchronous I/O requests are yielded.

Return Value

0 if the request was successful, otherwise returns a nonzero platform-dependent error code.

Header File

[ASExpT.h](#)

Related Callbacks

[ASFileSysAsyncAbortProc](#)
[ASFileSysAsyncReadProc](#)
[ASFileSysAsyncWriteProc](#)

Related Methods

[ASFileRead](#)



ASIODoneProc

```
ACCB1 void ACCB2 ASIODoneProc (ASIORequest req);
```

Description

Callback in [ASIORequest](#). Used by the asynchronous read/write **ASFileSys** implementation. Provided by the **ASFile** implementation to the **ASFileSys**. The **ASFileSys** must call this method when an asynchronous request is completed:

- When an I/O request has some or all of its data.
- If the request is successfully queued but an error prevents it from completing.
- If the request is aborted by calling [ASFileSysAsyncAbortProc](#). In this case, the **totalBytesCompleted**=0 and **pError**=-1.

If the request fails, this method must still be called, with the error. It is not called if there is an error queueing the read or write request, however.

Parameters

req	The I/O request for which data has been read/written.

Return Value

None

Header File

ASExpT.h

Related Callbacks

[ASFileSysAsyncAbortProc](#)
[ASFileSysAsyncReadProc](#)
[ASFileSysAsyncWriteProc](#)
[ASFileSysYieldProc](#)



ASProcStmDestroyProc

```
ACCB1 void ACCB2 ASProcStmDestroyProc (void* clientData);
```

Description

Callback for use by [ASProcStmWrOpen](#).

Called at end of stream so you can do clean up and free allocated memory.

Parameters

clientData	User-supplied data that was passed in the call to ASProcStmWrOpen .

Return Value

None

Header File

ASExpT.h

Related Callbacks

[ASStmProc](#)

Related Methods

[ASProcStmWrOpen](#)

ASReportProc

```
ACCB1 void ACCB2 ASReportProc (ASReportType reportType,  
ASInt32 errorCode, ASText message, ASText replacementText,  
ASCab moreInfo, void* reportProcData);
```

Description

A report proc can be used to report errors, warnings, and other messages to the user. Normally a report proc will notify the user of an error using a dialog, but in some contexts (such as when batch processing) it may log the error or warning to a file, or ignore it.

It is the **ASReportProc**'s responsibility to destroy all objects passed to it, and it may do so at any time.

Parameters

reportType	The type of information that is reported.
errorCode	An error code defined by the system or by ASRegisterErrorString . If message is not NULL , the errorCode can be 0.
message	Specifies the text the user should read. If the message field is NULL the system will retrieve the message associated with the errorCode . If both message and errorCode are specified the message argument is used.
replacementText	If the replacementText is not NULL , the system will attempt to replace the string "%s" in the message with the replacement text. This applies whether the text is specified via the message argument or retrieved from the system using the errorCode argument.
moreInfo	Not used currently. The report proc will destroy this cabinet immediately.
reportProcData	A pointer to the data associated with the reportProc (which should be passed to you when you acquire the report proc).

Header File

ASExtraExpT.h

Related Methods

[AVAppGetReportProc](#)
[AVCommandGetReportProc](#)

ASStmProc

ACCB1 [ASInt32](#) ACCB2 ASStmProc (char* data, [ASInt32](#) nData, void* clientData);

Description

Callback for use by [ASProcStmRdOpenEx](#) and [ASProcStmWrOpen](#). This procedure must return the number of bytes specified by **nData**, obtaining them in any way it wishes. If your procedure reads data from a file, it is generally quite inefficient to open the file, read the bytes, then close the file each time bytes are requested. Instead, consider opening the file the first time bytes are requested from it, reading the entire file into a secondary buffer, and closing the file. When subsequent requests for data from the file are received, simply copy data from the secondary buffer, rather than re-opening the file.

Parameters

data	<i>(Filled by the callback)</i> Buffer into which your procedure must place the number of bytes specified by nData .
nData	Number of bytes to read from the stream and place into data .
clientData	User-supplied data that was passed in the call to ASProcStmRdOpenEx or ASProcStmWrOpen .

Return Value

The number of bytes actually read or written.

Header File

ASExpT.h

Related Callbacks

[ASProcStmDestroyProc](#)

Related Methods

[ASProcStmRdOpenEx](#)
[ASProcStmWrOpen](#)

AVActionCopyProc

ACCB1 [PDAction](#) ACCB2 AVActionCopyProc (void* actionHandlerObj, [AVDoc](#) fromDoc, [PDAction](#) anAction, [AVDoc](#) toDoc);

Description

(Optional) Callback for [AVActionHandlerProcs](#). Called upon to copy an action, possibly to another document. Action handlers should provide this callback to allow for copying their actions. Called by [AVDocCopyAction](#).

Parameters

actionHandlerObj	User-supplied data that was passed when the action handler was registered using AVAppRegisterActionHandler .
fromDoc	The document whose action is copied.
anAction	The action to copy.
toDoc	The document to which the action is copied.

Return Value

The newly created [PDAction](#) copy.

Header File

AVExpT.h

Related Methods

[AVDocCopyAction](#)
[AVActionHandlerGetProcs](#)

AVActionDoPropertiesExProc

```
ACCB1 ASBool ACCB2 AVActionDoPropertiesExProc  
(void* actionHandlerObj, AVDoc doc, PDAction *actions,  
ASArraySize numActions );
```

Description

Callback for [AVActionHandlerProcs](#). If defined, it is called instead of **DoProperties**.

Called to allow the user to view or edit the properties for an array of actions. Typically this presents a modal OK/Cancel dialog to the user. The dialog should graphically denote properties that do not have the same value for every action in the array by using an "indeterminant" state. For example, if the action has a flag property that is set to **true** for one action and **false** for another, instead of "true" or "false", the dialog could show the string "<varies>" for that property. The user will still be able to change the "<varies>" string to either true or false, which would apply the new value to every action in the array.

Parameters

actionHandlerObj	User-supplied data that was passed when the action handler was registered using AVAppRegisterActionHandler .
doc	The document in which the actions are located.
actions	An array of actions whose properties are set.
numActions	The number of actions in the array.

Return Value

true if the actions were changed, **false** if unchanged.

Header File

`AVExpT.h`

Related Methods

[AVActionHandlerGetProcs](#)

AVActionDoPropertiesProc

```
ACCB1 void ACCB2 AVActionDoPropertiesProc  
(void* actionHandlerObj, PDAction action, AVDoc doc);
```

Description

Callback for [AVActionHandlerProcs](#). Displays a user interface that allows a user to set the action's properties (for example, for a **Launch** action, set the file to launch; for a **GoTo** action, select the destination page/zoom/coordinates).

Parameters

actionHandlerObj	User-supplied data that was passed when the action handler was registered using AVAppRegisterActionHandler .
action	The action whose properties are set.
doc	The document in which the action is located.

Return Value

None

Header File

[AVExpT.h](#)

Related Methods

[AVActionHandlerGetProcs](#)



AVActionEnumProc

```
ACCB1 ASBool ACCB2 AVActionEnumProc (ASAtom type,
char* userName, void* clientData);
```

Description

Callback used by [AVAppEnumActionHandlers](#). It is called once for each action handler.

Parameters

type	The action handler's name.
userName	The action's name, as it appears in the Acrobat viewer's user interface.
clientData	User-supplied data that was passed in the call to AVAppEnumActionHandlers .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

AVExpT.h

Related Methods

[AVActionHandlerGetProcs](#)
[AVAppEnumActionHandlers](#)

AVActionFillActionDictProc

```
ACCB1 void ACCB2 AVActionFillActionDictProc  
(void* actionHandlerObj, CosObj actionDict, AVDoc doc);
```

Description

(Required) Callback for [AVActionHandlerProcs](#). This required function is called as soon as the user selects the action from the action types pop-up menu. It allows an action handler to populate a newly created action's **actionDict**. At the time this method is called, the information needed to completely specify an action is often not yet available. As a result, this method is generally a good place to populate the **actionDict** with default values.

Parameters

actionHandlerObj	User-supplied data that was passed when the action handler was registered using AVAppRegisterActionHandler .
actionDict	Action dictionary to populate with default values.
doc	The document in which the action is located.

Return Value

None

Header File

[AVExpT.h](#)

Related Methods

[AVActionHandlerGetProcs](#)

AVActionGetButtonTextProc

```
ACCB1 ASInt32 ACCB2 AVActionGetButtonTextProc
(void* actionHandlerObj, PDAction action, char* buffer,
ASInt32 bufLen, AVDoc doc);
```

Description

Callback for [AVActionHandlerProcs](#). This optional function should store into buffer a null-terminated C string which is a localized string for the “edit action” button in the action dialog.

For example, the Acrobat viewer’s built-in “OpenFile” action returns “Select File” for this string.

Parameters

actionHandlerObj	User-supplied data that was passed when the action handler was registered using AVAppRegisterActionHandler .
action	The action whose button text is returned.
buffer	<i>(Filled by the callback)</i> The button text.
bufLen	Length of buffer , in bytes.
doc	The document in which the action is located.

Return Value

The number of characters copied into **buffer**.

Header File

AVExpT.h

Related Callbacks

[AVActionGetStringOneTextProc](#)
[AVActionGetStringTwoTextProc](#)

Related Methods

[AVActionHandlerGetProcs](#)

AVActionGetDetailsProc

```
ACCB1 void ACCB2 AVActionGetDetailsProc
(void* actionHandlerObj, AVDoc doc, PDAction action,
ASCab details);
```

Description

Optional callback for [AVActionHandlerProcs](#). Called to get informational details to present to the user. The handler can provide one or more **ASText** strings which are added to the **ASCab** provided by the caller. Generally, each string should form a single informational item, often provided in a "key: value" format. The caller will organize the strings into a list that is presented in the user interface.

Parameters

actionHandlerObj	User-supplied data that was passed when the action handler was registered using AVAppRegisterActionHandler .
doc	The document in which the action is located.
action	The action whose details are returned.
details	<i>(Filled by the callback)</i> The ASCab to populate with the informational details, provided as ASText strings.

Return Value

None

Header File

AVExpT.h

Related Methods

[AVActionHandlerGetProcs](#)

AVActionGetInstructionsProc

```
ACCB1 ASInt32 ACCB2 AVActionGetInstructionsProc  
(void* actionHandlerObj, PDAction action, char* buffer,  
ASInt32 bufLen, AVDoc doc);
```

Description

Callback for [AVActionHandlerProcs](#). This optional function should store into **buffer** a null-terminated C string which contains localized instructions for the action creation/properties dialog.

Parameters

actionHandlerObj	User-supplied data that was passed when the action handler was registered using AVAppRegisterActionHandler .
action	The action whose instructions are returned.
buffer	<i>(Filled by the callback)</i> The instruction text.
bufLen	Length of buffer , in bytes.
doc	The document in which the action is located.

Return Value

The number of characters copied into **buffer**.

Header File

AVExpT.h

Related Callbacks

[AVActionPerformExProc](#)

[AVActionPerformProc](#)

Related Methods

[AVActionHandlerGetProcs](#)

AVActionGetStringOneTextProc

```
ACCB1 ASInt32 ACCB2 AVActionGetStringOneTextProc  
(void* actionHandlerObj, PDAction action, char* buffer,  
ASInt32 bufLen, AVDoc doc);
```

Description

(Optional) Callback for [AVActionHandlerProcs](#). This function should store into buffer a null-terminated C string which is a localized string placed above the “edit action” button in the action dialog. A **NULL** proc will cause the button to hide.

For example, the Acrobat viewer’s built-in “OpenFile” action returns “File N: <the current filename>” for this string.

Parameters

actionHandlerObj	User-supplied data that was passed when the action handler was registered using AVAppRegisterActionHandler .
action	The action whose “string 1” text is returned.
buffer	(Filled by the callback) The string text appearing above the button.
bufLen	Length of buffer , in bytes.
doc	The document in which the action is located.

Return Value

The number of characters copied into **buffer**.

Header File

AVExpT.h

Related Callbacks

[AVActionGetButtonTextProc](#)
[AVActionGetStringTwoTextProc](#)

Related Methods

[AVActionHandlerGetProcs](#)

AVActionGetStringTwoTextProc

```
ACCB1 ASInt32 ACCB2 AVActionGetStringTwoTextProc
(void* actionHandlerObj, PDAction action, char* buffer,
ASInt32 bufLen, AVDoc doc);
```

Description

Callback for [AVActionHandlerProcs](#). This optional function should store into buffer a null-terminated C string which is a localized string placed below the “edit action” button in the action dialog.

For example, the Acrobat viewer’s built-in “OpenFile” action returns nothing for this string, but the built-in “GoToView” action returns a description of the current zoom type.

Parameters

actionHandlerObj	User-supplied data that was passed when the action handler was registered using AVAppRegisterActionHandler .
action	The action whose “string 2” text is returned.
buffer	<i>(Filled by the callback)</i> The string text appearing below the button.
bufLen	Length of buffer , in bytes.
doc	The document in which the action is located.

Return Value

The number of characters copied into **buffer**.

Header File

AVExpT.h

Related Callbacks

[AVActionGetButtonTextProc](#)
[AVActionGetStringOneTextProc](#)

Related Methods

[AVActionHandlerGetProcs](#)

AVActionPerformExProc

```
ACCB1 void ACCB2 AVActionPerformExProc
(void *actionHandlerObj, PDAction action, AVDoc doc,
AVActionContext context);
```

Description

Callback for [AVActionHandlerProcs](#). If defined, it is called instead of [AVActionPerformProc](#). It gets passed the context of an action, which provides information on who triggered the action.

Parameters

actionHandlerObj	User-supplied data that was passed when the action handler was registered using AVAppRegisterActionHandler .
action	An action to perform.
doc	The document in which the action is located.
context	The context of the action.

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVActionPerformProc](#)

Related Methods

[AVActionHandlerGetProcs](#)

AVActionPerformProc

```
ACCB1 void ACCB2 AVActionPerformProc (void* actionHandlerObj,
    PDAction action, AVDoc doc);
```

Description

Callback for [AVActionHandlerProcs](#). Performs the action.

Parameters

actionHandlerObj	User-supplied data that was passed when the action handler was registered using AVAppRegisterActionHandler .
action	The action to perform.
doc	The document in which the action is located.

Return Value

None

Header File

AVExpT.h

Related Callbacks

- [AVActionGetInstructionsProc](#)
- [AVActionPerformExProc](#)
- [AVActionDoPropertiesExProc](#)

Related Methods

- [AVActionHandlerGetProcs](#)

AVAnnotHandlerAdjustCursorExProc

```
ACCB1 ASBool ACCB2 AVAnnotHandlerAdjustCursorExProc  
(AVAnnotHandler annotHandler, PDAnnot anAnnot,  
AVPageView pageView, AVAdjustCursorParams params);
```

Description

(Optional) Callback for [AVAnnotHandler](#). It controls the cursor shape when the cursor is within the annotation. If **NULL**, the annotation behaves as if the **AdjustCursor** callback returned **false**.

NOTE: Introduced in Acrobat 6.0. Supersedes [AVAnnotHandlerAdjustCursorProc](#).

Parameters

annotHandler	The annotation handler responsible for this annotation.
anAnnot	The annotation containing the cursor.
pageView	The AVPageView in which the annotation is located.
params	The parameters structure containing the cursor's current x- and y-coordinates.

Return Value

true if the callback handled the adjust cursor event, **false** otherwise. The callback would return **false**, for example, if the annotation is irregularly shaped and the cursor is not currently over the real annotation even though it is within the rectangular bounding box that the Acrobat viewer uses to specify annotations.

Header File

`AVExpT.h`

Related Callbacks

[AVAnnotHandlerAdjustCursorProc](#)

AVAnnotHandlerAdjustCursorProc

```
ACCB1 ASBool ACCB2 AVAnnotHandlerAdjustCursorProc
(AVAnnotHandler annotHandler, PDAnnot anAnnot,
AVPageView pageView, AVDevCoord xHit, AVDevCoord yHit);
```

Description

NOTE: Deprecated in Acrobat 6.0. Use [AVAnnotHandlerAdjustCursorExProc](#).

(Optional) Callback for [AVAnnotHandler](#). It controls the cursor shape when the cursor is within the annotation. If **NULL**, the annotation behaves as if the **AdjustCursor** callback returned **false**.

Parameters

annotHandler	The annotation handler responsible for this annotation.
anAnnot	The annotation containing the cursor.
pageView	The AVPageView in which the annotation is located.
xHit	The cursor's current x-coordinate.
yHit	The cursor's current y-coordinate.

Return Value

true if the callback handled the adjust cursor event, **false** otherwise. The callback would return **false**, for example, if the annotation is irregularly shaped and the cursor is not currently over the real annotation even though it is within the rectangular bounding box that the Acrobat viewer uses to specify annotations.

Header File

AVExpT.h

Related Callbacks

[AVAnnotHandlerAdjustCursorExProc](#)

AVAnnotHandlerAppearanceDrawingProc

```
ACCB1 void (ACCB2 *AVAnnotHandlerAppearanceDrawingProc)
(AVAnnotHandler annotHandler, PDAnnot anAnnot,
AVPageView pageView, const AVDevRect* annotRect,
const AVDevRect* updateRect);
```

Description

Callback for [AVAnnotHandler](#) that draws the appearance of the annotation for the given page view. Use this prototype for three callbacks, **BeginAppearanceDrawing**, **FinishAppearanceDrawing**, and **CancelAppearanceDrawing**.

If the viewer has to abandon the drawing of the appearance, it calls the **CancelAppearanceDrawing** callback. If the viewer succeeds in drawing the appearance, it calls the **FinishAppearanceDrawing** callback. In either case, the handler can destroy the appearance object. See [AVAnnotHandler](#).

Parameters

annotHandler	The annotation handler responsible for the annotation.
anAnnot	The annotation whose appearance is drawn.
pageView	The page view containing the annotation.
annotRect	A pointer to a rectangle in the page view's device space that contains the currently drawn appearance. The procedure must not modify this object.
updateRect	A pointer to a rectangle in the page view's device space that contains the newly drawn appearance. The procedure must not modify this object. As an optimization, the handler can avoid drawing anything outside this rectangle, which the viewer will clip.

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVAnnotHandlerGetAppearanceProc](#)

AVAnnotHandlerCopyProc

```
ACCB1 PDAnnot ACCB2 AVAnnotHandlerCopyProc  
(AVAnnotHandler annotHandler, AVDoc fromDoc, PDAnnot anAnnot,  
AVDoc toDoc);
```

Description

(Optional) Callback for [AVAnnotHandler](#). Called upon to copy an annotation, possibly to another document. Annotation handlers should provide this callback to allow for copying their annotations. Called by [AVDocCopyAnnot](#).

Parameters

annotHandler	The annotation handler responsible for this annotation.
fromDoc	The document whose annotation is copied.
anAnnot	The annotation to copy.
toDoc	The document to which the annotation is copied.

Return Value

The newly created [PDAnnot](#) copy.

Header File

`AVExpT.h`

Related Methods

[AVDocCopyAnnot](#)

AVAnnotHandlerCanPerformOpProc

```
ACCB1 ASBool ACCB2 AVAnnotHandlerCanPerformOpProc  
(AVAnnotHandler annotHandler, PDAnnot annot,  
AVPageView pageView, AVAnnotOp annotOp, AVAnnotOpData opData);
```

Description

Called to determine if this annotation can perform the specified operation.

Parameters

annotHandler	The annotation handler responsible for this annotation.
annot	The annotation.
pageView	The AVPageView in which the annotation is located.
annotOp	The operation to be performed.
opData	The data associated with the operation.

Return Value

true if the operation can be performed; **false** otherwise.

Header File

[AVExpT.h](#)

Related Callbacks

[AVAnnotHandlerPerformOpProc](#)

AVAnnotHandlerCursorEnterProc

```
ACCB1 void ACCB2 AVAnnotHandlerCursorEnterProc  
(AVAnnotHandler annotHandler, PDAnnot anAnnot,  
AVPageView pageView);
```

Description

(Optional) Callback for [AVAnnotHandler](#). Called whenever the cursor moves over an annotation handled by this annotation handler.

Parameters

annotHandler	The annotation handler responsible for this annotation.
anAnnot	The annotation.
pageView	The AVPageView in which the annotation is located.

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVAnnotHandlerCursorExitProc](#)

AVAnnotHandlerCursorExitProc

```
ACCB1 void ACCB2 AVAnnotHandlerCursorExitProc
(AVAnnotHandler annotHandler, PDAnnot anAnnot,
AVPageView pageView);
```

Description

(Optional) Callback for [AVAnnotHandler](#). Called whenever the cursor moves off an annotation handled by this annotation handler.

Parameters

annotHandler	The annotation handler responsible for this annotation.
anAnnot	The annotation the cursor is exiting from.
pageView	The AVPageView in which the annotation is located.

Return Value

None

Header File

`AVExpT.h`

Related Callbacks

[AVAnnotHandlerCursorEnterProc](#)

AVAnnotHandlerDeleteInfoProc

```
ACCB1 void ACCB2 AVAnnotHandlerDeleteInfoProc
(AVAnnotHandler avanh, AVAnnotHandlerInfo info);
```

Description

(Optional) Callback for [AVAnnotHandler](#). Deletes information associated with an annotation.

To effectively use a [PDAnnotHandler](#) associated with an annotation, its [AVAnnotHandler](#) must have its [AVAnnotHandlerGetInfoProc](#) and [AVAnnotHandlerDeleteInfoProc](#) callbacks defined.

Parameters

avanoh	The annotation handler responsible for this annotation.
info	Information associated with the annotation.

Return Value

None

Header File

AVExpT.h

Related Callbacks

- [AVAnnotHandlerGetInfoProc](#)
- [PDAnnotHandlerDeleteAnnotInfoProc](#)

AVAnnotHandlerDoClickExProc

```
ACCB1 ASBool ACCB2 AVAnnotHandlerDoClickExProc  
(AVAnnotHandler annotHandler, PDAnnot hitAnnot,  
AVPageView pageView, AVClickParams clickParams);
```

Description

(Optional) Callback for [AVAnnotHandler](#). It handles both left and right mouse button clicks within the annotation. If **NULL**, the annotation behaves as if the callback returned **false**.

NOTE: Introduced in Acrobat 6.0. Supersedes [AVAnnotHandlerDoClickProc](#).

Parameters

annotHandler	The annotation handler responsible for this annotation.
hitAnnot	The annotation in which the mouse was clicked.
pageView	The AVPageView in which the annotation is located.
clickParams	The parameters for the click position and properties.

Return Value

true if the callback handled the mouse click, **false** if it did not. If the callback does not handle the click, it is passed to any annotation at the same location in lower layers.

Header File

AVExpT.h

Related Callbacks

[AVAnnotHandlerDoClickProc](#)

AVAnnotHandlerDoClickProc

```
ACCB1 ASBool ACCB2 AVAnnotHandlerDoClickProc
(AVAnnotHandler annotHandler, PDAnnot hitAnnot,
AVPageView pageView, AVDevCoord xHit, AVDevCoord yHit,
AVFlagBits16 flags, AVTCount clickNo);
```

Description

NOTE: Deprecated in Acrobat 6.0. Use [AVAnnotHandlerDoClickExProc](#).

(Optional) Callback for [AVAnnotHandler](#). It handles both left and right mouse button clicks within the annotation. If **NULL**, the annotation behaves as if the callback returned **false**.

Parameters

annotHandler	The annotation handler responsible for this annotation.
hitAnnot	The annotation in which the mouse was clicked.
pageView	The AVPageView in which the annotation is located.
xHit	The x-coordinate of the mouse click.
yHit	The y-coordinate of the mouse click.
flags	Indicates which modifier keys are pressed. Must be an OR of the Modifier Keys values.
clickNo	1 if this is a single click, 2 if a double click, 3 if triple click.

Return Value

true if the callback handled the mouse click, **false** if it did not. If the callback does not handle the click, it is passed to any annotation at the same location in lower layers.

Header File

AVExpT.h

Related Callbacks

[AVAnnotHandlerDoKeyDownExProc](#)

AVAnnotHandlerDoKeyDownExProc

```
ACCB1 ASBool ACCB2 AVAnnotHandlerDoKeyDownExProc  
(AVAnnotHandler annotHandler, PDAnnot annot,  
AVPageView pageView, AVKeyCode key, AVFlagBits16 flags);
```

Description

Called for each keystroke received when an annotation has focus.

NOTE: The **key** and **flags** numeric types have changed in Acrobat 6.0. Supersedes [AVAnnotHandlerDoKeyDownProc](#).

Parameters

annotHandler	The annotation handler responsible for this annotation.
annot	The annotation.
pageView	The AVPageView in which the annotation is located.
key	The operation to be performed.
flags	Indicates which modifier keys are pressed. See Modifier Keys .

Return Value

true if the callback handled the keystroke, **false** otherwise.

Header File

`AVExpT.h`

Related Callbacks

[AVAnnotHandlerDoKeyDownProc](#)

AVAnnotHandlerDoKeyDownProc

```
ACCB1 ASBool ACCB2 AVAnnotHandlerDoKeyDownProc  
(AVAnnotHandler annotHandler, PDAnnot anAnnot, AVKeyCode key,  
AVFlagBits16 flags);
```

Description

NOTE: Deprecated in Acrobat 6.0. Use [AVAnnotHandlerDoKeyDownExProc](#).

(Optional) Callback for [AVAnnotHandler](#). It is called to handle key presses in the annotation. If **NULL**, it is as if the **callback** returned **false**.

Called when there is a key-down event and the annotation is selected and the active tool does not want the event.

Parameters

annotHandler	The handler responsible for this annotation.
anAnnot	The annotation in which the key press occurred.
key	The key that was pressed
flags	Indicates which modifier keys are pressed. Must be an OR of the Modifier Keys values.

Return Value

true if the callback handled the key press, **false** otherwise.

Header File

AVExpT.h

Related Callbacks

[AVAnnotHandlerDoKeyDownExProc](#)

AVAnnotHandlerDoPropertiesExProc

```
ACCB1 void ACCB2 AVAnnotHandlerDoPropertiesExProc  
(AVAnnotHandler annotHandler, PDAannot anAnnot,  
AVPageView pageView);
```

Description

(Optional) Callback for [AVAnnotHandler](#). It displays whatever user interface it wishes to allow a user to change an annotation's properties. Set it to **NULL** if the annotation type has no user-specified properties.

Called when the user selects the "Properties..." item from the "Edit" menu while an annotation of this type is selected. If **NULL**, the "Properties" menu item is dimmed when a corresponding object is selected.

NOTE: Supersedes [AVAnnotHandlerDoPropertiesProc](#) in Acrobat 6.0.

Parameters

annotHandler	The annotation handler responsible for the annotation.
anAnnot	The annotation whose properties are set.
pageView	The page view containing the annotation.

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVAnnotHandlerDoPropertiesProc](#)

AVAnnotHandlerDoPropertiesProc

```
ACCB1 void ACCB2 AVAnnotHandlerDoPropertiesProc  
(AVAnnotHandler annotHandler, PDAannot anAnnot, AVDoc doc);
```

Description

NOTE: Deprecated in Acrobat 6.0. Use [AVAnnotHandlerDoPropertiesExProc](#).

(Optional) Callback for [AVAnnotHandler](#). It displays whatever user interface it wishes to allow a user to change an annotation's properties. Set it to **NULL** if the annotation type has no user-specified properties.

Called when the user selects the "Properties..." item from the "Edit" menu while an annotation of this type is selected. If **NULL**, the "Properties" menu item is dimmed when a corresponding object is selected.

Parameters

annotHandler	The annotation handler responsible for the annotation.
anAnnot	The annotation whose properties are set.
doc	The document containing the annotation.

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVAnnotHandlerDoPropertiesExProc](#)

AVAnnotHandlerDrawExProc

```
ACCB1 ASBool ACCB2 AVAnnotHandlerDrawExProc  
(AVAnnotHandler annotHandler, PDAnnot annot,  
AVPageView pageView, AVDevRect* updateRect);
```

Description

Called to request that the annotation appearance be drawn. If the **DrawEx** callback is provided, it is used in preference to the **Draw** callback. If the **GetAppearance** callback is provided and returns **true**, the Acrobat 6.0 appearance-drawing callbacks are used in preference to either of the older-style callbacks. See [AVAnnotHandler](#).

If the annotation has an appearance (**AP**) entry, use this information to draw the annotation. Read the annotation's appearance state (**AS**) entry to determine which appearance to use. Then read the Cos stream for the appropriate appearance and display it with [AVPageViewDrawCosObj](#). See the example for [AVAnnotHandlerDrawProc](#).

NOTE: Supersedes [AVAnnotHandlerDrawProc](#) in Acrobat 6.0.

Parameters

annotHandler	The annotation handler responsible for this annotation.
annot	The annotation.
pageView	The AVPageView in which the annotation is located.
updateRect	The portion of the annotation's bounding rect that should be drawn.

Return Value

true if the callback completed drawing successfully, **false** otherwise.

Header File

[AVExpT.h](#)

Related Callbacks

[AVAnnotHandlerDrawProc](#)
[AVAnnotHandlerGetAppearanceProc](#)

AVAnnotHandlerDrawProc

```
ACCB1 void ACCB2 AVAnnotHandlerDrawProc
(AVAnnotHandler annotHandler, PDAnnot anAnnot,
AVPageView pageView);
```

Description

NOTE: Deprecated in Acrobat 6.0. Use [AVAnnotHandlerDrawExProc](#).

(Optional) Callback for [AVAnnotHandler](#) that draws the annotation. Set it to **NULL** if the annotation handler has no **Draw** method.

If the annotation has an appearance (**AP**) entry, use this information to draw the annotation. Read the annotation's appearance state (**AS**) entry to determine which appearance to use. Then read the Cos stream for the appropriate appearance and display it with [AVPageViewDrawCosObj](#). See the code example below.

Parameters

annotHandler	The annotation handler responsible for the annotation.
anAnnot	The annotation to draw.
pageView	The AVPageView containing the annotation.

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVAnnotHandlerDrawExProc](#)

Example

```
ASAtom coState;
CosObj coApp;
AVRect avr;

/* PDAnnotGetState is a client defined function that returns an ASAtom
for the appearance that the /AS key points to */
coState = PDAnnotGetState(annot);
/* PDAnnotGetAppearance is a client defined function that returns the
CosStream for that appearance */
coApp = PDAnnotGetAppearance(annot, FaceNormal_K, coState);
AVPageViewGetAnnotRect(pageView, annot, &avr);
AVPageViewDrawCosObj(pageView, coApp, &avr);
```

AVAnnotHandlerEnumProc

```
ACCB1 ASBool ACCB2 AVAnnotHandlerEnumProc  
(AVAnnotHandler annotHandler, void* clientData);
```

Description

Callback for [AVAppEnumAnnotHandlers](#). It is called once for each annotation handler currently registered with the Acrobat viewer (see [AVAppRegisterAnnotHandler](#)).

Parameters

annotHandler	The annotation handler.
clientData	User-supplied data that was passed in the call to AVAppEnumAnnotHandlers .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

AVExpT.h

Related Methods

[AVAppEnumAnnotHandlers](#)

AVAnnotHandlerGetAnnotViewBBoxProc

```
ACCB1 void ACCB2 AVAnnotHandlerGetAnnotViewBBoxProc  
(AVAnnotHandler annotHandler, AVPageView pageView,  
PDAnnot anAnnot, AVDevRect* bbox);
```

Description

Callback for [AVAnnotHandler](#). It returns the rectangle enclosing the annotation on the screen.

Parameters

annotHandler	The annotation handler responsible for the annotation.
pageView	The AVPageView in which the annotation is located.
anAnnot	The annotation whose bounding box is returned.
bbox	<i>(Filled by the callback)</i> The annotation's bounding rectangle.

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVAnnotHandlerPtInAnnotViewBBoxProc](#)

AVAnnotHandlerGetAppearanceProc

```
ACCB1 ASBool (ACCB2 *AVAnnotHandlerGetAppearanceProc)
(AVAnnotHandler annotHandler, PDAnnot anAnnot, ASInt32 *flags,
AVPageView pageView, CosObj *apprObj);
```

Description

Callback for [AVAnnotHandler](#). Fills in a **CosObj** representing a Form field. Upon return, the annotation appearance is drawn according to the returned flag values

Appearance drawing is new in Acrobat 6.0. If the handler implements this method and it returns **true**, the new drawing style is used. If this function returns **false**, the viewer calls the drawing procedure to draw the annotation appearance. See [AVAnnotHandler](#).

The handler must not destroy or modify the appearance object while it is being drawn. It can do so only after the **EndAppearanceDrawing** or **CancelAppearanceDrawing** callback is exercised.

Parameters

annotHandler	The annotation handler responsible for the annotation.
anAnnot	The annotation whose appearance is drawn.
flags	A pointer to an OR of the AVAnnot Appearance Flags . On entry, Acrobat modifies the flags to reflect the annotation's current NoZoom and NoRotate bits. The method can modify these to change the viewer's drawing behavior for this annotation. The modifications take effect when the appearance is drawn—for example, if the callback clears the NoZoom bit, the appearance is drawn as if the original annotation did not have that bit set.
pageView	The page view in which the annotation is drawn.
apprObj	<i>(Filled by the method)</i> A CosObj representing a Form field.

Return Value

true to tell the viewer to position and draw the annotation appearance, **false** to call the **DrawEx** or **Draw** procedure.

Header File

AVExpT.h

Related Callbacks

[AVAnnotHandlerAppearanceDrawingProc](#)

AVAnnotHandlerGetFlagsProc

```
ACCB1 AVFlagBits32 (ACCBPROTO2  
*AVAnnotHandlerGetFlagsProc) (AVAnnotHandler annotHandler,  
PDAnnot anAnnot);
```

Description

Callback for [AVAnnotHandler](#). It returns the flags value for the annotation on the screen.

Parameters

annotHandler	The annotation handler responsible for the annotation.
anAnnot	The annotation whose flags value is returned.

Return Value

The flags value.

Header File

`AVExpT.h`

Related Callbacks

[AVAnnotHandlerGetAppearanceProc](#)

AVAnnotHandlerGetInfoProc

```
ACCB1 AVAnnotHandlerInfo ACCB2 AVAnnotHandlerGetInfoProc  
(AVAnnotHandler avanh) ;
```

Description

(Optional) Callback for [AVAnnotHandler](#). Gets information associated with an annotation.

To effectively use a **PDAnnotHandler** associated with an annotation, its **AVAnnotHandler** must have its **AVAnnotHandlerGetInfoProc** and [AVAnnotHandlerDeleteInfoProc](#) callbacks defined.

Parameters

avanoh	The annotation handler responsible for this annotation.

Return Value

Information associated with the annotation.

Header File

`AVExpT.h`

Related Callbacks

[AVAnnotHandlerDeleteInfoProc](#)

Related Methods

[AVAppRegisterAnnotHandler](#)

AVAnnotHandlerGetLayerExProc

```
ACCB1 ASFixed (ACCB2 AVAnnotHandlerGetLayerExProc)  
(AVAnnotHandler annotHandler, PDAnnot anAnnot  
AVPageView pageView);
```

Description

Callback for [AVAnnotHandler](#). It returns the annotation's layer. The layer need not be a constant. For example, the Acrobat viewer's built-in text annotations have a different layer depending on whether they are opened or closed. This ensures that a closed text annotation never appears on top of an open text annotation.

NOTE: Introduced in Acrobat 6.0. Supersedes [AVAnnotHandlerGetLayerProc](#).

Parameters

annotHandler	The annotation handler responsible for the annotation.
anAnnot	The annotation whose layer is returned.
pageView	The page view containing the annotation.

Return Value

The annotation's layer.

Header File

AVExpT.h

Related Callbacks

[AVAnnotHandlerGetLayerProc](#)

AVAnnotHandlerGetLayerProc

ACCB1 [ASFixed](#) ACCB2 AVAnnotHandlerGetLayerProc
([AVAnnotHandler](#) annotHandler, [PDAnnot](#) anAnnot);

Description

NOTE: Deprecated in Acrobat 6.0. Use [AVAnnotHandlerGetLayerExProc](#).

Callback for [AVAnnotHandler](#). It returns the annotation's layer. The layer need not be a constant. For example, the Acrobat viewer's built-in text annotations have a different layer depending on whether they are opened or closed. This ensures that a closed text annotation never appears on top of an open text annotation.

Parameters

annotHandler	The annotation handler responsible for the annotation.
anAnnot	The annotation whose layer is returned.

Return Value

The annotation's layer.

Header File

AVExpT.h

Related Callbacks

[AVAnnotHandlerGetLayerExProc](#)

AVAnnotHandlerGetTypeProc

```
ACCB1 ASAtom ACCB2 AVAnnotHandlerGetTypeProc  
(AVAnnotHandler annotHandler) ;
```

Description

(Required) Callback for [AVAnnotHandler](#). It returns an **ASAtom** indicating the annotation type for which the handler is responsible. This corresponds to the annotation's **Subtype** key in the PDF file.

This is the method that [AVAppGetAnnotHandlerByName](#) uses to find the correct handler.

Parameters

annotHandler	The annotation handler whose type is returned.

Return Value

The annotation type for which this handler is responsible.

Header File

`AVExpT.h`

Related Methods

[AVAppRegisterAnnotHandler](#)

AVAnnotHandlerNewProc

```
ACCB1 ASBool ACCB2 AVAnnotHandlerNewProc  
(AVAnnotHandler annotHandler, PDAnnot anAnnot,  
AVPageView pageView);
```

Description

(Unused) Callback for [AVAnnotHandler](#). Allows the annotation handler to add any attributes to the annotation object to make sure the new annotation is in a valid initial state for its subclass.

Parameters

annotHandler	The annotation handler.
anAnnot	Annotation to modify.
pageView	The AVPageView in which the annotation is located.

Return Value

true if the new annotation handler is in a valid initial state for its subclass, **false** otherwise.

Header File

AVExpT.h

AVAnnotHandlerNotifyAnnotAddedToSelectionProc

```
ACCB1 void ACCB2 AVAnnotHandlerNotifyAnnotAddedToSelectionProc  
(AVAnnotHandler annotHandler, PDAnnot anAnnot,  
AVPageView pageView);
```

Description

(Optional) Callback for [AVAnnotHandler](#). It is called when an annotation is added to the selection, and should highlight the annotation. Set it to **NULL** if omitted.

To allow only a single annotation to select at a time, keep a global variable containing the selected annotation and on each invocation of **NotifyAnnotAddedToSelection** first deselect the current selection, if any (that is, if **selectedAnnot** is non-**NULL**, call its [AVAnnotHandlerNotifyAnnotRemovedFromSelectionProc](#) select the new annotation, and set **selectedAnnot**. Of course, **RemovedFrom** should set **selectedAnnot** to **NULL**.

Parameters

annotHandler	The annotation handler responsible for the annotation that was added to the selection.
anAnnot	The annotation that was added to the selection.
pageView	The AVPageView containing the annotation that was added to the selection.

Return Value

None

Header File

AVExpT.h

Related Methods

[AVDocSetSelection](#)

AVAnnotHandlerNotifyAnnotRemovedFromSelectionProc

```
ACCB1 void ACCB2
AVAnnotHandlerNotifyAnnotRemovedFromSelectionProc
(AVAnnotHandler annotHandler, PDAnnot anAnnot,
AVPageView pageView);
```

Description

(Optional) Callback for [AVAnnotHandler](#). It is called when an annotation is removed from the selection, and should unhighlight the annotation. Set it to **NULL** if omitted.

Parameters

annotHandler	The annotation handler responsible for the annotation that was removed from the selection.
anAnnot	The annotation that was removed from the selection.
pageView	The AVPageView in which the annotation appears.

Return Value

None

Header File

AVExpT.h

Related Methods

[AVDocClearSelection](#)

[AVDocSetSelection](#)



AVAnnotHandlerNotifyDestroyProc

```
ACCB1 void ACCB2 AVAnnotHandlerNotifyDestroyProc  
(AVAnnotHandler annotHandler);
```

Description

Currently unused.

Parameters

annotHandler	The annotation handler.

Return Value

None

Header File

AVExpT.h

AVAnnotHandlerPerformOpProc

```
ACCB1 ASBool ACCB2 AVAnnotHandlerPerformOpProc  
(AVAnnotHandler annotHandler, PDAnnot annot,  
AVPageView pageView, AVAnnotOp annotOp, AVAnnotOpData opData);
```

Description

Called to initiate the operation.

Parameters

annotHandler	The annotation handler responsible for this annotation.
annot	The annotation.
pageView	The AVPageView in which the annotation is located.
annotOp	The operation to be performed.
opData	The data associated with the operation.

Return Value

true if the operation is performed; **false** otherwise.

Header File

AVExpT.h

Related Callbacks

[AVAnnotHandlerCanPerformOpProc](#)

AVAnnotHandlerPtInAnnotViewBBoxProc

```
ACCB1 ASBool ACCB2 AVAnnotHandlerPtInAnnotViewBBoxProc  
(AVAnnotHandler annotHandler, AVPageView pageView,  
PDAnnot anAnnot, AVDevCoord xHit, AVDevCoord yHit);
```

Description

Callback for [AVAnnotHandler](#). It is called by [AVPageViewIsAnnotAtPoint](#) to determine whether a point is within an annotation. The annotation handler is free to determine what it means for the point to be “in” the annotation. For example, if the annotation appears only as the outline of a circle, the point may be “in” the annotation only when it is near the border of the circle, but not elsewhere within the circle.

NOTE: The coordinate numeric types have changed in Acrobat 6.0.

Parameters

annotHandler	The annotation handler responsible for this annotation.
pageView	The AVPageView in which the annotation appears.
anAnnot	The annotation being tested.
xHit	The x-coordinate of the point to test.
yHit	The y-coordinate of the point to test.

Return Value

true if the point is in the annotation, **false** otherwise.

Header File

`AVExpT.h`

Related Methods

[AVPageViewIsAnnotAtPoint](#)

AVAuxDataPerformProc

```
ACCB1 ASBool ACCB2 AVAuxDataPerformProc (ASAtom auxDataType,
void* auxData, ASInt32 auxDataLen, AVDoc avDoc);
```

Description

(Optional) Callback for [AVAuxDataHandler](#). It is called to process auxiliary data sent to the [AVDoc](#) using [AVDocSendAuxData](#). This callback must process the data appropriately for whatever [auxDataType](#) is sent.

If [NULL](#), default behavior is used.

Parameters

auxDataType	Specifies the type of auxData . This determines how auxData is interpreted.
auxData	The auxiliary data.
auxDataLen	The length of auxData , in bytes.
avDoc	The document with which the auxiliary data is associated.

Return Value

[true](#) if the data is acted upon, [false](#) if nothing is done.

Header File

[AVExpT.h](#)

Related Methods

[AVDocSendAuxData](#)



AVCmdHandlerInitProc

```
ACCB1 ASBool ACCB2 AVCmdHandlerInitProc (ASAtom handlerName);
```

Description

Initialize the command handler. Called once for each command handler registered.

Parameters

handlerName	The name of the command handler.

Return Value

true if initialization succeeds, **false** otherwise.

Header File

AVExpT.h

Related Callbacks

[AVCmdHandlerTermProc](#)



AVCmdHandlerTermProc

```
ACCB1 void ACCB2 AVCmdHandlerTermProc (ASAtom handlerName) ;
```

Description

Terminate the handler. Called once for each handler registered when Acrobat shuts down. Called before clients are unloaded.

Parameters

handlerName	The name of the handler being terminated.

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVCmdHandlerInitProc](#)



AVCommandCancelProc

```
ACCB1 AVCommandStatus ACCB2 AVCommandCancelProc  
(AVCommand cmd) ;
```

Description

Stop working and clean up as though the command executed to completion.

Parameters

<code>cmd</code>	The command being canceled.

Return Value

One of the [AVCommandStatus](#) codes.

Header File

`AVExpT.h`

Related Methods

[AVCommandGetCancelProc](#)
[AVCommandCancel](#)

AVCommandCreatedProc

```
ACCB1 void ACCB2 AVCommandCreatedProc (AVCommand cmd) ;
```

Description

Called after a command is created. The command handler can establish default parameters, and so forth, for the newly created command.

Parameters

cmd	The command that was created.

Return Value

None

Header File

`AVExpT.h`

Related Callbacks

[AVCommandDestroyProc](#)

AVCommandDestroyProc

```
ACCB1 void ACCB2 AVCommandDestroyProc (AVCommand cmd) ;
```

Description

Called before a command is destroyed. The command handler should free any memory allocated by the command.

Parameters

cmd	The command being destroyed.

Return Value

None

Header File

`AVExpT.h`

Related Callbacks

[AVCommandCreatedProc](#)

Related Methods

[AVCommandDestroy](#)

AVCommandGetProc

```
ACCB1 void ACCB2 AVCommandGetProc (AVCommand cmd, ASCab
theCab) ;
```

Description

Called to retrieve a cabinet from a command. Used in the **GetParams** and **GetProps** members of the **AVCommandHandlerRec** structure. When retrieving command parameters, the handler should first remove any existing items from **theCab** using **ASCabMakeEmpty** and then copy all parameter values from the command into **theCab**. When retrieving properties, the command handler should replace any entries in **theCab** with key names it recognizes with copies of the command-specific properties.

Parameters

cmd	The command whose procedure is retrieved.
theCab	(Filled by the callback) The appropriate command cabinet.

Return Value

None

Header File

AVExpT.h

AVCommandPostflightFileProc

```
ACCB1 AVCommandStatus ACCB2 AVCommandPostflightFileProc  
(AVCommand cmd, PDDoc doc);
```

Description

Every command in a sequence gets its “Postflight” command callback called after all commands in a given sequence have been executed but before the file is closed.

Preflights and Postflights are good for getting user data or preparing the command at the beginning of the sequence. For example, you could use the pre-flight to ask what password the “Add Security” command should use. This is important since you only want to ask once (not for every file), and you do not want to store the password in the sequence file (or the command’s persistent parameters).

See [AVCommandPreflightSequenceProc](#) for the order in which the [AVCommand](#) Pre/Postflight callbacks are called.

Parameters

<code>cmd</code>	The command.
<code>doc</code>	The PDDoc .

Return Value

One of the [AVCommandStatus](#) codes.

Header File

`AVExpT.h`

AVCommandPostflightSequenceProc

```
ACCB1 AVCommandStatus ACCB2 AVCommandPostflightSequenceProc  
(AVCommand cmd) ;
```

Description

Every command in a sequence gets its Postflight command callback called after the sequence is executed.

Preflights and Postflights are good for getting user data or preparing the command at the beginning of the sequence. For example, you could use the pre-flight to ask what password the “Add Security” command should use. This is important since you only want to ask once (not for every file), and you do not want to store the password in the sequence file (or the command’s persistent parameters).

See [AVCommandPreflightSequenceProc](#) for the order in which the [AVCommand](#) Pre/Postflight callbacks are called.

Parameters

<code>cmd</code>	The command.

Return Value

One of the [AVCommandStatus](#) codes.

Header File

`AVExpT.h`

AVCommandPreflightFileProc

```
ACCB1 AVCommandStatus ACCB2 AVCommandPreflightFileProc  
(AVCommand cmd, PDDoc doc);
```

Description

Every command in a sequence gets its “Preflight” callback called after each file has been opened to be processed but before any commands have been executed on that file. If any of the Preflight callbacks returns an error, the sequence is aborted.

Preflights and Postflights are good for getting user data or preparing the command at the beginning of the sequence. For example, you could use the pre-flight to ask what password the “Add Security” command should use. This is important since you only want to ask once (not for every file), and you do not want to store the password in the sequence file (or the command’s persistent parameters).

See [AVCommandPreflightSequenceProc](#) for the order in which the [AVCommand](#) Pre/Postflight callbacks are called.

Parameters

<code>cmd</code>	The command.
<code>doc</code>	The PDDoc .

Return Value

One of the [AVCommandStatus](#) codes.

Header File

`AVExpT.h`

AVCommandPreflightSequenceProc

```
ACCB1 AVCommandStatus ACCB2 AVCommandPreflightSequenceProc  
(AVCommand cmd) ;
```

Description

Every command in a sequence gets its Preflight callback called before the sequence is executed. If any of the Preflight callbacks returns an error, the sequence is aborted.

Preflights and Postflights are good for getting user data or preparing the command at the beginning of the sequence. For example, you could use the pre-flight to ask what password the “Add Security” command should use. This is important since you only want to ask once (not for every file), and you do not want to store the password in the sequence file (or the command’s persistent parameters).

The sequence call the procedures in this order:

1. Sequence begins
2. [AVCommandPreflightSequenceProcs](#) for all commands are called
3. Open File #1
4. [AVCommandPreflightFileProcs](#) for all commands are called
5. Execute all commands on given file
6. [AVCommandPostflightFileProcs](#) for all commands are called
7. Close File #1
8. Open File #2
9. ... (repeat pre/post file procs)
10. Close last file
11. [AVCommandPostflightSequenceProcs](#) for all commands are called.
12. Sequence ends

Parameters

<code>cmd</code>	The command.

Return Value

One of the [AVCommandStatus](#) codes.

Header File

`AVExpT.h`

AVCommandRegisterCommandsProc

```
ACCB1 void ACCB2 AVCommandRegisterCommandsProc  
(ASAtom handlerName);
```

Description

The application maintains a global list of commands that the user can choose from when building his own command. During the initialization sequence, this method registers all the commands that the registered command handler builds and wants included in the global command list.

Parameters

handlerName	The name of the command handler being registered.

Return Value

None

Header File

AVExpT.h

AVCommandResetProc

```
ACCB1 AVCommandStatus ACCB2 AVCommandResetProc  
(AVCommand cmd) ;
```

Description

Stop working, clear any errors, and try to get back into a Ready state. For many commands this is equivalent to canceling.

Parameters

<code>cmd</code>	The command being reset.

Return Value

One of the [AVCommandStatus](#) codes.

Header File

`AVExpT.h`

Related Callbacks

[AVCommandSetProc](#)

Related Methods

[AVCommandReset](#)

AVCommandSetProc

```
ACCB1 AVCommandStatus ACCB2 AVCommandSetProc (AVCommand cmd,
ASCab cab) ;
```

Description

Called to set a cabinet within a command. Used in the **SetParams** member of the [AVCommandHandlerRec](#) structure. The command handler should copy any information from the cabinet into the command. It must not destroy or modify the cabinet.

Parameters

cmd	The command.
cab	The cabinet to store.

Return Value

One of the **AVCommandStatus** codes.

Header File

AVExpT.h

Related Callbacks

[AVCommandResetProc](#)

AVCommandShowDialogProc

```
ACCB1 AVCommandStatus ACCB2 AVCommandShowDialogProc  
(AVCommand cmd) ;
```

Description

Display this command's parameter setting dialog and allow the user to alter the parameters.

Parameters

cmd	The command whose parameter setting dialog is displayed.

Return Value

One of the **AVCommandStatus** codes.

Header File

AVExpT.h

AVCommandWorkProc

```
ACCB1 AVCommandStatus ACCB2 AVCommandWorkProc (AVCommand cmd) ;
```

Description

Do some work. If you don't finish your work, return **kAVCommandWorking**. If you do finish your work, return **kAVCommandDone**. If the user cancels the operation, return **kAVCommandCanceled**. If an error occurs, return **kAVCommandInError**.

In most cases this method performs its work until it returns **kAVCommandDone**, but in some cases it may be called on to cancel or reset before its work is done.

Parameters

cmd	The command doing some work.

Return Value

One of the **AVCommandStatus** codes.

Header File

`AVExpT.h`

Related Methods

[AVCommandWork](#)

AVComputeEnabledProc

```
ACCB1 ASBool ACCB2 AVComputeEnabledProc (void* data);
```

Description

Callback that is used to determine whether a menu item, toolbar button, or tool is enabled. If used for a tool, it is one of the optional callbacks for [AVTool](#).

This procedure is called every time the menu or toolbar button is displayed, so it should not do compute-time-intensive processing. It is called before the menu item or toolbar button is displayed, or before a tool is activated. If it returns **false**, the menu item, toolbar button, or tool is disabled; otherwise it is enabled. If this callback is **NULL**, the menu item, toolbar button, or tool is always enabled.

Each menu item, toolbar button, or tool can have its own **AVComputeEnabledProc**, or they can be shared.

Parameters

data	User-supplied data that was passed in the call to AVMenuItemSetComputeEnabledProc or AVToolButtonSetComputeEnabledProc .

Return Value

true if the menu item, toolbar button, or tool is enabled, **false** otherwise.

Header File

AVExpT.h

Related Methods

[AVMenuItemSetComputeEnabledProc](#)
[AVToolButtonSetComputeEnabledProc](#)

AVComputeMarkedProc

```
ACCB1 ASBool ACCB2 AVComputeMarkedProc (void* data);
```

Description

Callback that is used to determine whether a menu item or toolbar button is marked (a marked menu item has a check mark next to it, and a marked toolbar button appears selected). It is called before the menu item or toolbar button is displayed. If it returns **false**, the menu item or toolbar button is not marked, otherwise it is marked.

Each menu item and toolbar button can have its own **AVComputeMarkedProc**, or they can be shared.

Parameters

data	User-supplied data that was passed in the call to AVMenuItemSetComputeMarkedProc or AVToolButtonSetComputeMarkedProc .

Return Value

true if the menu item or toolbar button is marked, **false** otherwise.

Header File

`AVExpT.h`

Related Methods

[AVMenuItemSetComputeMarkedProc](#)
[AVToolButtonSetComputeMarkedProc](#)

AVComputeVisibleProc

```
ACCB1 ASBool ACCB2 AVComputeVisibleProc (void* data);
```

Description

Callback that is used to determine whether a toolbar button, menu item, or HowTo panel is visible when its parent is opened. It is called before the item is displayed. If it returns **true**, the item is visible, otherwise it is not visible.

Each toolbar button, menu item, or HowTo panel can have its own visibility procedure, or they can be shared.

Because the procedure is called whenever the item is displayed, it should not do any resource-intensive computing.

Parameters

data	User-supplied data that was passed in the call to AVMenuItemSetComputeVisibleProc , AVToolButtonSetComputeVisibleProc , or AVAppSetHowToPanelComputeVisibleProc .

Return Value

true if the item is visible, **false** otherwise.

Header File

AVExpT.h

Related Methods

[AVAppSetHowToPanelComputeVisibleProc](#)
[AVMenuItemSetComputeVisibleProc](#)
[AVToolButtonSetComputeVisibleProc](#)

AVContextMenuAdditionProc

```
ACCB1 void ACCB2 AVContextMenuAdditionProc (ASAtom menuName,  
AVMenu menu, void *menuData, void *clientData);
```

Description

Called after a context menu has been created but before it is shown to the user. The callback can add menu items to or remove menu items from the menu. The client must register the callback using [AVAppRegisterForContextMenuAddition](#).

This callback should not raise an error.

Parameters

menuName	The menu name. One of: ● Page:The standard context menu for an AVPageView. ● Select:The context menu for selected text.
menu	The menu object.
menuData	The menu data as an AVDoc pointer.
clientData	User-supplied data that was passed in the call to AVAppRegisterForContextMenuAddition .

Return Value

None

Header File

AVExpT.h

Related Methods

[AVAppRegisterForContextMenuAddition](#)

AVConversionConvertFromPDFProc

```
ACCB1 AVConversionStatus ACCB2 AVConversionConvertFromPDFProc
(ASCab settings, AVConversionFlags flags, PDDoc doc,
ASPathName path, ASFileSys fileSys,
AVStatusMonitorProcs statusMonitor,
AVConversionClientData clientData);
```

Description

Called to convert a PDF file to a another file format.

Parameters

settings	An ASCab containing the settings for the conversion operation. Can be NULL . The implementation should use these settings rather than defaults since the batch framework may have provided custom settings.
flags	Indicates any non-default behavior to apply to the conversion. By default, conversions are synchronous, non-interactive, and do not display a settings dialog. The conversion framework will automatically call your settings dialog if kAVConversionSyncPopSettingsDialog is set—do not pop your settings dialog in your convert proc.
doc	The document that is to be converted.
path	The desired location for the output file.
fileSys	The file system from which path was obtained.
statusMonitor	Contains the progress monitor, cancel proc, and error reporting proc to be used by the converter. Can be NULL . If an error occurs during conversion, the implementation should not raise or throw an error but instead report the error using the reportProc , if it is available. The report proc member of the status monitor can be NULL , so developers should check for that condition before calling it.
clientData	The user-defined data that is provided to all AVConversionHandler callbacks.

Return Value

One of the **AVConversionStatus** codes.

Header File

AVExpT.h

**Related Callbacks**[AVConversionConvertToPDFProc](#)**Related Methods**[AVConversionConvertFromPDFWithHandler](#)

AVConversionConvertStreamFromStructNodeProc

```
ACCB1 AVConversionStatus ACCB2
AVConversionConvertStreamFromStructNodeProc (ASCab inSettings,
AVConversionFlags flags, AVStructNode inStructNode,
ASStm stream, ASCab metaData,
AVStatusMonitorProcs statusMonitor,
AVConversionClientData clientData);
```

Description

Called to convert a structure subtree rooted at a given node to a stream.

Parameters

inSettings	An ASCab containing the settings for the conversion operation. Can be NULL . The implementation should use these settings rather than defaults since the batch framework may have provided custom settings.
flags	Bit flags for any non-default behavior to apply to the conversion. By default, conversions are synchronous, non-interactive, and do not display a settings dialog. The conversion framework calls your settings dialog if kAVConversionSyncPopSettingsDialog is set; this procedure should not open the settings dialog.
inStructNode	The structure node to be converted.
stream	The output stream.
metaData	An ASCab containing metadata about the input stream, such as, for example, a key url containing the URL of HTML data. Can be NULL .
statusMonitor	Contains the progress monitor, cancel proc, and error reporting proc to be used by the converter. Can be NULL . If an error occurs during conversion, the implementation should not raise or throw an error but instead report the error using the reportProc , if it is available.
clientData	The user-defined data that is provided to all AVConversionHandler callbacks.

Return Value

One of the **AVConversionStatus** codes.

Header File

`AVExpT.h`

Related Callbacks

[AVConversionConvertStreamToPDFProc](#)

Related Methods

[AVConversionConvertStreamFromStructNodeWithHandler](#)

AVConversionConvertStreamToPDFProc

```
ACCB1 AVConversionStatus ACCB2
AVConversionConvertStreamToPDFProc (ASCab inSettings,
AVConversionFlags flags, ASStm stream, ASCab metaData,
PDDoc *outPDDoc, AVStatusMonitorProcs statusMonitor,
AVConversionClientData clientData);
```

Description

Called to convert a non-PDF file to a PDF file.

Parameters

inSettings	An ASCab containing the settings for the conversion operation. Can be NULL . The implementation should use these settings rather than defaults since the batch framework may have provided custom settings.
flags	Bit flags for any non-default behavior to apply to the conversion. By default, conversions are synchronous, non-interactive, and do not display a settings dialog. The conversion framework calls your settings dialog if kAVConversionSyncPopSettingsDialog is set; this procedure should not open the settings dialog.
stream	The input stream.
metaData	An ASCab containing metadata about the input stream, such as, for example, a key url containing the URL of HTML data. Can be NULL .
doc	The output PDDoc . The implementation should not clean up this PDDoc —Acrobat will do this.
statusMonitor	Contains the progress monitor, cancel proc, and error reporting proc to be used by the converter. Can be NULL . If an error occurs during conversion, the implementation should not raise or throw an error but instead report the error using the reportProc , if it is available.
clientData	The user-defined data that is provided to all AVConversionHandler callbacks.

Return Value

One of the **AVConversionStatus** codes.

**Header File**

`AVExpT.h`

Related Callbacks

[AVConversionConvertToPDFProc](#)

Related Methods

[AVConversionConvertToPDFWithHandler](#)

AVConversionConvertToPDFProc

```
ACCB1 AVConversionStatus ACCB2 AVConversionConvertToPDFProc
(ASCab settings, ASPathName path, ASFileSys fileSys,
PDDoc* doc, AVStatusMonitorProcs statusMonitor,
AVConversionClientData clientData);
```

Description

Called to convert a non-PDF file to a PDF file.

Parameters

settings	An ASCab containing the settings for the conversion operation. Can be NULL . The implementation should use these settings rather than defaults since the batch framework may have provided custom settings.
path	The location of the input file.
fileSys	The file system from which path was obtained.
doc	The output PDDoc . The implementation should not clean up this PDDoc —Acrobat will do this.
statusMonitor	Contains the progress monitor, cancel proc, and error reporting proc to be used by the converter. Can be NULL . If an error occurs during conversion, the implementation should not raise or throw an error but instead report the error using the reportProc , if it is available.
clientData	The user-defined data that is provided to all AVConversionHandler callbacks.

Return Value

One of the **AVConversionStatus** codes.

Header File

AVExpT.h

Related Callbacks

[AVConversionConvertFromPDFProc](#)

Related Methods

[AVConversionConvertToPDFWithHandler](#)

AVConversionDefaultSettingsProc

```
ACCB1 ASCab ACCB2 AVConversionDefaultSettingsProc  
(const char* filterDescription,  
AVConversionClientData clientData);
```

Description

Called to get the default settings for the conversion operation.

It is the caller's responsibility to release the resources associated with the returned **ASCab**.

Parameters

filterDescription	A string that represents the filterDescription parameter of the AVFileFilterRec for the conversion handler.
clientData	The user-defined data that is provided to all AVConversionHandler callbacks.

Return Value

An **ASCab** containing the default settings for the conversion operation, return **NULL** to indicate none.

Header File

`AVExpT.h`

AVConversionFromPDFEnumProc

```
ACCB1 ASBool ACCB2 AVConversionFromPDFEnumProc  
(AVConversionFromPDFHandler handler,  
AVConversionEnumProcData data);
```

Description

Called once for each [AVConversionFromPDFHandler](#) registered with Acrobat, or until the callback returns **false** to halt the enumeration.

Parameters

handler	The AVConversionFromPDFHandler .
data	User-defined data passed to the call to AVConversionEnumFromPDFConverters .

Return Value

true to continue the enumeration, **false** otherwise.

Header File

[AVExpT.h](#)

Related Callbacks

[AVConversionToPDFEnumProc](#)

Related Methods

[AVConversionEnumFromPDFConverters](#)

AVConversionParamDescProc

```
ACCB1 void ACCB2 AVConversionParamDescProc  
(const ASCab settings, ASCab paramDesc,  
AVConversionClientData clientData);
```

Description

Called to obtain conversion parameter information.

Parameters

settings	A read-only ASCab containing the requested parameters.
paramDesc	<i>(Filled by the callback)</i> The parameter descriptions (ASText objects) stored under numeric keys starting with key "1". For example, key="1", value="Title: API Reference" (ASText object).
clientData	The user-defined data that is provided to all AVConversion callbacks.

Return Value

None

Header File

AVCalls.h

AVConversionSettingsDialogProc

```
ACCB1 ASBool ACCB2 AVConversionSettingsDialogProc  
(ASCab settings, AVConversionClientData clientData);
```

Description

Called to request the handler to display its settings dialog, if it has one. An **ASCab** containing conversion settings is passed in to fill in the dialog.

The implementation should use these settings. Ensure to use this cabinet of settings rather than defaults since the batch framework may provide different settings.

If the user commits changes, the settings should be stored in the **ASCab** that was provided.

- For "ConvertToPDF" handlers, two keys are present in the settings **ASCab**:
 - ASPathName** — Path to input file.
 - ASFileSys** — The associated file system.
- For "ConvertFromPDF" handlers, three keys are present in the settings **ASCab**:
 - PDDoc** — Input **PDDoc**.
 - ASPathName** — Output path.
 - ASFileSys** — The associated file system.

Parameters

settings	The ASCab used to populate the dialog.
clientData	The user-defined data that is provided to all AVConversion callbacks.

Return Value

true to proceed with the conversion, **false** otherwise.

Header File

AVExpT.h

AVConversionToPDFEnumProc

```
ACCB1 ASBool ACCB2 AVConversionToPDFEnumProc  
(AVConversionToPDFHandler handler,  
AVConversionEnumProcData data);
```

Description

Called once for each **AVConversionToPDFHandler** registered with Acrobat, or until the callback returns **false** to halt the enumeration.

Parameters

handler	The AVConversionToPDFHandler .
data	User-defined data passed to the call to AVConversionEnumToPDFConverters .

Return Value

true to continue the enumeration, **false** otherwise.

Header File

`AVExpT.h`

Related Callbacks

[AVConversionFromPDFEnumProc](#)

Related Methods

[AVConversionEnumToPDFConverters](#)



AVDocEnumProc

```
ACCB1 ASBool ACCB2 AVDocEnumProc (AVDoc doc,  
void* clientData);
```

Description

Callback used by [AVAppEnumDocs](#). It is called once for each open **AVDoc**.

Parameters

doc	The current document. Do not close this AVDoc in this callback function.
clientData	User-supplied data that was passed in the call to AVAppEnumDocs .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

AVExpT.h

Related Methods

[AVAppEnumDocs](#)

AVDocPermReqProc

```
ACCB1 PDPermReqStatus ACCB2 AVDocPermReqProc (AVDoc doc,
PDPermReqObj obj, PDPermReqOpr opr);
```

Description

A callback that can be associated with an **AVDoc** when it is opened (via an **AVDocOpenParamsRec**). It can restrict the set operations allowed on the document. When **AVDocPermRequest** is called, this callback is be consulted to deny or grant the permission. If it denies permission, **AVDocPermRequest** will also deny permission. If it grants permission, the security handler for the document will be consulted to determine the result of **AVDocPermRequest**. This callback can only deny permissions allowed by the security handler; it cannot grant permissions that the security handler does not grant.

Parameters

doc	The current document.
obj	Description of target object.
opr	Description of target operation.

Return Value

The status.

Header File

AVExpT.h

AVDocSelectionAcquireQuadsProc

```
ACCBPROTO1 ASArraySize (ACCBPROTO2  
*AVDocSelectionAcquireQuadsProc) (AVDoc doc, void* data,  
ASInt32 pageNum, ASFixedQuad** quads);
```

Description

Callback for **AVDocSelectionServer** that gets a quad-based region for the selection.

NOTE: This procedure is new in Acrobat 6.0.

Parameters

doc	The document containing the selection.
data	The current selection data. Its content and organization is up to the selection server for the current selection type.
pageNum	The page number of the selection.
quads	<i>(Filled by the method.)</i> A pointer to an array of quads, or NULL . ● If non-NULL, the selection server allocates an array of ASFixedQuads in user space describing the selection, and stores a pointer to the array at *quad. ● If NULL, the selection server returns the number of quads in the selection without allocating an array.

Return Value

The number of quads in the selection.

Header File

AVExpT.h

AVDocSelectionAddedToSelectionProc

```
ACCB1 void* ACCB2 AVDocSelectionAddedToSelectionProc
(AVDoc doc, void* curData, void* addData, ASBool highlight);
```

Description

Callback for [AVDocSelectionServer](#). Adds the specified item to the selection, highlights it, and returns the new selection containing the newly-added item.

Parameters

doc	The document containing the data to add to the selection.
curData	Data representing the current selection. Its format is specific to the selection server.
addData	Item to add to the selection.
highlight	true if the selection should be highlighted (because it has not already been highlighted), false if the selection should not be highlighted (because it has already been highlighted by whoever called this callback). See AVDocSetSelection for additional information on highlighting.

Return Value

New selection data containing all current selections (that is, the previous selection plus the newly-added selection), or **NULL** if failure. If the selection server allows only a single item to be selected at a time, clear the previous selection, highlight the selection specified by **addData** (if **highlight** is **true**), and simply return **addData**.

Header File

AVExpT.h

Related Callbacks

[AVDocSelectionRemovedFromSelectionProc](#)

AVDocSelectionCanCopyProc

```
ACCB1 ASBool ACCB2 AVDocSelectionCanCopyProc (AVDoc doc,  
void* selData);
```

Description

Callback for [AVDocSelectionServer](#). It is used to determine whether the current selection can be copied. This controls, for example, whether the **Copy** menu item is enabled.

The **Copy** menu item is only enabled if the selection server's [AVDocSelectionCanCopyProc](#) returns **true** and the selection server has an [AVDocSelectionCopyProc](#).

Parameters

<code>doc</code>	The document containing the selection.
<code>selData</code>	The current selection data.

Return Value

true if the current selection can be copied, **false** otherwise.

Header File

`AVExpT.h`

Related Callbacks

[AVDocSelectionCopyProc](#)

Related Methods

[AVDocCopySelection](#)

AVDocSelectionCanCutProc

```
ACCB1 ASBool ACCB2 AVDocSelectionCanCutProc (AVDoc doc,  
void* data);
```

Description

Callback for [AVDocSelectionServer](#). It is used to determine whether the current selection can be cut. This controls, for example, whether the **Cut** menu item is enabled.

The **Cut** menu item is only enabled if the selection server's [AVDocSelectionCanCutProc](#) returns **true** and the selection server has an [AVDocSelectionCutProc](#).

Parameters

doc	The document containing the current selection.
data	The current selection data.

Return Value

true if the current selection can be cut, **false** otherwise.

Header File

`AVExpT.h`

Related Callbacks

[AVDocSelectionCutProc](#)

[AVDocSelectionCanPasteProc](#)

AVDocSelectionCanDeleteProc

```
ACCB1 ASBool ACCB2 AVDocSelectionCanDeleteProc (AVDoc doc,  
void* selData);
```

Description

Callback for [AVDocSelectionServer](#). It is used to determine whether the current selection can be deleted. This controls, for example, whether the **Delete** menu item is enabled.

The **Delete** menu item is only enabled if the selection server's [AVDocSelectionCanDeleteProc](#) returns **true** and the selection server has an [AVDocSelectionDeleteProc](#).

Parameters

<code>doc</code>	The document containing the current selection.
<code>selData</code>	The current selection data.

Return Value

true if the current selection can be deleted, **false** otherwise.

Header File

`AVExpT.h`

Related Callbacks

[AVDocSelectionCanDeleteProc](#)

Related Methods

[AVDocDeleteSelection](#)

AVDocSelectionCanPasteProc

```
ACCB1 ASBool ACCB2 AVDocSelectionCanPasteProc (AVDoc doc);
```

Description

Callback for [AVDocSelectionServer](#). It is used to determine whether the current selection can be pasted. This controls, for example, whether the **Paste** menu item is enabled.

The **Paste** menu item is only enabled if the selection server's [AVDocSelectionCanPasteProc](#) returns **true** and the selection server has an [AVDocSelectionPasteProc](#).

Parameters

<code>doc</code>	The document into which the selection is pasted.

Return Value

true if the data currently on the clipboard can be pasted, **false** otherwise.

Header File

`AVExpT.h`

Related Callbacks

[AVDocSelectionPasteProc](#)
[AVDocSelectionCanCutProc](#)

AVDocSelectionCanPropertiesProc

```
ACCB1 ASBool ACCB2 AVDocSelectionCanPropertiesProc (AVDoc doc,  
void* selData);
```

Description

Callback for [AVDocSelectionServer](#). It is used to determine whether the current selection has user-specified properties. This controls whether the “**Properties...**” menu item is enabled.

The “**Properties...**” menu item will not be enabled if the selection server does not have a [AVDocSelectionPropertiesProc](#) callback.

Parameters

<code>doc</code>	The document containing the current selection.
<code>selData</code>	The current selection data.

Return Value

true if the current selection has a Properties UI, **false** otherwise.

Header File

`AVExpT.h`

Related Methods

[AVDocDoSaveAsWithParams](#)

AVDocSelectionCanSelectAllProc

```
ACCB1 ASBool ACCB2 AVDocSelectionCanSelectAllProc (AVDoc doc,  
void* selData);
```

Description

Callback for [AVDocSelectionServer](#). It is used to determine whether the current selection type can perform a “select all” operation. This controls whether the **Select All** menu item is enabled.

Parameters

doc	The document containing the current selection.
selData	The current selection’s data.

Return Value

true if “select all” can be performed on the current selection type, **false** otherwise.

Header File

`AVExpT.h`

Related Callbacks

[AVDocSelectionSelectAllProc](#)

AVDocSelectionCopyProc

```
ACCB1 ASBool ACCB2 AVDocSelectionCopyProc (AVDoc doc,  
void* selData);
```

Description

Callback for [AVDocSelectionServer](#). It copies the selected item to the clipboard. The Acrobat viewer will have already cleared the clipboard and placed some private data onto it so that it can identify the selection server that put data onto the clipboard. Because of this, a client must not clear the clipboard, but only add its private data. In addition, if the current selection can reasonably be represented as text, clients are strongly encouraged to place a text representation of the selection onto the clipboard, in addition to their own private format.

Parameters

doc	The document whose selection is copied.
selData	The current selection data in doc .

Return Value

true if the data was actually copied, **false** otherwise.

Header File

AVExpT.h

Related Methods

[AVDocCopySelection](#)
[UnixAppClipboardGetItemId](#)

AVDocSelectionCutProc

```
ACCB1 ASBool ACCB2 AVDocSelectionCutProc (AVDoc doc,  
void* data);
```

Description

Callback for [AVDocSelectionServer](#). Cuts the current selection. See the discussion under [AVDocSelectionCopyProc](#) for information on how the selection server must use the clipboard.

Parameters

doc	Document whose selection is cut.
data	The current selection data in doc .

Return Value

true if the data was actually cut, **false** otherwise.

Header File

`AVExpT.h`

Related Methods

[UnixAppClipboardGetItemId](#)

AVDocSelectionDeleteProc

```
ACCB1 ASBool ACCB2 AVDocSelectionDeleteProc (AVDoc doc,  
void* selData);
```

Description

Callback for [AVDocSelectionServer](#). Deletes the current selection.

Parameters

doc	Document whose selection is deleted.
selData	The current selection in doc .

Return Value

true if the data was actually deleted, **false** otherwise.

Header File

AVExpT.h

Related Methods

[AVDocDeleteSelection](#)

AVDocSelectionEnumPageRangesProc

```
ACCB1 void ACCB2 AVDocSelectionEnumPageRangesProc (AVDoc doc,  
void* selectionData, AVSelectionPageRangeEnumProc enumProc,  
void* clientData);
```

Description

Callback for [AVDocSelectionServer](#). It allows enumeration of the set of pages the selection covers.

Parameters

doc	The document containing the selection.
selectionData	The current selection data. Its content and organization is up to the selection server for the current selection type.
enumProc	The current selection data. Its content and organization is up to the selection server for the current selection type.
clientData	User-supplied data that was passed in the call to AVDocSelectionEnumPageRanges .

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVDocSelectionEnumSelectionProc](#)

AVDocSelectionEnumSelectionProc

```
ACCB1 void ACCB2 AVDocSelectionEnumSelectionProc (AVDoc doc,  
void* data, AVSelectionEnumProc proc, void* clientData);
```

Description

(Optional) Callback for [AVDocSelectionServer](#). Called by [AVDocEnumSelection](#). This callback enumerates the current selection, calling the specified [AVSelectionEnumProc](#) for each “item” in the selection (the selection server is free to decide what constitutes an item).

If omitted, the selection is enumerated by calling **proc** once, passing the entire selection to it.

Parameters

doc	The document whose selection is enumerated.
data	The current selection in doc .
proc	The procedure to call for each item in the selection. This callback must halt enumeration if proc returns false , and continue enumeration if proc returns true .
clientData	User-supplied data that was passed in the call to AVDocEnumSelection . Pass this as the clientData each time proc is called.

Return Value

None

Header File

[AVExpT.h](#)

Related Callbacks

[AVDocSelectionEnumPageRangesProc](#)

Related Methods

[AVDocEnumSelection](#)

AVDocSelectionGetAVRectProc

```
ACCB1 ASBool ACCB2 AVDocSelectionGetAVRectProc(AVDoc doc,  
PDPageNumber pageNo, AVRect* rect, void* selData);
```

Description

Called to identify the bounding rectangle of a selection. Used by the Info palette to display the width and height of the selection.

Rectangle coordinates are in device space.

NOTE: The page number numeric type has changed in Acrobat 6.0.

Parameters

doc	The document containing the selection.
pageNo	The number of the page containing the bounding rectangle.
rect	<i>(Filled by the callback)</i> The bounding rect of the selection.
selData	Server-dependent selection data.

Return Value

true if the bounding rect was successfully determined, **false** otherwise.

Header File

AVExpT.h

AVDocSelectionGetSelectionTypeProc

```
ACCB1 ASAtom ACCB2 AVDocSelectionGetSelectionTypeProc  
(AVDoc doc, void* data);
```

Description

Callback for [AVDocSelectionServer](#). It provides a way for a single selection server to register different selection types based on the selection data. If this callback is not supplied, the selection type defaults to the return value from [AVDocSelectionGetTypeProc](#).

This callback does not affect existing selection servers.

Parameters

doc	The document containing the selection.
data	The current selection data. Its content and organization is up to the selection server for the current selection type.

Return Value

None

Header File

[AVExpT.h](#)

Related Callbacks

[AVDocSelectionGetTypeProc](#)

AVDocSelectionGettingSelectionProc

```
ACCB1 void ACCB2 AVDocSelectionGettingSelectionProc
(AVDoc doc, void* selData, ASBool highlight);
```

Description

Callback for [AVDocSelectionServer](#). It is called when the selection is set (for example, via [AVDocSetSelection](#)).

Along with whatever else this callback chooses to do, it must highlight the specified selection (if requested), using the selection server's [AVDocSelectionHighlightSelectionProc](#) callback.

Parameters

<code>doc</code>	The document containing the selection.
<code>selData</code>	The selection data being added.
<code>highlight</code>	If <code>true</code> , highlight the selection, <code>false</code> otherwise.

Return Value

None

Header File

`AVExpT.h`

Related Methods

[AVDocSetSelection](#)

AVDocSelectionGetTypeProc

```
ACCB1 ASAtom ACCB2 AVDocSelectionGetTypeProc (void);
```

Description

Callback for [AVDocSelectionServer](#). Returns the selection type this server handles (for example, "Text" or "Bookmark"). This information is used so that the Acrobat viewer knows which selection server to call.

Parameters

None

Return Value

The selection type this selection server handles.

Header File

AVExpT.h

Related Methods

[AVDocGetSelectionServerByType](#)

AVDocSelectionHighlightSelectionExProc

```
ACCB1 void ACCB2 AVDocSelectionHighlightSelectionExProc  
(AVDoc doc, AVPageView pageView, void* data);
```

Description

NOTE: Supersedes [AVDocSelectionHighlightSelectionProc](#) in Acrobat 6.0.

Callback for [AVDocSelectionServer](#) that highlights the selection.

This function is called when an update event is being processed for a page view. The page view passed into this call is the one being updated, which might not be the active page view returned by [AVDocGetPageView](#).

Parameters

doc	The document containing the selection.
pageView	The page view being updated.
data	The current selection data. Its content and organization is up to the selection server for the current selection type.

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVDocSelectionGettingSelectionProc](#)

[AVDocSelectionLosingSelectionProc](#)

AVDocSelectionHighlightSelectionProc

```
ACCB1 void ACCB2 AVDocSelectionHighlightSelectionProc  
(AVDoc doc, void* data);
```

Description

NOTE: Superseded by [AVDocSelectionHighlightSelectionExProc](#) in Acrobat 6.0.

(Previously known as **AVDocHighlightSelectionProc**) Callback for [AVDocSelectionServer](#). It highlights the selection. This method is unnecessary if the selection type highlights itself as, for example, annotations do.

Parameters

doc	The document containing the selection.
data	The current selection data. Its content and organization is up to the selection server for the current selection type.

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVDocSelectionHighlightSelectionExProc](#)
[AVDocSelectionGettingSelectionProc](#)
[AVDocSelectionLosingSelectionProc](#)

AVDocSelectionKeyDownProc

```
ACCB1 ASBool ACCB2 AVDocSelectionKeyDownProc (AVDoc doc,  
void* data, AVKeyCode key, AVFlagBits16 flags);
```

Description

(Optional) Callback for [AVDocSelectionServer](#). Handles a key press. Needed only if the selection server processes key presses.

NOTE: The **key** and **flags** numeric types have changed in Acrobat 6.0.

Parameters

doc	The document in which the click occurred.
data	The current selection data for doc .
key	The key that was pressed.
flags	Modifier keys that were pressed with key. Must be an OR of the Modifier Keys values.

Return Value

true if it the keypress was handled, **false** if it was not and therefore needs to be passed to the next procedure in the key handling chain.

Header File

AVExpT.h

AVDocSelectionLosingSelectionProc

```
ACCB1 void ACCB2 AVDocSelectionLosingSelectionProc (AVDoc doc,  
void* selData, ASBool highlight);
```

Description

Callback for [AVDocSelectionServer](#). This method is called by (among others) [AVDocClearSelection](#), to let the selection server responsible for the old selection do whatever cleanup it needs.

Along with whatever else this callback chooses to do, it must de-highlight the specified selection (if requested), using the selection server's [AVDocSelectionHighlightSelectionProc](#) callback.

Parameters

doc	The document whose selection is cleared.
selData	The current selection data.
highlight	If true , the selection specified by selData should be de-highlighted, false otherwise.

Return Value

None

Header File

AVExpT.h

Related Methods

[AVDocClearSelection](#)

AVDocSelectionPasteProc

```
ACCB1 void ACCB2 AVDocSelectionPasteProc (AVDoc doc);
```

Description

Callback for [AVDocSelectionServer](#). Pastes the current selection from the clipboard.

Parameters

doc	Document into whose selection the clipboard is pasted.

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVDocSelectionCutProc](#)

[AVDocSelectionCanPasteProc](#)

AVDocSelectionPropertiesProc

```
ACCB1 void ACCB2 AVDocSelectionPropertiesProc (AVDoc doc,  
void* selData);
```

Description

(Optional) Callback for [AVDocSelectionServer](#). Displays the “set properties” user interface, if any, for the selection server and lets the user set the server’s properties. This callback is not needed unless the selection server has properties that can be set by the user (for example, text highlight color). This callback is called by [AVDocDoSaveAsWithParams](#).

Parameters

doc	The document in which the selection server’s properties are set.
selData	The current selection data.

Return Value

None

Header File

AVExpT.h

Related Methods

[AVDocDoSaveAsWithParams](#)

AVDocSelectionRemovedFromSelectionProc

```
ACCB1 void* ACCB2 AVDocSelectionRemovedFromSelectionProc  
(AVDoc doc, void* curData, void* remData, ASBool highlight);
```

Description

Callback for [AVDocSelectionServer](#). De-highlights the old item given in **remData**, and returns a new **curData** or **NULL** if failure.

Parameters

doc	The document in which an item is removed from the selection.
curData	The current selection data.
remData	The item to remove from the selection. The content and format of selData differs for each selection server, and is up to the selection server's implementors.
highlight	If true , the item removed should be de-highlighted. If false , it should not.

Return Value

The new selection data (after the specified item has been removed).

Header File

AVExpT.h

Related Callbacks

[AVDocSelectionAddedToSelectionProc](#)

AVDocSelectionSelectAllProc

```
ACCB1 void* ACCB2 AVDocSelectionSelectAllProc (AVDoc doc,  
void* selData);
```

Description

Callback for [AVDocSelectionServer](#). Selects all items of the current type.

Parameters

doc	The document in which the “Select All” is performed.
selData	The current selection data in doc .

Return Value

The new selection data, after all items of the specified type have been selected.

Header File

AVExpT.h

Related Callbacks

[AVDocSelectionCanSelectAllProc](#)

AVDocSelectionShowMenuProc

```
ACCB1 ASBool ACCB2 AVDocSelectionShowMenuProc (AVDoc doc,  
void* selData, AVDevCoord x, AVDevCoord y);
```

Description

Called to request that the selection server display a context menu appropriate for the current selection.

The given coordinates provide a suggested location for displaying the menu and are in device space for the current [AVPageView](#).

NOTE: The coordinate types have changed in Acrobat 6.0.

Parameters

doc	The document containing the selection.
selData	Server-dependent selection data.
x	The x-coordinate of the point specifying the upper left corner of the menu.
y	The y-coordinate of the point specifying the upper left corner of the menu.

Return Value

true if the server showed the menu successfully, **false** otherwise.

Header File

AVExpT.h

AVDocSelectionShowSelectionProc

```
ACCB1 void ACCB2 AVDocSelectionShowSelectionProc (AVDoc doc,  
void* data);
```

Description

Callback for [AVDocSelectionServer](#). Changes the view (for example, by scrolling the current page or moving to the appropriate page) so that the current selection is visible.

Parameters

doc	The document whose selection is displayed.
data	The current selection data in doc .

Return Value

None

Header File

AVExpT.h

Related Methods

[AVDocShowSelection](#)



AVExecuteProc

```
ACCB1 void ACCB2 AVExecuteProc (void* data);
```

Description

Callback that is called whenever a menu item or toolbar button is executed. It implements whatever the menu item or toolbar button does (for example, opening a file or initiating a search).

This method may also be called from an external application displaying a PDF file in its window, using the [ExternalDocServerCreationData](#) structure.

Parameters

data	User-supplied data that was passed when AVMenuItemSetExecuteProc or AVToolButtonSetExecuteProc were called.

Return Value

None

Header File

AVExpT.h

Related Methods

[AVMenuItemSetExecuteProc](#)
[AVToolButtonSetExecuteProc](#)

AVIconHandlerGetFlagsProc

```
ACCB1 AVFlagBits32 (ACCB2 *AVIconHandlerGetFlagsProc)
(AVIconBundle6 bundle) ;
```

Description

Callback for [AVIconHandler](#). It returns the flags value for the icon.

Parameters

bundle	The icon bundle for which the flags are obtained. The following flags are defined: AVICON_DONT_CACHE=1

Return Value

The flags value.

Header File

AVExpT.h

Related Methods

[AVAppCreateIconBundle6](#)

AVIconHandlerMeasureProc

```
ACCB1 void (ACCB2 *AVIconHandlerMeasureProc) (AVIconBundle6
bundle, ASInt32 *w, ASInt32 *h);
```

Description

Callback for [AVIconHandler](#) that retrieves the measurements of the icon. All icons in the bundle are assumed to be the same size.

Parameters

bundle	The icon bundle for which the icon size is measured.
w	<i>(Filled by the method)</i> The icon width in pixels.
h	<i>(Filled by the method)</i> The icon height in pixels.

Return Value

None

Header File

```
AVExpT.h
```

Related Methods

```
AVAppCreateIconBundle6
```

AVIconHandlerOpenStmProc

```
ACCB1 ASStm (ACCB2 *AVIconHandlerOpenStmProc)
(AVIconBundle6 bundle, AVIconColorFormat colorFormat);
```

Description

Callback for [AVIconHandler](#). It opens a stream so that a drawing function can read the data contained in the icon set.

Parameters

bundle	The icon bundle for which the stream is opened.
colorFormat	The color format for the icon bundle.

Return Value

The stream object.

Header File

AVExpT.h

Related Methods

[AVAppCreateIconBundle6](#)

AVIconHandlerReleaseProc

```
ACCB1 void (ACCB2 *AVIconHandlerReleaseProc)
(AVIconBundle6 bundle);
```

Description

Callback for [AVIconHandler](#). It releases the icon object.

Parameters

bundle	The icon bundle that is released.

Return Value

None.

Header File

AVExpT.h

Related Methods

[AVAppCreateIconBundle6](#)



AIdleProc

```
ACCB1 void ACCB2 AIdleProc (void* clientData);
```

Description

Callback that is called periodically when the Acrobat viewer is otherwise idle.

Parameters

clientData	User-supplied data that was passed in the call to AVAppRegisterIdleProc .

Return Value

None

Header File

AVExpT.h

Related Methods

[AVAppRegisterIdleProc](#)
[AVAppUnregisterIdleProc](#)



AVMenuPredicate

```
ACCB1 ASBool ACCB2 AVMenuPredicate (AVMenu menu,  
void* clientData);
```

Description

Callback that is called for each menu enumerated by [AVMenubarAcquireMenuByPredicate](#). The first menu for which this callback returns **true** is acquired.

Parameters

menu	The current menu in the enumeration.
clientData	User-supplied data that was passed in the call to AVMenubarAcquireMenuByPredicate .

Return Value

true to acquire the current menu and halt enumeration, **false** to continue enumeration.

Header File

AVExpT.h

Related Methods

[AVMenubarAcquireMenuByPredicate](#)

AVMenuItemPredicate

```
ACCB1 ASBool ACCB2 AVMenuItemPredicate (AVMenuItem menuItem,  
void* clientData);
```

Description

Callback that is called for each menu item enumerated by [AVMenubarAcquireMenuItemByPredicate](#). The first menu item for which this callback returns **true** is acquired.

Parameters

menuItem	The current menu item in the enumeration.
clientData	User-supplied data that was passed in the call to AVMenubarAcquireMenuItemByPredicate .

Return Value

true to acquire the current menu item and halt enumeration, **false** to continue enumeration.

Header File

AVExpT.h

Related Methods

[AVMenubarAcquireMenuItemByPredicate](#)



AVNotifyTooltipProc

```
ACCB1 ASText ACCB2 AVNotifyTooltipProc (void* data);
```

Description

Callback that is called whenever the mouse hovers long enough to display the tooltip text. It returns text that is displayed in the tooltip.

Each toolbar button can have its own tooltip procedure, or they can be shared.

Parameters

data	User-supplied data that was passed in the call to AVToolButtonSetNotifyTooltipProc .

Return Value

A text object to be displayed in the tooltip.

Header File

AVExpT.h

Related Methods

[AVToolButtonSetNotifyTooltipProc](#)

AVOpenSaveDialogSettingsComputeEnabledProc

```
ACCB1 ASBool ACCB2 AVOpenSaveDialogSettingsComputeEnabledProc  
(AVFileFilterRec* currentFilter, void* data);
```

Description

A client can provides this optional callback if it wishes to control whether the settings button in the open or save dialog is enabled or disabled. If a user does not provide this callback function, then the state of the settings button, enabled or disabled, will be determined by whether the conversion handler has a settings proc or not. See below for an example of a **ConvertToPDFComputeEnabledProc**:

Parameters

currentFilter	The currently selected filter in the dialog.
data	void* of clientData , which is the last member of the AVOpenSaveDialogParamsRec structure.

Return Value

true if the Settings button should be enabled; **false** otherwise.

Header File

AVExpT.h

Related Methods

[AVOpenSaveDialogSettingsExecuteProc](#)

Example

```

typedef struct _t_FindConverterData
{
    AVFileFilterRec *filter;
    ASBool match;
} FindConverterData;

static ACCB1 ASBool ACCB2 EnumToPDFConverters(AVConversionToPDFHandler
handler, AVConversionEnumProcData data)
{
    FindConverterData * findData =
reinterpret_cast<FindConverterData*>(data);
    if (findData->filter == &handler->convFilter)
    {
        findData->match = true;
        return false;
    }
    else
        return true;
}

ACCB1 void ACCB2 ConvertToPDFComputeEnabledProc(AVFileFilterRec
*currentFilter, void *data)
{
    FindConverterData findData;
    findData.filter = currentFilter;
    findData.match = false;
    AVConversionEnumToPDFConverters(EnumToPDFConverters,
reinterpret_cast<AVConversionEnumProcData>(&findData));
    if (findData.match)
    {
        AVConversionToPDFHandler handler =
reinterpret_cast<AVConversionToPDFHandler>(currentFilter);
        if (handler && handler->settingsDialog)
            return true; // If we found a match and the conversion handler has
a settings dialog, enable the button
        else
            return false; // If we found a match but the conversion handler
does not have a settings dialog, disable the button
    }
    return false; // If this is not a conversion handler, disable
the settings button
}

```

AVOpenSaveDialogSettingsExecuteProc

ACCB1 [ASBool](#) ACCB2 AVOpenSaveDialogSettingsExecuteProc
([AVFileFilterRec](#)* currentFilter, void* data);

Description

A client provides this optional callback to decide what action is taken when the user clicks on the settings button. The function is called back with the currently selected filter. See below for an example of a **ConvertToPDFSettingsExecuteProc**.

Parameters

currentFilter	The currently selected filter in the dialog.
data	void* of clientData , which is the last member of the AVOpenSaveDialogParamsRec structure.

Return Value

Boolean

Header File

AVExpT.h

Related Methods

[AVOpenSaveDialogSettingsComputeEnabledProc](#)

Example

```
ACCB1 void ACCB2 ConvertToPDFSettingsExecuteProc (AVFileFilterRec
*currentFilter, void *data)
{
    FindConverterData findData;
    findData.filter = currentFilter;
    findData.match = false;
    AVConversionEnumToPDFConverters (EnumToPDFConverters,
reinterpret_cast<AVConversionEnumProcData>(&findData));
    if (findData.match)
    {
        AVConversionToPDFHandler handler =
reinterpret_cast<AVConversionToPDFHandler>(currentFilter);
        if (handler && handler->settingsDialog)
            handler->settingsDialog (AVConversionToPDFGetSetting handler->
settingsDialog (AVConversionToPDFGetSettings (handler), handler->
clientData);
    }
}
```

AVPageViewClickProc

```
ACCB1 ASBool ACCB2 AVPageViewClickProc (AVPageView pageView,
AVDevCoord x, AVDevCoord y, AVFlagBits16 flags,
AVTCount clickNo, void* data);
```

Description

User-supplied callback that is called whenever there is a mouse click in its **AVPageView**. This callback is registered using [AVAppRegisterForPageViewClicks](#).

NOTE: The numeric argument types have changed in Acrobat 6.0.

Parameters

pageView	The AVPageView in which the click occurred.
x	The click's x-coordinate.
y	The click's y-coordinate.
flags	Modifier keys that are held down while the mouse was clicked. They must be an OR of the Modifier Keys value.
clickNo	1 if single click, 2 if double-click, 3 if triple-click.
data	User-supplied data that was passed in the call to AVAppRegisterForPageViewClicks .

Return Value

true if the callback handled the mouse click, **false** if it does not and the click should be passed on to the next click handler.

Header File

AVExpT.h

Related Methods

[AVAppRegisterForPageViewClicks](#)
[AVAppUnregisterForPageViewClicks](#)

AVPageViewCursorProc

```
ACCB1 ASBool ACCB2 AVPageViewCursorProc (AVPageView pageView,  
AVDevCoord x, AVDevCoord y, void* data);
```

Description

User-supplied callback that is called whenever the cursor is moved. This callback is registered using [AVAppRegisterForPageViewAdjustCursor](#).

NOTE: The coordinate numeric types have changed in Acrobat 6.0.

Parameters

pageView	The AVPageView in which the cursor is located.
x	The cursor's x-coordinate.
y	The cursor's y-coordinate.
data	User-supplied data that was passed in the call to AVAppRegisterForPageViewAdjustCursor .

Return Value

true if the callback handled the cursor movement, **false** if it did not and the cursor handler should be allowed to.

Header File

AVExpT.h

Related Methods

[AVAppRegisterForPageViewAdjustCursor](#)
[AVAppUnregisterForPageViewAdjustCursor](#)

AVPageViewDrawProc

```
ACCB1 void ACCB2 AVPageViewDrawProc (AVPageView pageView,  
AVRect* updateRect, void* data);
```

Description

User-supplied callback that is called whenever the **AVPageView** is drawn. This callback is registered using [AVAppRegisterForPageViewDrawing](#).

NOTE: The numeric types have changed in Acrobat 6.0.

Parameters

pageView	The AVPageView to redraw.
updateRect	The rectangle enclosing the region to redraw.
data	User-supplied data that was passed in the call to AVAppRegisterForPageViewDrawing .

Return Value

None

Header File

AVExpT.h

Related Methods

[AVAppRegisterForPageViewDrawing](#)
[AVAppUnregisterForPageViewDrawing](#)

AVPageViewKeyDownProc

```
ACCB1 ASBool ACCB2 AVPageViewKeyDownProc (AVPageView pageView,  
AVKeyCode keyCode, AVFlagBits16 flags, void* data);
```

Description

Called whenever there is a key down in its **AVPageView**. This callback is registered using [AVAppRegisterForPageViewKeyDown](#).

NOTE: The numeric argument types have changed in Acrobat 6.0.

Parameters

pageView	The AVPageView in which the keystroke occurred.
keyCode	An ASCII code representing the key that was pressed. In some cases a different code results from a key combination (for example, a Control +key combination results in an ASCII code <= 31).
flags	Modifier keys that are held down while the key was pressed. They must be an OR of the Modifier Keys value.
data	User-supplied data that was passed in the call to AVAppRegisterForPageViewKeyDown .

Return Value

false to process the keydown event, **true** otherwise.

Header File

AVExpT.h

Related Methods

[AVAppRegisterForPageViewKeyDown](#)
[AVAppUnregisterForPageViewKeyDown](#)

AVRegisterCommandsProc

```
ACCB1 void ACCB2 AVRegisterCommandsProc (ASAtom handlerName) ;
```

Description

Callback for [AVCommandHandlerRec](#). The application maintains a global list of commands that the user can choose from when building a batch sequence. During the initialization sequence, all registered command handlers are asked to build and register all the commands that the handler wants included in the global command list. This is done by calling the command handler's **RegisterCommands** callback, if it's not **NULL**.

Parameters

handlerName	The name of the command handler.

Return Value

None

Header File

[AVExpT.h](#)

AVSelectionEnumProc

```
ACCB1 ASBool ACCB2 AVSelectionEnumProc (AVDoc doc,
void* clientData, void* aSelectedObject);
```

Description

User-supplied callback that is passed in the call to [AVDocEnumSelection](#). It is called once for each “item” in the selection.

[AVDocEnumSelection](#) calls the [AVDocSelectionEnumSelectionProc](#) for the current selection’s server to actually enumerate the selection.

Parameters

doc	The document whose selection is enumerated.
clientData	User-supplied data that was passed in the call to AVDocEnumSelection .
aSelectedObject	The selected “item” currently being enumerated. The format of the data is up to the selection server. See Selection Types for a list of the data formats for the Acrobat viewer’s built-in selection servers.

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

```
AVExpT.h
```

Related Methods

```
AVDocEnumSelection
```

AVSelectionPageRangeEnumProc

```
ACCB1 ASBool ACCB2 AVSelectionPageRangeEnumProc (AVDoc doc,  
void* clientData, PDPageNumber firstPage,  
PDPageNumber lastPage);
```

Description

User-supplied callback that is passed in the call to [AVDocSelectionEnumPageRanges](#). It is called once for each page in the selection, and consecutive pages are grouped into a single page range.

NOTE: The page number numeric type has changed in Acrobat 6.0.

Parameters

doc	The document whose selection is enumerated.
clientData	User-supplied data that was passed in the call to AVDocSelectionEnumPageRanges .
firstPage	The first page in a consecutive range of pages with a selection.
lastPage	The first page in a consecutive range of pages with a selection.

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

AVExpT.h

Related Methods

[AVDocEnumSelection](#)
[AVDocSelectionEnumPageRanges](#)



AVSetCursorProc

```
ACCB1 void ACCB2 AVSetCursorProc (CursHandle curs,
void* clientData);
```

Description

(Macintosh only) Callback in [ExternalDocWindowData](#) for opening PDF files in external windows. Called for a mouse-related event, such as mouse down or mouse movement.

Parameters

curs	Handle to cursor. It is a CursHandle .
clientData	User-supplied data that was passed in ExternalDocWindowData .

Return Value

None

Header File

AVExpT.h

Related Methods

[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromASFileWithParams](#)
[AVDocOpenFromPDDocWithParams](#)



AVSetFocusProc

```
ACCB1 void ACCB2 AVSetFocusProc (void* clientData);
```

Description

Callback in [ExternalDocServerCreationData](#) to return focus to the browser displaying the document.

Parameters

clientData	User-supplied data.

Return Value

None

Header File

AVExpT.h



AVSetMessageProc

```
ACCB1 void ACCB2 AVSetMessageProc (char* msg,
void* clientData);
```

Description

(Unused) Callback in [ExternalDocServerCreationData](#) for opening PDF files in external windows.

Parameters

doc	Message for external application.
clientData	User-supplied data that was passed in the call to AVSetMessageProc .

Return Value

None

Header File

AVExpT.h

Related Methods

[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromASFileWithParams](#)
[AVDocOpenFromPDDocWithParams](#)

AVSystemFontEnumProc

```
ACCB1 ASBool ACCB2 AVSystemFontEnumProc
(AVSystemFont systemFont, void* clientData);
```

Description

(Macintosh only). Callback for [AVAppEnumSystemFonts](#). It is called once for each system font.

Parameters

systemFont	The AVSystemFont currently being enumerated.
clientData	User-supplied data that was passed in the call to AVAppEnumSystemFonts .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

```
AVExpT.h
```

Related Methods

[AVAppEnumSystemFonts](#)

AVTextCopyProc

```
ACCB1 void ACCB2 AVTextCopyProc (ASAtom format, void* buf,
ASInt32 bufLen, void* clientData);
```

Description

Callback for [AVDocGetPageText](#). Text is passed to it in the specified format.

Parameters

format	Text format. See the description of the format parameter of AVDocGetPageText for a list of the allowed types.
buf	The text.
bufLen	Length of buf , in bytes.
clientData	User-supplied data that was passed in the call to AVDocGetPageText .

Return Value

None

Header File

AVExpT.h

Related Methods

[AVDocGetPageText](#)

AVToolButtonEnumProc

```
ACCB1 ASBool ACCB2 AVToolButtonEnumProc (AVToolButton button,
void* clientData);
```

Description

Callback for [AVToolBarEnumButtons](#). It is called once for each toolbar button.

Parameters

button	The toolbar button currently being enumerated.
clientData	User-supplied data that was passed in the call to AVToolBarEnumButtons .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

```
AVExpT.h
```

Related Methods

[AVToolBarEnumButtons](#)



AVToolDestroyProc

```
ACCB1 void ACCB2 AVToolDestroyProc (AVTool tool);
```

Description

An [AVToolRec](#) callback. If provided, called at shutdown time to allow the tool to free dynamic memory.

Parameters

<code>tool</code>	The tool to be destroyed.

Return Value

None

Header File

`AVExpT.h`



AVToolEnumProc

```
ACCB1 ASBool ACCB2 AVToolEnumProc (AVTool tool,  
void* clientData);
```

Description

Callback for [AVAppEnumTools](#). It is called once for each tool.

Parameters

tool	The tool currently being enumerated.
clientData	User-supplied data that was passed in the call to AVAppEnumTools .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

AVExpT.h

Related Methods

[AVAppEnumTools](#)



AVToolGetLabelProc

```
ACCB1 ASConstText ACCB2 AVToolGetLabelProc (AVTool tool);
```

Description

Callback for [AVToolRec](#). If provided, called when text describing the tool needs to be displayed.

Parameters

<code>tool</code>	The tool.

Return Value

The text object.

Header File

`AVExpT.h`

AVToolGetLabelIconProc

```
ACCB1 AVIcon ACCB2 AVToolGetLabelIconProc (AVTool tool);
```

Description

Callback for [AVToolRec](#). If provided, called when a static tool icon is needed. This icon should be similar to, but not the same as the tool button icon for this tool.

Parameters

<code>tool</code>	The tool.

Return Value

The icon object.

Header File

`AVExpT.h`

AVTransHandlerCompleteTransDictProc

```
ACCB1 void ACCB2 AVTransHandlerCompleteTransDictProc  
(AVTransHandler avth, const char* uiName, CosObj transDict);
```

Description

Callback for [AVTransHandler](#). This method is called after the user has selected a distinct transition. The transition handler must fill in any dictionary items necessary to create the effect specified by the **uiName** passed in. For example, if the “Wipe” transition handler is passed an **uiName** of “Wipe Left,” it would set the **Dir** key in **transDict** to the value 180.

[AVTransHandlerCompleteTransDictProc](#) should fill in standard information like direction, dimension, motion, and so forth—information gathered entirely from the UI name. Other specific information should be filled in by [AVTransHandlerDoPropertiesProc](#).

Parameters

avth	The transition handler.
uiName	User interface name of the transition.
transDict	Transition dictionary to set.

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVTransHandlerDoPropertiesProc](#)

AVTransHandlerDoPropertiesProc

```
ACCB1 void ACCB2 AVTransHandlerDoPropertiesProc  
(AVTransHandler avth, const char* uiName, PDTrans trans);
```

Description

Callback for [AVTransHandler](#). This method is called when the user clicks the button in the transition settings dialog. This allows the transition to bring up its own custom dialog allowing the user to further specify the desired transition effect.

Once the user selects a transition effect from the popup menu, the viewer immediately creates a transition (using [PDTransNewFromCosDoc](#) or [PDTransNew](#)) and calls [AVTransHandlerInitTransDictProc](#) and [AVTransHandlerCompleteTransDictProc](#). If the handler provides both an [AVTransHandlerDoPropertiesProc](#) and [AVTransHandlerGetButtonTextProc](#) callback, the dialog box displays a button. When the user clicks on the button, the viewer calls the handler's [AVTransHandlerDoPropertiesProc](#) callback. **DoProperties** is responsible for making any needed alterations to the transition; **InitTransDict** and **CompleteTransDict** are not called after **DoProperties**.

After the user clicks **OK** in the dialog box, **trans** is filled in using the supplied data.

Parameters

avth	The transition handler.
uiName	The user interface name for the transition handled by avth .
trans	The PDTrans to initialize.

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVTransHandlerCompleteTransDictProc](#)
[AVTransHandlerInitTransDictProc](#)

AVTransHandlerEnumProc

```
ACCB1 ASBool ACCB2 AVTransHandlerEnumProc  
(AVTransHandler avth, void* clientData);
```

Description

Callback for [AVAppEnumTransHandlers](#). It is called once for each transition handler.

Parameters

avth	The transition handler.
clientData	User-supplied data that was passed in the call to AVAppEnumTransHandlers .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

AVExpT.h

Related Methods

[AVAppEnumTransHandlers](#)

AVTransHandlerExecuteProc

```
ACCB1 void ACCB2 AVTransHandlerExecuteProc
(AVTransHandler avth, PDTrans trans, AVTransitionPort srcTP,
AVTransitionPort dstTP, ASFixed duration);
```

Description

Callback for [AVTransHandler](#). Executes the specified transition. The transition handler is responsible for copying the pixels specified by **srcTP** to the location specified by **dstTP**. In the process the handler can create any visual effect it desires, as long as the source pixels are eventually copied over the destination pixels in the end.

The handler should do its best to execute the visual effect in the number of seconds specified by **duration**.

The implementation will ensure that the source and destination rectangles are the same size, though their corners may not coincide.

Parameters

avth	The transition handler.
trans	The transition to execute.
srcTP	Source transition port.
dstTP	Destination transition port.
duration	Duration of the transition, in seconds.

Return Value

None

Header File

AVExpT.h

AVTransHandlerGetButtonTextProc

```
ACCB1 ASInt32 ACCB2 AVTransHandlerGetButtonTextProc  
(AVTransHandler avth, char* buffer, ASInt32 bufLen);
```

Description

Callback for [AVTransHandler](#). Gets a localized string that appears in the button on the transition settings dialog box. If [AVTransHandlerGetButtonTextProc](#) is **NULL** or the string it returns is empty, no button will appear.

Parameters

avth	The transition handler.
buffer	<i>(Filled by the callback)</i> The button text.
bufLen	Length of buffer , in bytes.

Return Value

The number of characters copied into **buffer**.

Header File

AVExpT.h

Related Callbacks

[AVTransHandlerDoPropertiesProc](#)

AVTransHandlerGetInstructionsProc

```
ACCB1 ASInt32 ACCB2 AVTransHandlerGetInstructionsProc  
(AVTransHandler avth, char* buffer, ASInt32 bufLen);
```

Description

(Unused) Callback for [AVTransHandler](#).

Parameters

avth	The transition handler.
buffer	(Filled by the callback) The instruction text.
bufLen	Length of buffer , in bytes.

Return Value

The number of characters copied into **buffer**.

Header File

AVExpT.h

Related Callbacks

[AVActionGetInstructionsProc](#)

AVTransHandlerGetStringOneTextProc

```
ACCB1 ASInt32 ACCB2 AVTransHandlerGetStringOneTextProc  
(AVTransHandler avth, char* buffer, ASInt32 bufLen);
```

Description

Callback for [AVTransHandler](#). Gets a localized string that appears above the button on the transition settings dialog box.

Parameters

avth	The transition handler.
buffer	<i>(Filled by the callback)</i> The string text appearing above the button.
bufLen	Length of buffer , in bytes.

Return Value

The number of characters copied into **buffer**.

Header File

AVExpT.h

Related Callbacks

[AVTransHandlerGetStringTwoTextProc](#)

AVTransHandlerGetStringTwoTextProc

```
ACCB1 ASInt32 ACCB2 AVTransHandlerGetStringTwoTextProc  
(AVTransHandler avth, char* buffer, ASInt32 bufLen);
```

Description

Callback for [AVTransHandler](#). Gets a localized string that appears below the button on the transition settings dialog box.

Parameters

avth	The transition handler.
buffer	<i>(Filled by the callback)</i> The string text appearing below the button.
bufLen	Length of buffer , in bytes.

Return Value

The number of characters copied into **buffer**.

Header File

AVExpT.h

Related Callbacks

[AVTransHandlerGetStringOneTextProc](#)

AVTransHandlerGetTypeProc

```
ACCB1 ASAtom ACCB2 AVTransHandlerGetTypeProc  
(AVTransHandler avth);
```

Description

Callback for [AVTransHandler](#). Gets the transition type serviced by this handler. The handler for a given transition is found by comparing the result of [PDTransGetSubtype](#) to the value returned by the registered transition handler's [AVTransHandlerGetTypeProc](#) callbacks.

Parameters

avth	The transition handler.

Return Value

Type of transition handler, which may be one of the types provided in the Acrobat viewer or a new type registered by a plug-in.

Header File

AVExpT.h

Related Methods

[PDTransGetSubtype](#)

AVTransHandlerGetItemUINameProc

```
ACCB1 ASInt32 ACCB2 AVTransHandlerGetItemUINameProc  
(AVTransHandler avth, ASInt32 item, char* buffer,  
ASInt32 bufLen);
```

Description

Callback for [AVTransHandler](#).

A transition handler can handle several distinct transitions. For example, the “Wipe” transition handler can create four distinct effects: wipe left, wipe right, wipe up, and wipe down.

The transition setting dialog box should create a separate user interface entry for each distinct transition. It determines both the number and names of the distinct transition types by repeatedly calling each transition handler’s

[AVTransHandlerGetItemUINameProc](#) callback, starting with an item number of 0 and increasing until [AVTransHandlerGetItemUINameProc](#) returns an empty string.

Thus when the transaction handler is selected from the list, this callback is called. The transition handler should fill in the **Type** and **S** fields.

[AVTransHandlerGetItemUINameProc](#) should fill in any default values. This information is passed into the [AVTransHandlerDoPropertiesProc](#) in the form of a **PDTrans** if that callback exists.

Parameters

avth	The transition handler.
item	The item number.
buffer	<i>(Filled by the callback)</i> The name of the transition in the user interface. This string should be localized.
bufLen	Length of buffer , in bytes.

Return Value

The number of characters copied into **buffer**.

Header File

AVExpT.h

Related Callbacks

[AVTransHandlerGetUINameProc](#)

AVTransHandlerGetUINameProc

```
ACCB1 ASInt32 ACCB2 AVTransHandlerGetUINameProc  
(AVTransHandler avth, PDTrans trans, char* buffer,  
ASInt32 bufLen);
```

Description

Callback for [AVTransHandler](#). Retrieves the user-interface name for an existing **PDTrans**. For example, if the transition type is “Wipe” and the direction is 180, **AVTransHandlerGetUINameProc** would return “Wipe Left,” localized.

A transition handler can handle several distinct transitions. For example, the “Wipe” transition handler can create four distinct effects: wipe left, wipe right, wipe up, and wipe down.

The transition setting dialog box creates a separate user-interface entry for each distinct transition. It determines both the number and names of the distinct transition types by repeatedly calling each transition handler’s [AVTransHandlerGetUINameProc](#) callback, starting with an item number of 0 and increasing until the [AVTransHandlerGetUINameProc](#) callback returns an empty string.

The string returned by [AVTransHandlerGetUINameProc](#) should be localized.

The **AVTransHandlerGetUINameProc** is used to enumerate the entire list of supported transition effects that the handler wishes to display in the popup menu (for example, “Wipe Left” for item == 0, “Wipe Right” for item == 1, and so on).

Parameters

avth	The transition handler.
trans	The transition whose name is obtained.
buffer	<i>(Filled by the callback)</i> The user-interface name of the transition.
bufLen	Length of buffer , in bytes.

Return Value

The number of characters copied into **buffer**.

Header File

AVExpT.h

Related Callbacks

[AVTransHandlerGetItemUINameProc](#)

AVTransHandlerInitTransDictProc

```
ACCB1 void ACCB2 AVTransHandlerInitTransDictProc
(AVTransHandler avth, CosObj transDict);
```

Description

Callback for [AVTransHandler](#). This method should set default values in the transition dictionary, **transDict**.

As soon as the handler is selected from the list, [AVTransHandlerInitTransDictProc](#) is called. This function should fill in the **Type** and **S** fields of **transDict**. [AVTransHandlerInitTransDictProc](#) should also fill in any default values. This information is passed to [AVTransHandlerDoPropertiesProc](#) in the form of a [PDTrans](#) if [AVTransHandlerDoPropertiesProc](#) exists.

Normally the **Type** and **S** fields are filled in when the transition is created via [PDTransNewFromCosDoc](#). The implementation then calls [AVTransHandlerInitTransDictProc](#) and [AVTransHandlerCompleteTransDictProc](#) immediately on the newly created [PDTrans](#).

Parameters

avth	The transition handler.
transDict	Transition dictionary to set.

Return Value

None

Header File

[AVExpT.h](#)

Related Callbacks

[AVTransHandlerCompleteTransDictProc](#)

Related Methods

[PDTransNewFromCosDoc](#)

AVUndoBeginEndProc

```
ACCB1 void ACCB2 AVUndoBeginEndProc (AVDoc doc, ASBool bUndo);
```

Description

Callback type for [AVUndoHandler](#). The **beginUndoRedo** procedure is called when the user initiates an **Undo** or **Redo** command and the [AVUndoVerifyProc](#) returns **true**. The **endUndoRedo** procedure is called when execution of the operation is complete. Use callbacks of this type to notify the handler that an undo or redo operation is beginning or ending.

These callbacks are optional. They can be used to allocate and deallocate memory for the operations, for example, or, when grouping undo records, to suspend UI updates during the operation.

Parameters

doc	The document containing the undo record.
bUndo	true if the user initiated an undo operation, false if it is a redo operation.

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVUndoGetTitleProc](#)

[AVUndoExecuteProc](#)

[AVUndoVerifyProc](#)

AVUndoGetTitleProc

```
ACCB1 void ACCB2 AVUndoGetTitleProc (AVUndo undo,
    AText title);
```

Description

Callback for [AVUndoHandler](#). Called when the user initiates an **Undo** or **Redo** command and the [AVUndoVerifyProc](#) returns **true**. Use this to return the UI title string for the undo record.

Parameters

undo	The undo record.
title	<i>(Filled by the method)</i> The UI title string for the undo record, as a text object.

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVUndoVerifyProc](#)

AVUndoExecuteProc

ACCB1 [ASBool](#) ACCB2 AVUndoExecuteProc ([AVUndo](#) undo);

Description

Callback for [AVUndoHandler](#). Called when the user initiates an **Undo** or **Redo** command and the [AVUndoVerifyProc](#) returns **true**. Use this to perform the requested operation.

Parameters

<code>undo</code>	The undo record

Return Value

true if the requested operation is performed successfully, **false** otherwise.

Header File

`AVExpT.h`

Related Callbacks

[AVUndoVerifyProc](#)



AVUndoReleaseProc

```
ACCB1 void ACCB2 AVUndoReleaseProc (AVUndo undo) ;
```

Description

Callback for [AVUndoHandler](#). Called when the undo object is no longer needed. Use this to free any dynamic data that was associated with the object by the handler.

Parameters

<code>undo</code>	The undo record

Return Value

None

Header File

AVExpT.h

Related Methods

[AVDocClearUndos](#)



AVUndoVerifyProc

```
ACCB1 ASBool ACCB2 AVUndoVerifyProc (AVUndo undo);
```

Description

Callback for [AVUndoHandler](#). Called when the user initiates an **Undo** or **Redo** command. Use this to verify that the undo record is still valid and the operation can be performed.

Parameters

<code>undo</code>	The undo record

Return Value

true if the operation can be performed, **false** otherwise.

Header File

`AVExpT.h`

Related Callbacks

[AVUndoExecuteProc](#)

AVWindowAdjustCursorProc

```
ACCB1 void ACCB2 AVWindowAdjustCursorProc (AVWindow win,  
AVWindowCoord x, AVWindowCoord y);
```

Description

Callback for [AVWindowHandler](#). Called periodically while the cursor is over the **AVWindow** (if the window is active). Use this to adjust the cursor's appearance.

NOTE: The coordinate numeric types have changed in Acrobat 6.0.

Parameters

win	The window containing the cursor.
x	The cursor's x-coordinate.
y	The cursor's y-coordinate.

Return Value

None

Header File

AVExpT.h

AVWindowCanPerformEditOpProc

```
ACCB1 ASBool ACCB2 AVWindowCanPerformEditOpProc (AVWindow win,  
ASAtom editOp);
```

Description

Callback for [AVWindowHandler](#). Called before showing the **Edit** menu, to determine whether to enable the **Edit** menu item corresponding to the given **ASAtom**.

Parameters

win	The current window.
editOp	ASAtom specifying the edit operation. Must be an ASAtom corresponding to one the strings: Cut , Copy , Paste , Clear , SelectAll , and Undo .

Return Value

true if the specified operation can be performed, **false** otherwise.

Header File

AVExpT.h

Related Callbacks

[AVWindowPerformEditOpProc](#)

AVWindowDestroyPlatformThingProc

```
ACCB1 void ACCB2 AVWindowDestroyPlatformThingProc  
(AVWindow win, void* platformThing);
```

Description

Callback for [AVWindowHandler](#). Called when it's time to dispose of the `platformThing` for the window passed to [AVWindowNewFromPlatformThing](#).

Parameters

win	The window.
platformThing	The platform-specific object (WindowPtr in Mac OS, an HWND in Windows, and a Widget in UNIX) that was used for this window.

Return Value

None

Header File

AVExpT.h

Related Methods

[AVWindowNewFromPlatformThing](#)

AVWindowDidActivateProc

```
ACCB1 void ACCB2 AVWindowDidActivateProc (AVWindow win);
```

Description

Callback for [AVWindowHandler](#). Called after the window has been activated. The window being activated will not always become the front-most window.

Parameters

win	The window being activated.

Return Value

None

Header File

`AVExpT.h`

Related Callbacks

[AVWindowWillDeactivateProc](#)

AVWindowDidBecomeKeyProc

```
ACCB1 void ACCB2 AVWindowDidBecomeKeyProc (AVWindow win);
```

Description

Callback for [AVWindowHandler](#). Called after the window becomes the key window.

Parameters

win	The window becoming the key window.

Return Value

None

Header File

`AVExpT.h`

Related Callbacks

[AVWindowWillResignKeyProc](#)

Related Methods

[AVWindowSetWantsKey](#)

AVWindowDidCloseProc

```
ACCB1 void ACCB2 AVWindowDidCloseProc (AVWindow win);
```

Description

Callback for [AVWindowHandler](#). Called immediately after the window has been closed, but before it has been freed. You may want to explicitly destroy the window in this routine. See also [AVWindowWillCloseProc](#).

Parameters

win	The window that was closed.

Return Value

None

Header File

`AVExpT.h`

Related Callbacks

[AVWindowWillCloseProc](#)

Related Methods

[AVWindowUserClose](#)



AVWindowDidResizeProc

```
ACCB1 void ACCB2 AVWindowDidResizeProc (AVWindow win,  
const AVScreenRect* newFrame);
```

Description

Callback for [AVWindowHandler](#). Called after the window has been resized.

Parameters

win	The window that was resized.
AVRect	Rectangle specifying the window's new size and location.

Return Value

None

Header File

AVExpT.h

AVWindowDrawProc

```
ACCB1 void ACCB2 AVWindowDrawProc (AVWindow win,  
const AVWindowRect* rect);
```

Description

Callback for [AVWindowHandler](#). Called whenever the window needs to refresh some part of its interior. It should redraw the contents of the specified rectangle.

Parameters

win	The window whose content is redrawn.
rect	The region of win to redraw.

Return Value

None

Header File

AVExpT.h

Related Methods

[AVWindowDrawNow](#)

AVWindowKeyDownProc

```
ACCB1 void ACCB2 AVWindowKeyDownProc (AVWindow win, char key,
void* platformEvent);
```

Description

Callback for [AVWindowHandler](#). Called to handle keystrokes when this is the key window.

Parameters

win	The window.
key	The key that was pressed.
platformEvent	Platform-specific event record: Macintosh — Pointer to an EventRecord . UNIX — eventPtr .

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVWindowMouseDownProc](#)

AVWindowMouseDownProc

```
ACCB1 void ACCB2 AVWindowMouseDownProc (AVWindow win,
AVWindowCoord x, AVWindowCoord y, void* platformEvent);
```

Description

Callback for [AVWindowHandler](#). Mouse clicks in the **AVWindow** are dispatched through this callback.

NOTE: The coordinate numeric types have changed in Acrobat 6.0.

Parameters

win	The window in which the click occurred.
x	The click's x-coordinate.
y	The click's y-coordinate.
platformEvent	Platform-specific event record: Macintosh — Pointer to an EventRecord . UNIX — eventPtr .

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVWindowKeyDownProc](#)

AVWindowPerformEditOpProc

```
ACCB1 void ACCB2 AVWindowPerformEditOpProc (AVWindow win,  
ASAtom editOp);
```

Description

Callback for [AVWindowHandler](#). Called when the user has chosen an **Edit** menu item, if the corresponding [AVWindowCanPerformEditOpProc](#) returned **true**.

Parameters

win	The window that the edit menu item is to act on.
editOp	An ASAtom specifying the edit operation. Must be an ASAtom corresponding to one the strings: Cut , Copy , Paste , Clear , SelectAll , and Undo .

Return Value

None

Header File

[AVExpT.h](#)

Related Callbacks

[AVWindowCanPerformEditOpProc](#)

AVWindowWillBeResizedProc

```
ACCB1 ASBool ACCB2 AVWindowWillBeResizedProc (AVWindow win,  
AVScreenRect* newFrame);
```

Description

Callback for [AVWindowHandler](#). Called when the window is about to resize.

Parameters

win	The window to resize.
newFrame	Rectangle specifying the size to which the window will be resized. This callback may change the new frame size.

Return Value

true to permit the resizing, **false** to abort it.

Header File

`AVExpT.h`

Related Methods

[AVWindowSetFrame](#)

AVWindowWillCloseProc

```
ACCB1 ASBool ACCB2 AVWindowWillCloseProc (AVWindow win,  
ASBool quitting);
```

Description

Callback for [AVWindowHandler](#). The window is about to close.

Parameters

win	The window to close.
quitting	If true , the application is trying to quit.

Return Value

true if the window should close, **false** to abort the operation.

Header File

AVExpT.h

Related Methods

[AVWindowUserClose](#)

AVWindowWillDeactivateProc

```
ACCB1 void ACCB2 AVWindowWillDeactivateProc (AVWindow win);
```

Description

Callback for [AVWindowHandler](#). Called before the window becomes deactivated or hidden.

Parameters

win	The window that will be deactivated.

Return Value

None

Header File

`AVExpT.h`

Related Callbacks

[AVWindowDidActivateProc](#)



AVWindowWillResignKeyProc

```
ACCB1 void ACCB2 AVWindowWillResignKeyProc (AVWindow win);
```

Description

Callback for [AVWindowHandler](#). Called before the window ceases to be the key window.

Parameters

win	The window to resign key status.

Return Value

None

Header File

AVExpT.h

Related Callbacks

[AVWindowDidBecomeKeyProc](#)



CancelProc

```
ASBool CancelProc (void* clientData);
```

Description

This call has been replaced by [ASCancelProc](#).

Callback to check for canceling operations. A **CancelProc** is typically passed to some method that takes a long time to complete. At frequent intervals, the method calls the **CancelProc**. If it returns **true**, then the method cancels its operation; if **false**, it continues.

Parameters

clientData	User-supplied data that was passed to the CancelProc .

Return Value

true if the processing is canceled, **false** otherwise.

Header File

ASExpT.h

Related Callbacks

PDFLPrintCancelProc (Only available with PDF Library SDK)

Related Methods

[AVAppGetCancelProc](#)

CosDocEnumEOFsProc

```
ACCB1 ASBool ACCB2 CosDocEnumEOFsProc (CosDoc cosDoc,
ASFileOffset fileOffset, void* clientData);
```

Description

Callback for [CosDocEnumEOFs](#). Called once for each EOF in a file.

Parameters

cosDoc	The CosDoc in which the EOF is found.
fileOffset	The offset into the file directly following the %%EOF keyword. If the procedure is called more than once, the file positions passed to it are in decreasing order (that is, the EOF positions are treated as “rollback points”). NOTE: The precise value passed to the procedure is not defined. It is at least one byte past the %%EOF keyword, but may include one or more white space characters. When the procedure is called only once, there is no guarantee that fileOffset is the same as the length of the file.
clientData	User-supplied data that was passed in the call to CosDocEnumEOFs .

Return Value

true to continue enumeration, **false** to halt the enumeration.

Header File

CosExpT.h

Related Methods

[CosDocEnumEOFs](#)

CosObjEnumProc

```
ACCB1 ASBool ACCB2 CosObjEnumProc (CosObj obj, CosObj value,
void* clientData);
```

Description

Callback for [CosObjEnum](#), [CosDocEnumIndirect](#), and [PDDocEnumOCGs](#). Called once for each component of a composite Cos object (dictionary, array, and stream).

Parameters

obj	Dictionary — Key. Array — Array element. Stream — The stream's dictionary (the whole thing, not one key at a time).
value	Dictionary — The value associated with the Key. Array — a null Cos object. Stream — a null Cos object. For CosDocEnumIndirect and PDDocEnumOCGs , this is always the NULL Cos object.
clientData	User-supplied data that was passed in the call to CosObjEnum , CosDocEnumIndirect , or PDDocEnumOCGs .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

CosExpT.h

Related Methods

[CosObjEnum](#)
[CosDocEnumIndirect](#)
[PDDocEnumOCGs](#)

CosObjOffsetProc

```
ACCB1 void ACCB2 CosObjOffsetProc (CosObj obj,  
ASFilePos fileOffset, ASArraySize length, void* clientData);
```

Description

Callback for [PDDocSaveParams](#) used by [PDDocSaveWithParams](#). Use this to get information about Cos objects of interest while a **PDDoc** is saved.

Parameters

obj	The CosObj found.
fileOffset	The offset of obj into the PDF file.
length	length of obj .
clientData	Pointer to user-supplied data passed in the offsetProcClientData parameter of the PDDocSaveParams structure.

Return Value

None

Header File

CosExpT.h

Related Methods

[PDDocSaveWithParams](#)

CosObjSetCallbackFlagProc

```
ACCB1 ASBool ACCB2 CosObjSetCallbackFlagProc (CosObj obj,  
ASBool set);
```

Description

Callback in [PDDocPreSaveInfo](#), which is used by the [PDDocPreSaveProc](#) callback. Use this callback to set a flag in each [CosObj](#) that you care about, so that you will be called back during the [PDDoc](#)'s save and given the Cos object's offset and length. After a PDF file is saved, the Cos objects previously obtained are no longer valid.

Parameters

obj	The CosObj marked.
set	true to set the flag to be called back during the save, false otherwise.

Return Value

None

Header File

[CosExpT.h](#)

Related Methods

[PDDocSaveWithParams](#)



CrossDocLinkProc

```
ACCB1 AVDoc ACCB2 CrossDocLinkProc (ASPathName path,  
ASFileSys fileSys, AVDocViewDef viewDef, AVDoc srcDoc,  
void* data);
```

Description

Callback in [ExternalDocServerCreationData](#). Called when a cross-document link is clicked in an [AVDoc](#) in an external application's window.

Parameters

path	Path to document to which the link points.
fileSys	The ASFileSys with which to open document.
viewDef	The AVDocViewDef with which to open document.
srcDoc	The AVDoc that contains the cross-document link.
data	User-supplied data that was passed in the call to CrossDocLinkProc .

Return Value

The [AVDoc](#) for the new document.

Header File

[AvExpT.h](#)

Related Methods

[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromASFileWithParams](#)
[AVDocOpenFromPDDocWithParams](#)

CrossDocLinkWithDestProc

```
ACCB1 AVDoc ACCB2 CrossDocLinkWithDestProc (ASPathName path,  
ASFileSys fileSys, AVDocViewDef viewDef, AVDestInfo destInfo,  
AVDoc srcDoc, void* data);
```

Description

Callback in [ExternalDocServerCreationData](#). Called when a cross-document link is clicked in an [AVDoc](#) in an external application's window.

Parameters

path	Path to document to which the link points.
fileSys	The ASFileSys with which to open document.
viewDef	The AVDocViewDef with which to open document.
destInfo	A destination in a PDF document.
srcDoc	The AVDoc that contains the cross-document link.
data	User-supplied data that was passed in the call to CrossDocLinkProc .

Return Value

The [AVDoc](#) for the new document.

Header File

[AvExpT.h](#)

Related Methods

[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromASFileWithParams](#)
[AVDocOpenFromPDDocWithParams](#)



DeactivateProcType

```
ACCB1 void ACCB2 DeactivateProcType (AVTool tool);
```

Description

Callback for [AVTool](#). Called when the tool will no longer be the active tool.

Parameters

<code>tool</code>	The tool to deactivate.

Return Value

None

Header File

`AVExpT.h`

Related Callbacks

[ActivateProcType](#)

Related Methods

[AVAppSetActiveTool](#)

DoClickProcType

```
ACCB1 ASBool ACCB2 DoClickProcType (AVTool tool,  
AVPageView pageView, AVDevCoord xHit, AVDevCoord yHit,  
AVFlagBits16 flags, AVTCount clickNo);
```

Description

Callback for [AVTool](#). Handles mouse clicks when the tool is active. For Mac OS, this handles button or option-button mouse clicks. For Windows, this handles right or left button mouse clicks.

NOTE: The numeric argument types have changed in Acrobat 6.0.

Parameters

tool	The tool.
pageView	The AVPageView in which the click occurred.
xHit	The click's x-coordinate.
yHit	The click's y-coordinate.
flags	Modifier keys that were held down while clicking. Must be an OR of the Modifier Keys values.
clickNo	1 if single click, 2 if double click, 3 if triple click.

Return Value

true if the callback handled the click, **false** if it did not and the click should be passed to the next click handling procedure.

Header File

AVExpT.h

Related Callbacks

[DoKeyDownProcType](#)



DoLeaveProcType

```
ACCB1 void ACCB2 DoLeaveProcType (AVTool tool,  
AVPageView pageView) ;
```

Description

Callback for [AVTool](#). Called when the tool leaves the page view, that is, when the cursor is moved out of the page view.

Parameters

tool	The tool.
pageView	The AVPageView that the tool left.

Return Value

None

Header File

AVExpT.h

DoKeyDownProcType

```
ACCB1 ASBool ACCB2 DoKeyDownProcType (AVTool tool,  
AVKeyCode key, AVFlagBits16 flags);
```

Description

Callback for [AVTool](#). Handles key presses when the tool is active.

NOTE: The **key** and **flags** numeric types have changed in Acrobat 6.0.

Parameters

tool	The tool.
key	The key that was pressed.
flags	Modifier keys that were also pressed. Must be an OR of the Modifier Keys values.

Return Value

true if the callback handled the key press, **false** if it did not and the key press should be passed to the key press handling procedure.

Header File

`AVExpT.h`

Related Callbacks

[DoClickProcType](#)

GetSelectionServerProcType

```
ACCB1 AVDocSelectionServer ACCB2 GetSelectionServerProcType  
(AVTool tool, AVDoc doc);
```

Description

Callback for **AVTool**. Gets the selection server associated with the tool, if any.

Parameters

tool	The tool.
doc	The active AVDoc .

Return Value

The selection server associated with this tool. Returns **NULL** if the tool has no selection server.

Header File

AVExpT.h



GetTypeProcType

```
ACCB1 ASAtom ACCB2 GetTypeProcType (AVTool tool);
```

Description

Callback for [AVTool](#). Returns the tool's name.

Parameters

<code>tool</code>	The tool.

Return Value

The [ASAtom](#) corresponding to the tool's name.

Header File

`AVExpT.h`

Related Methods

[AVAppGetToolByName](#)



HFTServerDestroyProc

```
ACCB1 void ACCB2 HFTServerDestroyProc (HFTServer hftServer,  
void* clientData);
```

Description

Callback for an **HFT** server. Destroys the specified **HFT** (for example, by calling [HFTServerDestroy](#)).

Parameters

hftServer	The HFT server associated with this destroy proc.
clientData	User-supplied data that was passed in the call to HFTServerNew .

Return Value

None

Header File

ASExpT.h

Related Methods

[HFTServerNew](#)
[HFTDestroy](#)

HFTServerProvideHFTProc

```
ACCB1 HFT ACCB2 HFTServerProvideHFTProc (HFTServer hftServer,  
ASVersion version, void* clientData);
```

Description

Callback for an **HFT** server. Returns an **HFT** with the specified version number. If the **HFT** has not yet been created, create and return it. If the **HFT** already exists, do not create a new copy of it; simply return the existing copy.

NOTE: The **version** numeric type has changed in Acrobat 6.0.

Parameters

hftServer	The HFT server associated with this proc.
version	The HFT version being requested.
clientData	User-supplied data passed in the call to HFTServerNew .

Return Value

The requested version of the **HFT**.

Header File

`ASExpT.h`

Related Methods

[HFTServerNew](#)



IsPersistentProcType

```
ACCB1 ASBool ACCB2 IsPersistentProcType (AVTool tool);
```

Description

Callback for [AVTool](#). Indicates whether the tool would like to stay active after it has been used, or is a one-shot tool.

The Acrobat viewer does not contain any code to enforce a tool's request to be persistent; it is up to each tool to be a good citizen. For example, if a tool is not persistent, after that tool is used once it should get the previously active tool (using [AVAppGetLastActiveTool](#)) and check whether that tool wishes to be persistent (using [AVToolIsPersistent](#)). If so, set the active tool to that tool. If not, set the active tool to the default tool (obtained using [AVAppGetDefaultTool](#)).

Parameters

<code>tool</code>	The tool.

Return Value

true if the tool wishes to be persistent, **false** otherwise.

Header File

`AVExpT.h`

Related Methods

[AVToolIsPersistent](#)

PDActionHandlerCanCopyProc

```
ACCB1 ASBool ACCB2 PDActionHandlerCanCopyProc  
(PDActionHandler pdah, PDAction action);
```

Description

(Optional) Callback for [PDActionHandler](#). Returns **true** if the copy operation is expected to succeed. Tests, for example, whether copying is allowed by document permissions.

NOTE: The handler is not expected to test other actions (if any) in the action chain.

Parameters

pdah	The action handler.
action	The action object.

Return Value

true if the action can be copied, **false** if it cannot.

Header File

`PDExpT.h`

Related Callbacks

[PDActionHandlerCanPasteProc](#)
[PDActionHandlerCopyProc](#)
[PDActionHandlerDestroyDataProc](#)
[PDActionHandlerPasteProc](#)

Related Methods

[PDActionCanCopy](#)
[PDRegisterActionHandler](#)

PDActionHandlerCanPasteProc

```
ACCB1 ASBool ACCB2 PDActionHandlerCanPasteProc  
(PDActionHandler pdah, PDDoc doc, PDActionHandlerData data);
```

Description

(Optional) Callback for [PDActionHandler](#). Returns **true** if the paste operation is expected to succeed. Tests, for example, whether pasting is allowed by document permissions.

NOTE: The handler is not expect to test other actions (if any) in the action chain.

Parameters

pdah	The action handler.
doc	The destination document.
data	The action data to test.

Return Value

true if the action can be pasted to the document, **false** if it cannot.

Header File

[PDExpT.h](#)

Related Callbacks

[PDActionHandlerCanCopyProc](#)
[PDActionHandlerCopyProc](#)
[PDActionHandlerDestroyDataProc](#)
[PDActionHandlerPasteProc](#)

Related Methods

[PDActionCanPaste](#)
[PDRegisterActionHandler](#)

PDActionHandlerCopyProc

```
ACCB1 PDActionHandlerData ACCB2 PDActionHandlerCopyProc  
(PDActionHandler pdah, PDAction action);
```

Description

(Optional) Callback for [PDActionHandler](#). Copies data from an action object to a new data structure, from which it can be pasted to a new document. The [PDActionHandlerData](#) does not store any information related to a Next action. Rebuilding the action chain is the responsibility of the caller and can be ignored by the [PDActionHandler](#).

Parameters

<code>pdah</code>	The action handler.
<code>action</code>	The action object.

Return Value

The new structure containing the action data.

Header File

`PDExpT.h`

Related Callbacks

[PDActionHandlerCanCopyProc](#)
[PDActionHandlerCanPasteProc](#)
[PDActionHandlerDestroyDataProc](#)
[PDActionHandlerPasteProc](#)

Related Methods

[PDActionCopy](#)
[PDRegisterActionHandler](#)

PDActionHandlerDestroyProc

```
ACCB1 void ACCB2 PDActionHandlerDestroyProc  
(PDActionHandler pdah) ;
```

Description

(Optional) Callback for [PDActionHandler](#). Destroys the action handler structure when the application no longer requires it. The handler should destroy any dynamic memory stored in the **userData** field.

Parameters

pdah	The action handler to destroy.

Return Value

None

Header File

PDExpT.h

Related Methods

[PDRegisterActionHandler](#)

PDActionHandlerDestroyDataProc

```
ACCB1 void ACCB2 PDActionHandlerDestroyDataProc  
(PDActionHandler pdah, PDActionHandlerData data);
```

Description

Callback for [PDActionHandler](#). Destroys data copied into an action clipboard data structure after it has been successfully pasted to a new document.

Parameters

pdah	The action handler.
data	The action data to destroy.

Return Value

None

Header File

PDExpT.h

Related Callbacks

[PDActionHandlerCanCopyProc](#)
[PDActionHandlerCanPasteProc](#)
[PDActionHandlerCopyProc](#)
[PDActionHandlerPasteProc](#)

Related Methods

[PDActionPaste](#)
[PDRegisterActionHandler](#)

PDActionHandlerGetTypeProc

```
ACCB1 ASAtom ACCB2 PDActionHandlerGetTypeProc  
(PDActionHandler pdah) ;
```

Description

Callback for [PDActionHandler](#). Returns an [ASAtom](#) indicating the action type for which the handler is responsible. Types are defined by the client when registering the action handler.

Parameters

pdah	The action handler whose type is returned.

Return Value

The action type for which this handler is responsible.

Header File

`PDExpT.h`

Related Methods

[PDRegisterActionHandler](#)

PDActionHandlerPasteProc

```
ACCB1 PDAction ACCB2 PDActionHandlerPasteProc  
(PDActionHandler pdah, PDDoc doc, PDActionHandlerData data);
```

Description

(Optional) Callback for [PDActionHandler](#). Creates a new [PDAction](#) in the destination document using data that was placed in a [PDActionClipboardData](#) structure using the [PDActionCopy](#) function, and returns the new action object.

Parameters

pdah	The action handler.
doc	The destination document.
data	The action data to paste.

Return Value

The new action object containing the pasted data and associated with the document.

Header File

PDExpT.h

Related Callbacks

[PDActionHandlerCanCopyProc](#)
[PDActionHandlerCanPasteProc](#)
[PDActionHandlerCopyProc](#)
[PDActionHandlerDestroyDataProc](#)

Related Methods

[PDActionPaste](#)
[PDRegisterActionHandler](#)

PDAnnotHandlerCanCopyProc

```
ACCB1 ASBool ACCB2 PDAnnotHandlerCanCopyProc  
(PDAnnotHandler pdanh, PDAnnot pdan);
```

Description

(Optional) Callback for [PDAnnotHandler](#). Returns **true** if the copy operation is expected to succeed. Tests, for example, whether copying is allowed by document permissions.

Parameters

pdanh	The annotation handler responsible for this annotation.
pdan	The annotation object.

Return Value

true if the annotation can be copied, **false** if it cannot.

Header File

`PDExpT.h`

Related Callbacks

[PDAnnotHandlerCanPasteProc](#)
[PDAnnotHandlerCopyProc](#)
[PDAnnotHandlerPasteProc](#)
[PDAnnotHandlerDestroyDataProc](#)

Related Methods

[PDAnnotCanCopy](#)
[PDRegisterAnnotHandler](#)

PDAnnotHandlerCanPasteProc

```
ACCB1 ASBool ACCB2 PDAnnotHandlerCanPasteProc  
(PDAnnotHandler pdanh, PDPage destPage,  
const ASFixedPoint *center, PDAnnotClipboardData data);
```

Description

(Optional) Callback for [PDAnnotHandler](#). Returns **true** if the paste operation is expected to succeed. Tests whether data from an annotation that has been copied to a clipboard can be pasted to a location on a page. Pasting can be disallowed by document permissions, or because the annotation cannot be accurately reproduced in the destination document.

Parameters

pdanh	The annotation handler responsible for this annotation.
destPage	The page to which the annotation would be pasted.
center	The location for the center of the annotation on the destination page, or a NULL pointer to center the annotation on the destination page's crop box.
data	The copied annotation data to test.

Return Value

true if the annotation data can be pasted, **false** otherwise.

Header File

`PDExpT.h`

Related Callbacks

[PDAnnotHandlerCanCopyProc](#)
[PDAnnotHandlerCopyProc](#)
[PDAnnotHandlerPasteProc](#)
[PDAnnotHandlerDestroyDataProc](#)

Related Methods

[PDAnnotCanPaste](#)
[PDRegisterAnnotHandler](#)

PDAnnotHandlerCopyProc

```
ACCB1 PDAnnotClipboardData ACCB2 PDAnnotHandlerCopyProc  
(PDAnnotHandler pdanh, PDPage sourcePage, PDAnnot pdan);
```

Description

(Optional) Callback for [PDAnnotHandler](#). Copies data from the annotation object to a new clipboard structure, and returns the clipboard structure.

Parameters

pdanh	The annotation handler responsible for this annotation.
sourcePage	The page containing the annotation to be copied.
pdan	The annotation object.

Return Value

The clipboard structure containing the copied data.

Header File

PDExpT.h

Related Callbacks

[PDAnnotHandlerCanCopyProc](#)
[PDAnnotHandlerCanPasteProc](#)
[PDAnnotHandlerCopyProc](#)
[PDAnnotHandlerDestroyDataProc](#)
[PDAnnotHandlerPasteProc](#)

Related Methods

[PDAnnotCopy](#)
[PDRegisterAnnotHandler](#)

PDAnnotHandlerDeleteAnnotInfoProc

```
ACCB1 void ACCB2 PDAnnotHandlerDeleteAnnotInfoProc  
(PDAnnotHandler pdanh, PDAnnotInfo info);
```

Description

(Optional) Callback for [PDAnnotHandler](#). Deletes information associated with an annotation. Frees all the memory associated with the annotation information.

Parameters

pdanh	The annotation handler responsible for this annotation.
info	Information associated with the annotation.

Return Value

None

Header File

PDExpT.h

Related Callbacks

[AVAnnotHandlerDeleteInfoProc](#)
[PDAnnotHandlerGetAnnotInfoProc](#)

Related Methods

[PDRegisterAnnotHandler](#)

PDAnnotHandlerDestroyProc

```
ACCB1 void ACCB2 PDAnnotHandlerDestroyProc  
(PDAnnotHandler pdanh) ;
```

Description

(Optional) Callback for [PDAnnotHandler](#). Destroys the annotation handler structure when the application no longer requires it. The handler should destroy any dynamic memory stored in the **userData** field.

Parameters

pdanh	The annotation handler to destroy.

Return Value

None

Header File

PDExpT.h

Related Methods

[PDRegisterAnnotHandler](#)

PDAnnotHandlerDestroyDataProc

```
ACCB1 void ACCB2 PDAnnotHandlerDestroyDataProc  
(PDAnnotHandler pdanh, PDAnnotClipboardData data);
```

Description

(Optional) Callback for [PDAnnotHandler](#). Destroys data from an annotation that has been copied to a clipboard. This callback may be executed regardless of how many times (if any) the annotation was pasted.

Parameters

pdanh	The annotation handler responsible for this annotation.
data	The copied annotation data to destroy.

Return Value

None

Header File

PDExpT.h

Related Callbacks

[PDAnnotHandlerCanCopyProc](#)
[PDAnnotHandlerCanPasteProc](#)
[PDAnnotHandlerCopyProc](#)
[PDAnnotHandlerPasteProc](#)

Related Methods

[PDAnnotDestroyClipboardData](#)
[PDRegisterAnnotHandler](#)

PDAnnotHandlerGetAnnotInfoFlagsProc

ACCB1 [ASFlagBits](#) ACCB2 PDAnnotHandlerGetAnnotInfoFlagsProc
([PDAnnotHandler](#) pdanh, [PDAnnot](#) pdan) ;

Description

Callback for [PDAnnotHandler](#). Gets the annotation handler info flags, which indicate the operations allowed with annotations of this type.

Parameters

pdanh	The annotation handler responsible for the annotation.
pdan	The annotation.

Return Value

Operations allowed. Is an OR of the following flags:

- [PDAnnotOperationSummarize](#) — OK to summarize annotations
- [PDAnnotOperationFilter](#) — OK to filter annotations
- [PDAnnotOperationManager](#) — OK to manage annotations
- [PDAnnotIgnorePerms](#) — Allow modifying this annotation type in a write-protected document
- [PDAnnotOperationAll](#) — All operations are allowed

Header File

PDExpT.h

Related Callbacks

[PDAnnotHandlerGetAnnotInfoProc](#)

Related Methods

[PDRegisterAnnotHandler](#)

PDAnnotHandlerGetAnnotInfoProc

ACCB1 [PDAnnotInfo](#) ACCB2 PDAnnotHandlerGetAnnotInfoProc
([PDAnnotHandler](#) pdanh, [PDAnnot](#) pdan, [PDPage](#) pdpage);

Description

Callback for [PDAnnotHandler](#). Gets the annotation information for an annotation.

Parameters

pdanh	The annotation handler responsible for this annotation.
pdan	The annotation for which information is obtained.
pdpage	The page associated with the annotation for which information is obtained. If the page associated with the annotation is not known, pass NULL .

Return Value

Annotation information, described in a **PDAnnotInfo** structure.

Header File

PDExpT.h

Related Callbacks

[AVAnnotHandlerGetInfoProc](#)
[PDAnnotHandlerDeleteAnnotInfoProc](#)

Related Methods

[PDRegisterAnnotHandler](#)

PDAnnotHandlerGetHeelPointProc

```
ACCB1 void ACCB2 PDAnnotHandlerGetHeelPointProc
(PDAnnotHandler pdanh, PDAnnot pdan, ASFixedPoint *point);
```

Description

Callback for [PDAnnotHandler](#). Gets the heel point (the focus or starting point) for the annotation. For a rectangular annotation, this is usually the top left corner.

Parameters

pdanh	The annotation handler whose heel point is obtained.
pdan	The annotation object.
point	<i>(Filled by the method)</i> The heel point for the annotation.

Return Value

None

Header File

PDExpT.h

Related Methods

[PDRegisterAnnotHandler](#)

PDAnnotHandlerGetPrintAppearanceProc

```
ACCB1 CosObj ACCB2 PDAnnotHandlerGetPrintAppearanceProc
(PDAnnotHandler pdanh, PDAnnot pdan, ASFixedRect *frAnnot,
PDAnnotPrintOp op) ;
```

Description

Callback for [PDAnnotHandler](#). Called by the host application to obtain a print appearance (a Form **XObject**). If this callback is not implemented the default appearance (if any) is used.

Parameters

pdanh	The annotation handler whose print appearance is obtained.
pdan	The annotation object.
frAnnot	The rectangle in user space in which the annotation will be printed. The handler is free to modify it.
op	The print operation being performed. This is kPDAnnotPrintStandard for a standard print operation, or kPDAnnotPrintVariableData if the user selected "Form Fields Only."

Return Value

The Form **XObject** representing the print appearance of the annotation.

Header File

PDExpT.h

Related Methods

[PDRegisterAnnotHandler](#)

PDAnnotHandlerGetTypeProc

```
ACCB1 ASAtom ACCB2 PDAnnotHandlerGetTypeProc  
(PDAnnotHandler pdanh) ;
```

Description

Callback for [PDAnnotHandler](#). Gets an **ASAtom** indicating the annotation type for which the handler is responsible. This corresponds to the annotation's **Subtype** key in the PDF file.

Parameters

pdanh	The annotation handler whose type is returned.

Return Value

The annotation type for which this handler is responsible.

Header File

`PDExpT.h`

Related Callbacks

[PDAnnotHandlerGetAnnotInfoProc](#)

Related Methods

[PDRegisterAnnotHandler](#)

PDAnnotHandlerPasteProc

```
ACCB1 PDAnnot ACCB2 PDAnnotHandlerCanPasteProc  
(PDAnnotHandler pdanh, PDPage destPage,  
const ASFixedPoint *center, PDAnnotClipboardData data);
```

Description

(Optional) Callback for [PDAnnotHandler](#). Creates a new annotation on the specified page using clipboard data generated by [PDAnnotCopy](#).

Parameters

pdanh	The annotation handler responsible for this annotation.
destPage	The page to which the annotation is pasted.
center	The location for the center of the annotation on the destination page, or a NULL pointer to center the annotation in the destination page's crop box.
data	The copied annotation data to paste.

Return Value

The new annotation object associated with the page's document.

Header File

PDExpT.h

Related Callbacks

[PDAnnotHandlerCanCopyProc](#)
[PDAnnotHandlerCanPasteProc](#)
[PDAnnotHandlerCopyProc](#)
[PDAnnotHandlerDestroyDataProc](#)

Related Methods

[PDAnnotCopy](#)
[PDAnnotPaste](#)
[PDRegisterAnnotHandler](#)

PDAnnotWillPrintProc

```
ACCB1 ASBool ACCB2 PDAnnotWillPrintProc (PDAnnotHandler pdanh,  
PDAnnot annot);
```

Description

Callback for [PDAnnotHandler](#). This method is called to determine whether an annotation is printed or not.

Parameters

pdanh	The annotation handler of the annotation type to print.
annot	The annotation to print.

Return Value

true if the annotation is printed, **false** if the annotation is not printed.

Header File

`PDExpT.h`

Related Callbacks

[PDDocWillExportAnnotProc](#)

[PDDocWillImportAnnotProc](#)

Related Methods

[PDRegisterAnnotHandler](#)

PDAuthProc

```
ACCB1 ASBool ACCB2 PDAuthProc (PDDoc pdDoc);
```

Description

Callback used by [PDDocOpen](#). It is called when an encrypted document is being opened to determine whether the user is authorized to open the file.

This callback implements whatever authorization strategy you choose and calls the callbacks of the appropriate security handler (the one that was used to secure the document) to obtain and check authorization data.

The **PDAuthProc** must call the security handler's [PDCryptGetAuthDataProc](#) to obtain whatever authorization data is needed (such as a password), then call [PDDocAuthorize](#) (which is mostly a call to the security handler's [PDCryptAuthorizeProc](#)) to determine whether this data authorizes access to the file (for example, did the user provide the correct password). The **PDAuthProc** must also free the authorization data by calling the security handler's [PDCryptFreeAuthDataProc](#) (or [ASfree](#), if the handler does not have a [PDCryptFreeAuthDataProc](#).)

For Acrobat 3.0 and earlier, the correct way to obtain the security handler in a **PDAuthProc** is to call [PDDocGetNewCryptHandler](#), relying on the fact that it returns the security handler if the document has no new security handler, and the fact that at the time the file is opened, it cannot yet have a new security handler. (In the future, one or more new methods may be added to make this procedure more straightforward.)



The Acrobat viewer's built-in authorization procedure works as follows:

Call the security handler's `PDFCryptAuthorizeProc` with `NULL` authorization data to automatically handle the case where no authorization data is needed (for example, the file has a `NULL` password).

If `PDFCryptAuthorizeProc` returns true
open the file

```
If PDFCryptAuthorizeProc returns false then {
    Loop for i = 1 to 3 {
        Call the security handler's
        PDFCryptGetAuthDataProc

        If PDFCryptGetAuthDataProc returns true {
            Call PDDocAuthorize

            If returns true {
                /* We got authorization */
                Call the security handler's
                PDFCryptFreeAuthDataProc
            }
            exit the loop and return from
            PDAuthProc
        }
        Call the security handler's
        PDFCryptFreeAuthDataProc
    }
    /* Failed to get authorization after
       three attempts */
    Display a dialog box indicating that
    user is not authorized to open the file.
}
return from PDAuthProc
```

Parameters

<code>pdDoc</code>	The <code>PDDoc</code> to open.

Return Value

`true` if the user is authorized to open the document, `false` otherwise.

Header File

`PDExpT.h`

Related Methods

[PDDocAuthorize](#)
[PDDocOpen](#)



PDDocOpenEx

PDAuthProcEx

```
ACCB1 ASBool ACCB2 PDAuthProcEx (PDDoc pdDoc,  
void* clientData);
```

Description

Callback used by [PDDocOpenEx](#). It is called when an encrypted document is opened, to determine whether the user is authorized to open the file.

This callback implements whatever authorization strategy you choose and calls the callbacks of the appropriate security handler (the one that was used to secure the document) to obtain and check authorization data.

The **PDAuthProcEx** should obtain the authorization data (usually a password) and call [PDDocAuthorize](#). [PDDocAuthorize](#) in turn calls the document encryption handler's **Authorize** function, which returns the permissions that the authorization data enables. [PDDocAuthorize](#) adds these permissions to those currently allowed, and returns the new set of allowed permissions.

Parameters

pdDoc	The PDDoc to open.
clientData	User-supplied data that was passed in the call to PDDocOpenEx .

Return Value

true if the user is authorized to open the document, **false** otherwise.

Header File

PDExpT.h

Related Methods

[PDDocAuthorize](#)
[PDDocOpen](#)
[PDDocOpenEx](#)

PDCharProcEnumProc

```
ACCB1 ASBool ACCB2 PDCharProcEnumProc (char* name,
PDCharProc obj, void* clientData);
```

Description

Callback for [PDFontEnumCharProcs](#). It is called once for each character in a Type 3 font.

Parameters

name	The name of the current character.
obj	Stream Cos object containing the PDF drawing operators that draw the character.
clientData	User-supplied data that was passed in the call to PDFontEnumCharProcs .

Return Value

true to continue enumerating, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDFontEnumCharProcs](#)

PDCryptAuthorizeExProc

```
ACCB1 PDPermReqStatus ACCB2 PDCryptAuthorizeExProc (PDDoc doc,
PDPermReqObj reqObj, PDPermReqOpr reqOpr, void* authData);
```

Description

Replaces [PDCryptAuthorizeProc](#). [PDPerms](#) are now obsolete because Acrobat 5.0 introduces new permission controls. However, Acrobat still supports old security handlers.

Called whenever Acrobat needs to get authorization data and/or check permissions for operations.

Parameters

doc	The document for which the request is made.
reqObj	Object type that is the focus of the request: one of the PDPermReqObj values.
reqOpr	Operation type that is the focus of the request: one of the PDPermReqOpr values.
authData	Authorization data. Its format is security handler-specific.

Return Value

The status of the request. One of the [PDPermReqStatus](#) values.

Header File

PDExpT.h

Related Callbacks

[PDCryptAuthorizeProc](#)

Related Methods

[PDDocPermRequest](#)

PDCryptAuthorizeProc

```
ACCB1 PDPperms ACCB2 PDCryptAuthorizeProc (PDDoc pdDoc,
void* authData, PDPperms permWanted);
```

Description

Callback for [PDCryptHandler](#). Called by [PDDocAuthorize](#) when a user tries to set security for an encrypted document and by a [PDAuthProc](#) when a user tries to open a file.

It must decide, based on the contents of the authorization data structure, whether the user is permitted to open the file, and what permissions the user has for this file. The authorization data structure is available in making this decision. Alternate implementations may not require authorization data and may, for example, make authorization decisions based on data contained in the security data structure (use [PDCryptNewSecurityDataProc](#)).

This callback must *not* obtain the authorization data (for example, by displaying a user interface into which a user can type a password). Obtaining authorization data is handled by the security handler's [PDCryptGetAuthDataProc](#), which must be called before this callback. Instead, [PDCryptAuthorizeProc](#) must work with whatever authorization data is passed to it.

It is legitimate for this callback to be called with **NULL** authorization data; the Acrobat viewer's built-in **authProc** does this in order to support authorization methods that do not require authorization data.

When this callback is invoked to determine whether a user is permitted to open a file, **permWanted** is set to **pdPermOpen**. In this case, the file's contents are not yet decrypted (since this callback is being asked to permit decryption), and some calls must be avoided. For example, a call that causes a page to be parsed results in an error, since the *encrypted* contents are parsed. In general, it is safe to obtain information about the presence or absence of things, or the number of things, and to examine any part of a document at the Cos level.

Parameters

pdDoc	The document for which authorized permissions are being requested.
authData	Authorization data. Its format is security handler-specific; each handler can select its own authorization data format.
permWanted	The permissions being requested. Is either pdPermOpen (if the file is being opened) or pdPermSecure (if a request is being made to change the document's security settings).

Return Value

The permissions granted based on the **authData**. For opening, the permissions returned usually should be **pdPermOpen** and some or all of **pdPermPrint**, **pdPermEdit**, and

pdPermCopy. For setting security, permissions returned should be **pdPermAll**. However, if authorization fails, **0** should be returned.

Header File

PDExpT.h

Related Callbacks

[PDCryptAuthorizeExProc](#)

Related Methods

[PDCryptAuthorizeFilterAccess](#)

[PDDocAuthorize](#)

[PDDocOpen](#)

[PDDocOpenEx](#)

PDCryptBatchAuthorizeProc

```
ACCB1 PDPPermReqStatus ACCB2 PDCryptBatchAuthorizeProc
(PDDoc pdDoc, PDPPermReqObj reqObj, PDPPermReqOpr reqOpr,
void* authData);
```

Description

Callback for [PDCryptBatchHandler](#). Called when PDF file is opened. First called with **NULL** **authData** for the case without a user password. Called again with authorization data provided for the security handler that matches the one used in the PDF file.

NOTE: This function is called a batch operation and therefore should not display any user interface. During a batch operation, a file will first be opened with the **pdPermSecure** bit set. It will then be opened with the **pdPermOpen** bit set.

Parameters

pdDoc	The document being opened.
reqObj	Object type that is the focus of the request: one of the PDPPermReqObj values.
reqOpr	Operation type that is the focus of the request: one of the PDPPermReqOpr values.
authData	Authorization data. Its format is security handler-specific.

Return Value

PDPPermReqStatus

Header File

PDExpT.h

PDCryptBatchFreeAuthDataProc

```
ACCB1 void ACCB2 PDCryptBatchFreeAuthDataProc  
(void* authData);
```

Description

Callback for [PDCryptBatchHandler](#). If provided, must be used to free **authData** acquired via **BatchGetAuthData** or **BatchNewAuthData**. If no **BatchFreeAuthData** function is provided, a default one will be used which calls [ASfree](#) on the authData if it is non-NULL.

Parameters

authData	Authorization data. Its format is security handler-specific.

Return Value

None

Header File

PDExpT.h

PDCryptBatchNewAuthDataProc

```
ACCB1 void ACCB2 PDCryptBatchNewAuthDataProc (void);
```

Description

Callback for [PDCryptBatchHandler](#). Different from the regular **PDCryptHandlerNewAuthData** function. Create and return a **void*** which is the authorization data for the batch security handler. This data should be used to both batch open files and batch secure files. Therefore, make sure to provide password information for both the **pdPermOpen** and **pdPermSecure** cases. This data will be passed to the [PDCryptBatchAuthorizeProc](#) callback and eventually to [PDCryptBatchFreeAuthDataProc](#) to free the data. This authorization data is collected before any files are opened in the batch sequence. It is permitted to display a UI at this point since the batch operation has not started yet. The data applies to all files, and therefore could represent one or more passwords which can be enumerated in the **BatchAuthorize** function which receives the batch **authData**.

Parameters

None

Return Value

None

Header File

PDExpT.h

PDFCryptBatchParamDescProc

```
ACCB1 void ACCB2 PDFCryptBatchParamDescProc
(const ASCab settings, ASCab paramDesc);
```

Description

Callback for [PDFCryptBatchHandler](#). Developer should provide information about the current batch settings for the security handler. Batch settings are provided as a read-only **ASCab** that is passed to the function. A writable **ASCab** is also provided, which should be used to store parameter information about the security handler. The description information should be stored starting in the **paramDesc ASCab** using [ASText](#) objects starting with key "1". Example: key="1", value="Title: API Reference" (**ASText** object)

Parameters

settings	Batch settings.
paramDesc	Description information.

Return Value

None

Header File

PDExpT.h

PDCryptBatchPostSequenceProc

```
ACCB1 void ACCB2 PDCryptBatchPostSequenceProc  
(ASCab settings);
```

Description

Callback for [PDCryptBatchHandler](#). This function is called at the end of a batch sequence after all files have been processed. Any memory that was allocated in the **BatchPreSequence** call should be cleaned up in this callback.

Parameters

settings	Batch settings.

Return Value

None

Header File

PDExpT.h

PDFCryptBatchPreSequenceProc

```
ACCB1 ASBool ACCB2 PDFCryptBatchPreSequenceProc
(ASCab settings);
```

Description

Callback for **PDFCryptBatchHandler**. This function is called at the beginning of a batch sequence before any files have been opened. This allows a security handler to be called back with the **ASCab** of settings that were filled out by the **BatchShowDialog** function, or by an **ASCab** that was read in from disk. Pointers of security information are not serialized to disk, and therefore a security information structure may need to be regenerated based on other security information in the **ASCab**. It is permitted for the **BatchPreSequence** callback to put up a UI asking the user for more information since the batch sequence has not started yet. If this function returns **false**, the viewer will assume that the command cannot be executed and will cancel the sequence.

Parameters

settings	Batch settings.

Return Value

See above.

Header File

PDExpT.h

PDCryptBatchShowDialogProc

```
ACCB1 ASBool ACCB2 PDCryptBatchShowDialogProc  
(ASCab settings);
```

Description

Callback for **PDCryptBatchHandler**. This callback puts up a dialog that allows a user to enter data that will be used to batch secure a series of files. The data is stored in an **ASCab** which is part of a batch sequence file. The actual security data, including password(s), should be stored as a pointer in the **ASCab** so that password information is not serialized to disk. Pointers are not serialized from **ASCabs**, but **ASTexts**, **ASInt32s**, and **ASBools** are serialized.

Parameters

settings	An object of type ASCab .

Return Value

true indicates success.

Header File

PDExpT.h

PDCryptBatchUpdateSecurityDataProc

```
ACCB1 ASBool ACCB2 PDCryptBatchUpdateSecurityDataProc  
(PDDoc pdDoc, ASCab settings, ASAtom* cryptHandler,  
void** secDataP);
```

Description

Callback for [PDCryptBatchHandler](#). This function should update the crypt handler's security data without bringing up a dialog. This data is provided by a [PDCryptBatchShowDialogProc](#). The current security data can be obtained by calling [PDDocGetNewSecurityData](#). This function should return **true** unless there is a problem with the batch data for this security handler.

NOTE: This function is called a batch operation and therefore should not display any user interface.

Parameters

pdDoc	The document whose data is updated.
settings	An object of type ASCab .
cryptHandler	An object of type ASAtom .
secDataP	Pointer to the document's security data.

Return Value

See above.

Header File

PDExpT.h

PDCryptDisplaySecurityDataProc

```
ACCB1 AVConversionStatus ACCB2 PDCryptDisplaySecurityDataProc  
(PDDoc doc, ASAtom cryptHandler);
```

Description

Called when the security handler should bring up a document (security) info dialog with the current settings. It also should return **true** when the user wants to change the settings.

If this callback is not supplied, the default info dialog is displayed with **PDPerms** bits information (Acrobat 4.x equivalent dialog).

Parameters

doc	The document whose info is displayed.
cryptHandler	The registered name of the handler.

Return Value

true if the handler wishes a call back to change the settings, **false** otherwise.

Header File

PDExpT.h

PDCryptFillEncryptDictProc

```
ACCB1 void ACCB2 PDCryptFillEncryptDictProc (PDDoc pdDoc,
CosObj encryptDict);
```

Description

Callback for [PDCryptHandler](#). Called when an encrypted document is saved. Fills the document's Encryption dictionary with whatever information the security handler wants to store in the document.

Normally this callback is called after [PDCryptUpdateSecurityDataProc](#). The security data structure can be obtained with a call to [PDDocGetNewSecurityData](#), and the **encryptDict** is filled based on this data.

The sequencing of events that the viewer performs during creation of the **encryptDict** is as follows:

- the viewer creates the **encryptDict**
- the viewer adds the Filter attribute to the dictionary
- calls this [PDCryptFillEncryptDictProc](#) to allow the security handler to add its own attributes to the dictionary
- calls the [PDCryptNewCryptDataExProc](#) (then [PDCryptNewCryptDataProc](#) if unsuccessful) to get the algorithm version, key, and key length
- checks if the V attribute has been added to the dictionary and, if not, then sets V to the algorithm version
- sets the Length attribute if V is 2 or greater
- adds the encryptDict to the document

Parameters

pdDoc	The document to save.
encryptDict	A dictionary Cos object to fill with whatever information the security handler wants to store in the PDF file. Unlike all other strings and streams, direct object elements of the encryptDict are <i>not</i> encrypted automatically. If you want them encrypted, you must encrypt them <i>before</i> inserting them into the dictionary.

Header File

```
PDExpT.h
```

Related Methods

```
PDDocSave
```

PDCryptFilterAuthorizeProc

```
ACCB1 ASBool ACCB2 PDCryptFilterAuthorizeProc (CosDoc dP,  
ASAtom filterName, CosObj encryptDict, ASBool bEncrypt,  
ASBool bUIAllowed) ;
```

Description

(Optional) Callback for [PDCryptFilterHandler](#). Acrobat's security mechanism calls this method to determine whether the user should have access to this filter.

Parameters

dP	The document for which to perform authorization.
filterName	The name of the security filter.
encryptDict	The encryption dictionary to use.
bEncrypt	true if authorization is for an encryption operation, false if it is for a decryption operation.
bUIAllowed	true if the caller can bring up the security dialogs as needed, false otherwise.

Return Value

true if the user has access to this filter, **false** otherwise.

Header File

PDExpT.h

Related Callbacks

[PDCryptFilterGetDataProc](#)
[PDCryptFilterStreamProc](#)
[PDCryptFilterStringProc](#)

Related Methods

[PDCryptAuthorizeFilterAccess](#)
[PDDocSetNewCryptFilterMethod](#)
[PDDocSetNewDefaultFilters](#)

PDCryptFilterGetDataProc

```
ACCB1 void ACCB2 PDCryptFilterGetDataProc (CosDoc dP,  
ASAtom filterName, char **key, ASBool bNewKey,  
ASBool bUIAllowed);
```

Description

(Optional) Callback for [PDCryptFilterHandler](#). Acrobat's security mechanism calls this method to retrieve the encryption/decryption key for this filter. Called only when the filter's encryption method is **V2**.

Parameters

dP	The document whose data is retrieved.
filterName	The name of the security filter.
key	(Filled by the method) A pointer to the encryption/decryption key.
bNewKey	true if you want to retrieve a new key (changed since opening the file), false otherwise.
bUIAllowed	true if the caller can bring up the security dialogs as needed, false otherwise.

Return Value

None

Header File

PDExpT.h

Related Callbacks

[PDCryptFilterAuthorizeProc](#)
[PDCryptFilterStreamProc](#)
[PDCryptFilterStringProc](#)

Related Methods

[PDDocSetNewCryptFilterData](#)
[PDDocSetNewCryptFilterMethod](#)
[PDDocSetNewDefaultFilters](#)

PDCryptFilterStreamProc

```
ACCB1 void ACCB2 PDCryptFilterStreamProc (CosDoc dP,  
ASAtom filterName, ASCryptStm stm, ASBool handOff,  
CosObj params, ASInt32 stmLength);
```

Description

(Optional) Callback for [PDCryptFilterHandler](#). Callbacks that conform to this prototype are called to encrypt or decrypt streams from a document.

The first call to a procedure of this type must fill out an [ASCryptStmRec](#) structure with pointers to callback routines for various types of stream access; see [ASCryptStmProcs](#).

Parameters

dP	The document containing the stream.
filterName	The name of the security filter in use.
stm	The security streamstructure, containing the stream itself, access information, and an ASCryptStmRec . This callback function is called upon opening the stream; the first call must initialize the stream.
handOff	true if the method should close the base stream when it closes the filter, false otherwise.
params	A Cos object containing parameters for the operation, as specified for the filter.
stmLength	For a decryption operation, the requested number of bytes; for an encryption operation, the number of bytes in the stream contained in stm .

Return Value

None

Header File

PDExpT.h

Related Callbacks

[PDCryptFilterAuthorizeProc](#)
[PDCryptFilterGetDataProc](#)
[PDCryptFilterStringProc](#)

Related Methods

[PDDocSetNewCryptFilterData](#)
[PDDocSetNewCryptFilterMethod](#)



PDDocSetNewDefaultFilters

PDCryptFilterStringProc

```
ACCB1 ASInt32 ACCB2 PDCryptFilterStringProc (CosDoc dP,
ASAtom filterName, char *dest, char *src, ASInt32 dstSize,
ASInt32 srcLength, ASInt32 genNumber, ASInt32 objNumber);
```

Description

(Optional) Callback for [PDCryptFilterHandler](#). Callbacks that conform to this prototype are called to encrypt or decrypt strings from a document.

The function returns the result in **dest**. When **NULL** is passed as the destination buffer, the function must return the desired buffer size (required destination buffer length) in **dstSize**.

Parameters

dP	The document containing the string.
filterName	The name of the security filter in use.
dest	The character buffer for the result, or NULL to get the needed buffer size for an encryption or decryption operation.
src	The string to be encrypted or decrypted.
dstSize	The size of dest in bytes, or , when dest is NULL , the required destination buffer length.
srcLength	The length of src in bytes.
genNumber objNumber	Can be used to generate a unique key for the string.

Return Value

None

Header File

PDExpT.h

Related Callbacks

[PDCryptFilterAuthorizeProc](#)
[PDCryptFilterGetDataProc](#)
[PDCryptFilterStreamProc](#)

Related Methods

[PDDocSetNewCryptFilterData](#)
[PDDocSetNewCryptFilterMethod](#)
[PDDocSetNewDefaultFilters](#)

PDCryptFreeAuthDataProc

```
ACCB1 void ACCB2 PDCryptFreeAuthDataProc (PDDoc pdDoc,  
void* authData);
```

Description

(Optional) Callback for [PDCryptHandler](#). Used to free authorization data acquired via [PDCryptNewAuthDataProc](#). If this callback is omitted, the viewer defaults to freeing the data using [ASfree](#).

Parameters

pdDoc	The document whose authorization data is freed.
authData	<i>(Filled by the callback)</i> Pointer to the document's authorization data.

Return Value

None

Header File

PDExpT.h

Related Callbacks

[PDCryptNewAuthDataProc](#)

Related Methods

[PDDocGetNewSecurityInfo](#)

PDCryptFreeCryptDataProc

```
ACCB1 void ACCB2 PDCryptFreeCryptDataProc (PDDoc pdDoc,  
char* cryptData);
```

Description

(Optional) Callback for [PDCryptHandler](#). Used to free authorization data acquired via [PDCryptNewCryptDataProc](#). If this callback is omitted, the viewer defaults to freeing the data using [ASfree](#).

Parameters

pdDoc	The document whose encryption/decryption data is freed.
cryptData	(Filled by the callback) Pointer to the document's encryption/decryption data.

Return Value

None

Header File

PDExpT.h

Related Callbacks

[PDCryptNewCryptDataProc](#)

Related Methods

[PDDocGetNewSecurityInfo](#)

PDCryptFreeSecurityDataProc

```
ACCB1 void ACCB2 PDCryptFreeSecurityDataProc (PDDoc pdDoc,  
void* secData);
```

Description

(Optional) Callback for [PDCryptHandler](#). Used to free security data acquired via [PDCryptNewSecurityDataProc](#). If this callback is omitted, the viewer defaults to freeing the data using [ASfree](#).

Parameters

pdDoc	The document whose security data is freed.
secData	(Filled by the callback) Pointer to the document's security data.

Return Value

None

Header File

`PDExpT.h`

Related Callbacks

[PDCryptNewSecurityDataProc](#)

Related Methods

[PDDocGetNewSecurityInfo](#)

PDFCryptGetAuthDataExProc

```
ACCB1 ASBool ACCB2 PDFCryptGetAuthDataExProc (PDDoc pdDoc,
PDFPermReqObj reqObj, PDFPermReqOpr reqOpr, void** authDataP);
```

Description

Replaces [PDFCryptGetAuthDataProc](#). Called whenever Acrobat needs to get authorization data and/or check permissions for operations.

[PDFPerms](#) are now obsolete because Acrobat 5.0 introduces new permission controls. However, Acrobat still supports old security handlers.

Parameters

pdDoc	The document for which the request is made.
reqObj	Object type that is the focus of the request: one of the PDFPermReqObj values.
reqOpr	Operation type that is the focus of the request: one of the PDFPermReqOpr values.
authDataP	Pointer to an authorization data structure. Its format is security handler specific.

Return Value

true unless the operation should be canceled (for example, if the user cancels a dialog), **false** otherwise..

Header File

PDExpT.h

Related Callbacks

[PDFCryptGetAuthDataProc](#)

PDCryptGetAuthDataProc

```
ACCB1 ASBool ACCB2 PDCryptGetAuthDataProc (PDDoc pdDoc,
PDPerms permWanted, void** authDataP);
```

Description

Callback for [PDCryptHandler](#). This callback is called from a [PDAuthProc](#) when a file is opened after [PDCryptNewSecurityDataProc](#) is called.

The callback must determine the user's authorization properties for the document by obtaining authorization data, such as a user interface log in or password entry. It populates an authorization data structure with this data.

This callback may call the security handler's [PDCryptNewAuthDataProc](#) to allocate the authorization data structure. Use of an authorization data structure is optional (an implementation may wish to contain authorization data within the security data structure). The authorization data structure is subsequently used by the security handler's [PDCryptAuthorizeProc](#) to determine whether the user is authorized to open the file.

A security handler can specify the standard password dialog by using [AVCryptGetPassword](#). In this case, the **authData** is a **char***.

Parameters

pdDoc	The document to open.
permWanted	Either pdPermOpen or pdPermSecure . Since this value is also passed to PDCryptAuthorizeProc , it may not be necessary for this callback to use permWanted .
authDataP	Pointer to authorization data structure. Set to NULL if not used.

Return Value

true unless the operation should be canceled (for example, if the user cancels a dialog), **false** otherwise.

Header File

`PDExpT.h`

Related Callbacks

[PDCryptGetAuthDataExProc](#)
[PDCryptNewAuthDataProc](#)

Related Methods

[PDDocOpen](#)
[PDDocOpen](#)
[PDDocOpenEx](#)

PDCryptGetDocPermsProc

```
ACCB1 void ACCB2 PDCryptGetDocPermsProc (PDDoc pdDoc,
ASBool perms[PDPPermReqObjLast][PDPPermReqOprLast],
ASInt16 *version);
```

Description

Callback for [PDCryptHandler](#). This function should extract and return information about the document permissions for display to the user: whether the user can print, edit, copy text and graphics, edit notes and do form fill in and signing.

The permissions returned are logically ANDed with the document permissions returned by any other permissions handlers and displayed to the user. All crypt handlers should implement this call so that consolidated permissions can be displayed. To display your own crypt handler's permissions, implement [PDCryptDisplaySecurityDataProc](#).

If this callback is absent, Acrobat assumes that all the operations on the document are allowed.

Parameters

pdDoc	The document whose permissions are obtained.
perms	<i>(Filled by the callback)</i> An array of the document's permissions. For each combination of PDPPermReqObj and PDPPermReqOpr , the value is true if the operation is allowed for the object, false if it is not.
version	<i>(Filled by the callback)</i> Pointer to the version number of PDPPermReqObj and PDPPermReqOpr with which this crypt handler is compatible (specified by the constant PDPPermReqVersion).

Return Value

None

Header File

PDExpT.h

Related Methods

[PDDocPermRequest](#)

PDCryptGetSecurityInfoProc

```
ACCB1 void ACCB2 PDCryptGetSecurityInfoProc (PDDoc pdDoc,
ASInt32* secInfo);
```

Description

(Optional) Callback for [PDCryptHandler](#). Called by [PDDocGetNewSecurityInfo](#). Extracts the security information from the security data structure, and returns the security information.

This function is also used after a “Save As...” to reset the permissions according to the current document.

A default set of permissions is used if this callback is absent:

```
pdInfoCanPrint | pdInfoCanEdit | pdInfoCanCopy | pdInfoCanEditNotes
```

See [PDPerms](#).

Parameters

pdDoc	The document whose security info is obtained.
secInfo	(Filled by the callback) The document’s security info. The value must be an OR of the Security Info Flags . All unused bits in must be set to 1.

Return Value

None

Header File

PDExpT.h

Related Methods

[PDDocGetNewSecurityInfo](#)

PDCryptNewAuthDataProc

```
ACCB1 void* ACCB2 PDCryptNewAuthDataProc (PDDoc pdDoc);
```

Description

(Optional) Callback for [PDCryptHandler](#). Creates a new empty authorization data structure. This structure is subsequently filled by [PDCryptGetAuthDataProc](#), then passed to [PDCryptAuthorizeProc](#) and eventually to [ASfree](#).

This callback is not called by the Acrobat viewer, but a security handler may use it if it wishes. The Acrobat viewer's standard security handler does not use this method.

Parameters

pdDoc	The document for which a new authorization data structure is created.

Return Value

The newly created authorization data structure.

Header File

PDExpT.h

Related Callbacks

[PDCryptFreeAuthDataProc](#)
[PDCryptGetAuthDataProc](#)

PDCryptNewCryptDataProc

```
ACCB1 void ACCB2 PDCryptNewCryptDataProc (PDDoc pdDoc,  
char** cryptData, ASInt32* cryptDataLen);
```

Description

Callback for [PDCryptHandler](#). Sets up the key to be passed to initialize the RC4 cipher for encryption and decryption of a PDF file. It is called when an encrypted document is opened or saved.

Parameters

pdDoc	The document for which the key is set.
cryptData	<i>(Filled by the callback)</i> The key. cryptData must be allocated by ASmalloc because the Acrobat viewer will free it using ASfree .
cryptDataLen	<i>(Filled by the callback)</i> The number of bytes in cryptData . Cannot be greater than 5 bytes.

Return Value

None

Header File

PDExpT.h

Related Callbacks

[PDCryptNewAuthDataProc](#)

PDcryptNewCryptDataExProc

```
ACCB1 void ACCB2 PDcryptNewCryptDataExProc (PDDoc pdDoc,
char** cryptData, ASInt32* cryptDataLen,
ASInt32* cryptVersion);
```

Description

Callback for [PDcryptHandler](#). Sets up the key to be passed to initialize the RC4 cipher for encryption and decryption of a PDF file. It is called when an encrypted document is opened or saved.

The key is truncated when the length is greater than the viewer currently supports. Data is freed by [PDcryptFreeCryptDataProc](#) if provided. Otherwise, [ASfree](#) is used.

Parameters

pdDoc	The document for which the key is set.
cryptData	<i>(Filled by the callback)</i> The key. cryptData must be allocated by ASmalloc because the Acrobat viewer will free it using ASfree .
cryptDataLen	<i>(Filled by the callback)</i> The number of bytes in cryptData . Cannot be greater than 5 bytes.
cryptVersion	The Cos crypt version—the version of the algorithm that is used to encrypt and decrypt document data. cryptVersion equal to 0 is treated as cryptVersion equal to 1 to maintain backward compatibility.

Return Value

None

Header File

PDExpT.h

Related Callbacks

[PDcryptFreeCryptDataProc](#)

PDFCryptNewSecurityDataProc

```
ACCB1 void* ACCB2 PDFCryptNewSecurityDataProc (PDDoc pdDoc,
CosObj encryptDict);
```

Description

(Optional) Callback for [PDFCryptHandler](#). Creates and populates a new structure that contains whatever security-related information the security handler requires (for example, permissions, whether the file has owner and/or user passwords, owner and/or user passwords, or other data used internally by the security handler). If **encryptDict** is not **NULL**, the structure should be populated based on **encryptDict**'s contents. This method is intended only to initialize the security data structure.

This callback is called under two circumstances:

- When a document is opened, it is called with **encryptDict** set to the document's Encryption dictionary. The handler should then populate the new security data structure with data that is obtained from the Encryption dictionary.
- When the user chooses a new encryption method, it is called without an **encryptDict**. The handler should return a security data structure with default values.

If a security handler does not have this callback, the document's **newSecurityData** field is set to **NULL**.

If a file is to be saved, then [PDFCryptUpdateSecurityDataProc](#) is subsequently called to allow user interface modification of the contents.

Security data is freed using [PDFCryptFreeSecurityDataProc](#). If [PDFCryptFreeSecurityDataProc](#) is not defined, **ASfree** is used.

Parameters

pdDoc	The document for which a new security data structure is created.
encryptDict	If encryptDict is a dictionary, this callback must initialize the security data so that it corresponds to the dictionary. Otherwise, it must set up default values in the security data structure.

Return Value

The newly created security data structure.

Header File

```
PDFExpT.h
```

Related Callbacks

[PDFCryptFreeSecurityDataProc](#)



PDCryptReservedProc

```
ACCB1 void* ACCB2 PDCryptReservedProc (void);
```

Description

(Optional) Callback for [PDCryptHandler](#). Used by Acrobat WebBuy proprietary method of passing crypt data.

Parameters

None

Return Value

None

Header File

PDExpT.h

PDCryptUpdateSecurityDataProc

```
ACCB1 ASBool ACCB2 PDCryptUpdateSecurityDataProc (PDDoc pdDoc,
ASAtom* cryptHandler, void** secDataP);
```

Description

Callback for [PDCryptHandler](#). Updates the security data structure that was created by [PDCryptNewSecurityDataProc](#). This structure can be obtained by calling [PDDocGetNewSecurityData](#). The security data structure of the previously saved file can be obtained with a call to [PDDocGetSecurityData](#).

The security data structure should be updated to reflect the encryption parameters that will be used when saving the file. (This information is usually obtained via dialogs.) The encryption parameters are transferred to the Encrypt dictionary by a subsequent callback to [PDCryptFillEncryptDictProc](#).

The security data should be allocated by [ASmalloc](#) or a related function. Security data is freed using [PDCryptFreeSecurityDataProc](#). If [PDCryptFreeSecurityDataProc](#) is not defined, [ASfree](#) is used.

The callback can also update the security handler itself. For example, the standard encryption handler switches to no encryption if no passwords or permissions are set in the security dialog box. Return [ASAtomNull](#) in [cryptHandler](#) if no encryption is used in the saved file.

Parameters

pdDoc	The document whose security data is updated.
cryptHandler	The current security handler for pdDoc . Can be modified to change the security handler. Encryption is turned off if ASAtomNull is set.
secDataP	(Required) Security data structure. Its content and organization is up to the security handler.

Return Value

true unless the operation should be canceled (for example, the user clicked on the Cancel button).

Header File

`PDExpT.h`

Related Callbacks

[PDCryptValidateSecurityDataProc](#)

Related Methods

[PDDocGetSecurityData](#)

PDCryptValidateSecurityDataProc

```
ACCB1 void ACCB2 PDCryptValidateSecurityDataProc (PDDoc pdDoc,  
void* secData);
```

Description

(Optional) Callback for [PDCryptHandler](#). Validates the security data structure, which specifies the user's permissions. This callback may modify the security data structure, for example because the user is not authorized to change the security as they requested. A client may have called [PDDocNewSecurityData](#) to obtain a new security data structure, then modified it, and then called [PDDocSetNewSecurityData](#) to change the document security. This callback should be called before actually setting the document's security data.

This callback is not called automatically by the Acrobat viewer. It must be called, if desired, by the security handler's [PDCryptUpdateSecurityDataProc](#).

Parameters

pdDoc	The document whose security data is validated.
secData	<i>(May be modified by the callback)</i> The document's security data.

Return Value

None

Header File

PDExpT.h

Related Callbacks

[PDCryptUpdateSecurityDataProc](#)

Related Methods

[PDDocNewSecurityData](#)

[PDDocSetNewSecurityData](#)



PDDocEnumProc

```
ACCB1 ASBool ACCB2 PDDocEnumProc (PDDoc pdDoc,  
void* clientData);
```

Description

Callback for [PDEnumDocs](#). It is called once for each open **PDDoc**.

Parameters

pdDoc	The PDDoc currently being enumerated.
clientData	User-supplied data that was passed in the call to PDEnumDocs .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDEnumDocs](#)

PDDocPreSaveProc

```
ACCB1 void ACCB2 PDDocPreSaveProc (PDDoc pdDoc,
PDDocPreSaveInfo preSaveInfo, void* clientData);
```

Description

Callback in the [PDDocSaveParams](#) structure used by [PDDocSaveWithParams](#). Use this callback to flag Cos objects you wish to access while a **PDDoc** is being saved.

Parameters

pdDoc	The PDDoc to be saved.
preSaveInfo	A PDDocPreSaveInfo structure containing information to use during processing before the save.
clientData	User-supplied data that was specified in preSaveProcClientData of the PDDocSaveParams structure.

Return Value

None

Header File

PDExpT.h

Related Methods

[PDDocSaveWithParams](#)

PDDocPreWriteProc

```
ACCB1 void ACCB2 PDDocPreWriteProc (PDDoc pdDoc,  
void* clientData);
```

Description

Callback in the [PDDocSaveParams](#) structure. Invoked by [PDDocSaveWithParams](#) immediately before a **PDDoc** is saved to disk.

Parameters

pdDoc	The PDDoc to be saved.
clientData	User-supplied data that was specified in preWriteProcClientData of the PDDocSaveParams structure.

Return Value

None

Header File

PDExpT.h

Related Methods

[PDDocSaveWithParams](#)

PDDocRequestEntireFileProc

```
ACCB1 ASInt32 ACCB2 PDDocRequestEntireFileProc (PDDoc pdDoc,  
PDDocRequestReason reason, void* clientData);
```

Description

Callback used by [PDDocRequestEntireFile](#). Use this callback to process a document file.

Parameters

pdDoc	The PDDoc to be saved.
startPage	The first page requested.
nPages	The number of pages requested.
requestReason	A constant representing the status of the request. One of: ● kPDDocRequestUnderway ● kPDDocRequestComplete ● kPDDocRequestCancelled ● kPDDocRequestError
clientData	User-supplied data passed in the PDDocRequestPages method.

Return Value

0 when successful, otherwise a non-zero error code.

Header File

`PDExpT.h`

Related Methods

[PDDocRequestEntireFile](#)

PDDocRequestPagesProc

```
ACCB1 ASInt32 ACCB2 PDDocRequestPagesProc (PDDoc pdDoc,
ASInt32 startPage, ASInt32 nPages, PDDocRequestReason reason,
void* clientData);
```

Description

Callback used by [PDDocRequestPages](#). Use this callback to process a set of pages from the document.

Parameters

pdDoc	The PDDoc to be saved.
startPage	The first page requested.
nPages	The number of pages requested.
requestReason	A constant representing the status of the request. One of: • kPDDocRequestUnderway • kPDDocRequestComplete • kPDDocRequestCancelled • kPDDocRequestError
clientData	User-supplied data passed in the PDDocRequestPages method.

Return Value

0 when successful, otherwise a non-zero error code.

Header File

PDExpT.h

Related Methods

[PDDocRequestPages](#)

PDDocWillExportAnnotCallback

```
ACCB1 ASBool ACCB2 PDDocWillExportAnnotCallback (PDDoc doc,  
PDPAGE pdpage, PDAnnot annot, CosObj dict);
```

Description

Callback for [PDDocExportNotes](#). Determines whether an annotation is exported or not.

NOTE: This is a different callback than [PDDocWillExportAnnotProc](#).

Parameters

doc	The document from which annotations may be exported.
pdPage	The page from which the annotation may be exported.
annot	The annotation that may be exported.
dict	Copy of annot in a Cos object.

Return Value

true to export **annot**, **false** to not export **annot**.

Header File

PDExpT.h

Related Callbacks

[PDDocWillImportAnnotCallback](#)

PDDocWillExportAnnotProc

```
ACCB1 ASBool ACCB2 PDDocWillExportAnnotProc  
(PDAnnotHandler pdanh, PDAnnot src, PDAnnot dst);
```

Description

Callback for [PDAnnotHandler](#). Determines whether an annotation is exported or not.

NOTE: This is a different callback than [PDDocWillImportAnnotCallback](#).

Parameters

pdanh	The annotation handler of the annotation type to export.
src	The annotation that may be exported.
dst	Copy of src , which is actually exported.

Return Value

true to export **dst**, **false** to not export **dst**.

Header File

PDExpT.h

Related Callbacks

[PDAnnotWillPrintProc](#)
[PDDocWillImportAnnotProc](#)

Related Methods

[PDRegisterAnnotHandler](#)

PDDocWillImportAnnotCallback

```
ACCB1 ASBool ACCB2 PDDocWillImportAnnotCallback (PDDoc doc,  
PDPage pdPage, PDAnnot annot);
```

Description

Callback for [PDDocImportCosDocNotes](#) and [PDDocImportNotes](#). Determines whether an annotation will be imported or not.

NOTE: This is a different callback than [PDDocWillImportAnnotProc](#).

Parameters

doc	The document into which annotations may be imported.
pdPage	The page in which the annotation may be imported.
annot	The annotation that may be imported.

Return Value

true to import **annot**, **false** to not import **annot**.

Header File

`PDExpT.h`

Related Callbacks

[PDDocWillExportAnnotCallback](#)

PDDocWillImportAnnotProc

```
ACCB1 ASBool ACCB2 PDDocWillImportAnnotProc  
(PDAnnotHandler pdanh, PDDoc doc, PDPage pdpage,  
PDAnnot annot);
```

Description

Callback for [PDAnnotHandler](#). Determines whether an annotation will be imported or not.

NOTE: This is a different callback than [PDDocWillImportAnnotCallback](#).

Parameters

pdanh	The annotation handler of the annotation type to import.
doc	The document into which annotations may be imported.
pdPage	The page in which the annotation may be imported.
annot	The annotation that may be imported.

Return Value

true to import **annot**, **false** to not import **annot**.

Header File

[PDExpT.h](#)

Related Callbacks

[PDAnnotWillPrintProc](#)
[PDDocWillExportAnnotProc](#)

Related Methods

[PDRegisterAnnotHandler](#)

PDEAttrEnumProc

```
ACCB1 ASBool ACCB2 PDEAttrEnumProc (void* attrHdrP,  
ASInt32 refCount, ASUns16 size, void* clientData);
```

Description

Callback for [PDEAttrEnumTable](#). It is called once for each attribute in a table.

Parameters

attrHdrP	An opaque pointer to the attribute. The actual attribute type is not specified in this function, since the storage mechanism only knows the size of the attribute, not its type.
refCount	Reference count of the attribute.
size	Size of attrHdrP , in bytes.
clientData	User-supplied data that was specified in the call to PDEAttrEnumTable .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PEExpT.h

Related Methods

[PDEAttrEnumTable](#)



PDEClipEnumProc

```
ACCB1 ASBool ACCB2 PDEClipEnumProc (PDEElement elem,  
void* clientData);
```

Description

Callback for [PDEClipFlattenedEnumElems](#), which enumerates all of a [PDEClip](#)'s [PDEElements](#) in a flattened manner.

Parameters

elem	The PDEElement currently being enumerated.
clientData	User-supplied data that was passed in the call to PDEClipFlattenedEnumElems .

Return Value

If **false**, enumeration halts. If **true**, enumeration continues.

Header File

PEExpT.h

Related Methods

[PDEClipFlattenedEnumElems](#)



PDEElementEnumProc

```
ACCB1 ASBool ACCB2 PDEElementEnumProc (PDEElement element,
void* clientData);
```

Description

Callback for [PDEEnumElements](#). It is called once for each [PDEElement](#) in a page's Contents Stream or Resources dictionary.

Parameters

element	The PDEElement .
clientData	User-supplied data that was specified in the call to PDEEnumElements .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PEExpT.h

Related Methods

[PDEEnumElements](#)

PDEObjectDumpProc

```
ACCB1 void ACCB2 PDEObjectDumpProc (PDEObject obj,
char* dumpInfo, void* clientData);
```

Description

Callback for [PDELogDump](#) or [PDEObjectDump](#). It is called once for each **PDEObject**, its children, and their attributes for the specified number of levels.

Parameters

obj	The PDEObject .
dumpInfo	Contains information about an object. Information fields are delimited by tabs. There are no newline characters in this string.
clientData	User-supplied data that was specified in the call to PDELogDump or PDEObjectDump .

Return Value

None

Header File

PEExpT.h

Related Methods

[PDELogDump](#)
[PDEObjectDump](#)

PDFFileSpecAcquireASPathProc

```
ACCB1 ASPathName ACCB2 PDFFileSpecAcquireASPathProc  
(void* fileSpecHandlerObj, PDFFileSpec fileSpec,  
ASPathName relativeToThisPath);
```

Description

Callback for [PDFFileSpecHandler](#). Acquires the **ASPath** corresponding to a file specification.

Parameters

fileSpecHandlerObj	User-supplied data passed in the call to PDRegisterFileSpecHandler .
fileSpec	The PDFFileSpec for which an ASPath is acquired.
relativeToThisPath	A pathname relative to which the PDFFileSpec is interpreted. If NULL , fileSpec is assumed to be an absolute, not a relative, path.

Return Value

The **ASPathName** corresponding to the specified path.

Header File

`PDExpT.h`

Related Methods

[PDFFileSpecAcquireASPath](#)

PDFFileSpecNewFromASPathProc

```
ACCB1 PDFFileSpec ACCB2 PDFFileSpecNewFromASPathProc  
(void* fileSpecHandlerObj, PDDoc pdDoc, ASPathName path,  
ASPathName relativeToThisPath);
```

Description

Callback for [PDFFileSpecHandler](#). Creates a file specification from an **ASPath**.

Parameters

fileSpecHandlerObj	User-supplied data passed in the call to PDRegisterFileSpecHandler .
pdDoc	The PDDoc in which the file specification is created.
path	The ASPathName for which a corresponding file specification is created.
relativeToThisPath	A pathname relative to which path is interpreted. If NULL , path is assumed to be an absolute, not a relative, path.

Return Value

The file specification corresponding to the specified **ASPathName**.

Header File

PDExpT.h

Related Methods

[PDFFileSpecNewFromASPath](#)

PDFontEnumProc

```
ACCB1 ASBool ACCB2 PDFontEnumProc (PDFont font,  
PDFontFlags* fontFlags, void* clientData);
```

Description

Callback used by [PDDocEnumFonts](#) and [PDDocEnumLoadedFonts](#). It is called once for each font.

Parameters

font	The font currently being enumerated.
fontFlags	Contains additional information about the font.
clientData	User-supplied data passed in the call to PDDocEnumFonts or PDDocEnumLoadedFonts .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDDocEnumFonts](#)
[PDDocEnumLoadedFonts](#)



PDGetDataProc

```
ACCB1 ASBool ACCB2 PDGetDataProc (char* data, ASInt32 lenData,
void* clientData);
```

Description

Callback for [PDXObjectGetData](#). It is passed the **XObject**'s data. Currently, the **XObject**'s data is read 1 kB at a time and passed to this callback.

Parameters

data	Buffer containing the XObject 's data.
lenData	The amount of data in data , in bytes.
clientData	User-supplied data that was passed in the call to PDXObjectGetData .

Return Value

true to continue reading the **XObject**'s data, **false** to halt it.

Header File

PDExpT.h

Related Methods

[PDXObjectGetData](#)

PDGraphicEnumCacheDeviceProc

```
ACCB1 ASBool ACCB2 PDGraphicEnumCacheDeviceProc  
(ASFixed* parms, void* clientData);
```

Description

Callback for [PDGraphicEnumMonitor](#). Called for every **d1** (that is, **setcachedevice**) operator.

Parameters

parms	Array of numbers containing the 6 parameters passed to the d1 operator.
clientData	User-supplied data that was passed in the call to PDPageEnumContents .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDPageEnumContents](#)

PDGraphicEnumCharWidthProc

```
ACCB1 ASBool ACCB2 PDGraphicEnumCharWidthProc  
(ASFixedPoint width, void* clientData);
```

Description

Callback for [PDGraphicEnumMonitor](#). Called for every **d0** (that is, **setcharwidth**) operator.

Parameters

width	Array of numbers containing the two parameters passed to the d0 operator.
clientData	User-supplied data that was passed in the call to PDPageEnumContents .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

`PDExpT.h`

Related Methods

[PDPageEnumContents](#)

PDGraphicEnumImageProc

```
ACCB1 ASBool ACCB2 PDGraphicEnumImageProc (PDInlineImage obj,  
void* clientData);
```

Description

Callback for [PDGraphicEnumMonitor](#). Called for every image operator.

Parameters

obj	Image data.
clientData	User-supplied data that was passed in the call to PDPageEnumContents .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDPageEnumContents](#)

PDGraphicEnumPathProc

```
ACCB1 ASBool ACCB2 PDGraphicEnumPathProc (PDPath obj,  
void* clientData);
```

Description

Callback for [PDGraphicEnumMonitor](#). Called for every path operator.

Parameters

obj	The path data.
clientData	User-supplied data that was passed in the call to PDPageEnumContents .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDPageEnumContents](#)

PDGraphicEnumRestoreProc

```
ACCB1 ASBool ACCB2 PDGraphicEnumRestoreProc  
(void* clientData);
```

Description

Callback for [PDGraphicEnumMonitor](#). Called for every **Q (restore)** operator.

Parameters

clientData	User-supplied data that was passed in the call to PDPageEnumContents .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDPageEnumContents](#)

PDGraphicEnumSaveProc

```
ACCB1 ASBool ACCB2 PDGraphicEnumSaveProc (void* clientData);
```

Description

Callback for [PDGraphicEnumMonitor](#). Called for every **Q (save)** operator.

Parameters

clientData	User-supplied data that was passed in the call to PDPageEnumContents .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDPageEnumContents](#)

PDGraphicEnumTextProc

```
ACCB1 ASBool ACCB2 PDGraphicEnumTextProc (PDText obj,  
void* clientData);
```

Description

Callback for [PDGraphicEnumMonitor](#). Called for every text operator.

Parameters

obj	The text object.
clientData	User-supplied data that was passed in the call to PDPageEnumContents .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDPageEnumContents](#)

PDGraphicEnumXObjectRefProc

ACCB1 [ASBool](#) ACCB2 PDGraphicEnumXObjectRefProc (char* name, [ASFixedRect](#)* bbox, void* clientData);

Description

Callback for [PDGraphicEnumMonitor](#). Called for every **XObject (Do)** operator.

Parameters

name	The XObject 's name.
bbox	The XObject 's bounding box, describing the bounding box of the XObject in user space. This is only the case for top-level XObjects . If a Form XObject refers to another XObject , the second XObject 's bounding box is the "infinity" bounding box.
clientData	User-supplied data that was passed in the call to PDPageEnumContents .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDPageEnumContents](#)

PDGraphicEnumXObjectRefMatrixProc

```
ACCB1 ASBool ACCB2 PDGraphicEnumXObjectRefMatrixProc  
(ASFixedMatrix* matrix, void* clientData);
```

Description

Callback for [PDGraphicEnumMonitor](#). Gets the current matrix for the subsequent [XObject](#). Called immediately before [PDGraphicEnumXObjectRefProc](#).

Parameters

matrix	(Filled by the callback) The current transformation matrix for the subsequent XObject whose name is obtained by PDGraphicEnumXObjectRefProc .
clientData	User-supplied data that was passed in the call to PDPageEnumContents .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

`PDExpT.h`

Related Methods

[PDPageEnumContents](#)

PDImplicitMetadataProc

```
ACCB1 void ACCB2 PDImplicitMetadataProc (PDDoc pdDoc,
void* data);
```

Description

Calculates implicit metadata. Clients that maintain metadata items that have to be recalculated should register for the [PDDocCalculateMetadata](#) notification with this callback. The callback should obtain the metadata with which it's concerned, change it, and put the changed metadata back on the object from which it was obtained.

Parameters

pdDoc	The document containing the metadata.
data	User-supplied data.

Return Value

None

Header File

PDMetadataExpT.h

Related Notifications

[PDDocCalculateMetadata](#)

PDLaunchActionProc

```
ACCB1 ASBool ACCB2 PDLaunchActionProc
(void* fileSpecHandlerObj, PDDoc pdDoc, PDAction pdAction);
```

Description

(Optional) Callback for [PDFFileSpecHandler](#). Launches a specified file. Called when the Acrobat viewer encounters a Launch (GoTo File) action. If this callback is **NULL**, no launch action is performed.

Parameters

fileSpecHandlerObj	The registered PDFFileSpecHandler .
pdDoc	The document containing the Launch action.
pdAction	The action dictionary.

Return Value

true if the handler can do the Launch, **false** otherwise.

Header File

PDExpT.h



PDOCConfigEnumProc

```
ACCB1 ASBool ACCB2 PDOCConfigEnumProc (PDOCConfig occonfig,  
void* clientData );
```

Description

A callback used for enumerating optional-content configurations. Enumeration stops when all configurations have been enumerated, or when the callback returns **false**.

Parameters

occonfig	The optional-content configuration object
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDDocEnumOCConfigs](#)



PDOCGEnumProc

```
ACCB1 ASBool ACCB2 PDOCGEnumProc ( PDOCG pdocg,  
void* clientData );
```

Description

A callback used for enumerating optional-content groups (OCGs). Enumeration stops when all OCGs have been enumerated, or when the callback returns **false**.

Parameters

pdocg	The optional-content group object
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDDocEnumOCGs](#)
[PDPageEnumOCGs](#)

PDPageEnumInksCallback

```
ACCB1 ASBool ACCB2 PDPageEnumInksCallback (PDPageInk ink,  
void *clientData);
```

Description

A callback used for enumerating inks on a page. Enumeration stops when all inks have been enumerated, or when the callback returns **false**.

Parameters

ink	The ink object
clientData	Pointer to user-supplied data to pass to proc each time it is called.

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDPageEnumInks](#)

PDPageStmImageDataProc

```
ACCB1 ASBool ACCB2 PDPageStmImageDataProc (ASUns8* data,
ASSize_t dataLen, void* clientData);
```

Description

Callback for [PDPageStmGetInlineImage](#). Should be called when inline image data is encountered in [PDPageStmGetToken](#). This method may be called multiple times for one inline image. If so, each call provides sequential data for the image.

Parameters

data	The image data read so far.
dataLen	Length of data, in bytes.
clientData	User-supplied data that was passed in the call to PDPageStmGetInlineImage (which may have been passed in the PDPageStmGetToken method).

Return Value

true to continue reading the image's data, false to stop reading.

Header File

```
PDExpT.h
```

Related Methods

```
PDPageStmGetInlineImage
PDPageStmGetToken
```

PDPageStmStringOverflowProc

```
ACCB1 void ACCB2 PDPageStmStringOverflowProc (char* sVal,  
ASize\_t sValLen, void* clientData);
```

Description

Callback used by [PDPageStmGetToken](#). It is called when the length of a string token exceeds **kPDPageStmStringMax** bytes (see PDExpT.h) in [PDPageStmGetToken](#).

Parameters

sVal	The string value read so far.
sValLen	Length of sVal , in bytes.
clientData	User-supplied data that was passed in the call to PDPageStmGetToken .

Return Value

None

Header File

PDExpT.h

Related Methods

[PDPageStmGetToken](#)

PDPathClosePathProc

```
ACCB1 ASBool ACCB2 PDPathClosePathProc (void* clientData);
```

Description

Callback for [PDPathEnumMonitor](#). Called for every path closing operator.

Parameters

clientData	User-supplied data that was passed in the call to PDPathEnum .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDPathEnum](#)

PDPathCurveToProc

```
ACCB1 ASBool ACCB2 PDPathCurveToProc (ASFixedPoint* p1,  

ASFixedPoint* p2, ASFixedPoint* p3, void* clientData);
```

Description

Callback for [PDPathEnumMonitor](#). Called for every **c** operator.

Parameters

p1, p2, p3	The three points needed to specify the curve.
clientData	User-supplied data that was passed in the call to PDPathEnum .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

`PDExpT.h`

Related Methods

[PDPathEnum](#)



PDPathLineToProc

```
ACCB1 ASBool ACCB2 PDPathLineToProc (ASFixedPoint* p1,  
void* clientData);
```

Description

Callback for [PDPathEnumMonitor](#). Called for every **I** operator.

Parameters

p1	The one point needed to specify the line's ending point.
clientData	User-supplied data that was passed in the call to PDPathEnum .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDPathEnum](#)



PDPathMoveToProc

```
ACCB1 ASBool ACCB2 PDPathMoveToProc (ASFixedPoint* p1,
void* clientData);
```

Description

Callback for [PDPathEnumMonitor](#). Called for every **m** operator.

Parameters

p1	The one point needed to specify the location to move to.
clientData	User-supplied data that was passed in the call to PDPathEnum .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDPathEnum](#)



PDPathRectProc

```
ACCB1 ASBool ACCB2 PDPathRectProc (ASFixedPoint* p1,  

ASFixedPoint* p2, void* clientData);
```

Description

Callback for [PDPathEnumMonitor](#). Called for every **re** operator.

Parameters

p1, p2	The two points needed to specify the rectangle.
clientData	User-supplied data that was passed in the call to PDPathEnum .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDPathEnum](#)

PDPathVCurveToProc

```
ACCB1 ASBool ACCB2 PDPathVCurveToProc (ASFixedPoint* p1,  
ASFixedPoint* p2, void* clientData);
```

Description

Callback for [PDPathEnumMonitor](#). Called for every **v** operator.

Parameters

p1, p2	The two points needed to specify the curve.
clientData	User-supplied data that was passed in the call to PDPathEnum .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDPathEnum](#)

PDPathYCurveToProc

```
ACCB1 ASBool ACCB2 PDPathYCurveToProc (ASFixedPoint* p1,  
ASFixedPoint* p2, void* clientData);
```

Description

Callback for [PDPathEnumMonitor](#). Called for every **y** operator.

Parameters

p1, p2	The two points needed to specify the curve.
clientData	User-supplied data passed in the call to PDPathEnum .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDPathEnum](#)

PDResourceEnumColorSpaceProc

```
ACCB1 ASBool ACCB2 PDResourceEnumColorSpaceProc (char* name,  
CosObj colorSpace, void* clientData);
```

Description

Callback for [PDResourceEnumMonitor](#). It is called for color space resources.

Parameters

name	Color space name.
colorSpace	The name of the color space as it appears in the Resources dictionary.
clientData	User-supplied data that was passed in the call to PDFormEnumResources .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDFormEnumResources](#)



PDResourceEnumFontProc

```
ACCB1 ASBool ACCB2 PDResourceEnumFontProc (PDFont font,
char* name, void* clientData);
```

Description

Callback for [PDResourceEnumMonitor](#). Procedure called for font resources.

Parameters

font	The font.
name	The name of the font as it appears in the Resources dictionary.
clientData	User-supplied data that was passed in the call to PDFormEnumResources .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

`PDExpT.h`

Related Methods

[PDFormEnumResources](#)

PDResourceEnumProcSetProc

```
ACCB1 ASBool ACCB2 PDResourceEnumProcSetProc (char* name,  
void* clientData);
```

Description

Callback for [PDResourceEnumMonitor](#). Procedure called for **ProcSet** resources.

Parameters

name	The name of the ProcSet as it appears in the Resources dictionary.
clientData	User-supplied data that was passed in the call to PDFormEnumResources .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDFormEnumResources](#)

PDResourceEnumXObjectProc

```
ACCB1 ASBool ACCB2 PDResourceEnumXObjectProc  
(PDXObject xObject, char* name, void* clientData);
```

Description

Callback for [PDResourceEnumMonitor](#). Procedure called for **XObject** resources.

Parameters

xObject	The XObject .
name	The name of the XObject as it appears in the Resources dictionary.
clientData	User-supplied data that was passed in the call to PDFormEnumResources .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDFormEnumResources](#)



PDStringEnumProc

```
ACCB1 ASBool ACCB2 PDStringEnumProc (PDFont font,
char* string, ASInt32 stringLen, ASFixed delta,
void* clientData);
```

Description

Callback for [PDTextEnum](#). Called once for each string in a text object.

Parameters

font	The font used for string .
string	The string. This string may be converted using PDFontXlateToHost or PDFontXlateToUCS .
stringLen	The number of bytes in string .
delta	The difference, in thousandths of an EM, from the end of the previous string to the beginning of the current string. (An EM is a typographic unit of measurement equal to the size of a font. For example, in a 12-point font, an EM is 12 points.) See the description of the TJ operator in Section 5.3.2 in the PDF Reference .
clientData	User-supplied data that was passed in the call to PDTextEnum .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDTextEnum](#)



PDSysFontEnumProc

```
ACCB1 ASBool ACCB2 PDSysFontEnumProc (PDSysFont sysFont,  
void* clientData);
```

Description

Callback for [PDEnumSysFonts](#). It is called once for each system font.

Parameters

sysFont	The system font.
clientData	User-supplied data that was specified in the call to PDEnumSysFonts .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PSFExpT.h

Related Methods

[PDEnumSysFonts](#)

PDTextSelectEnumQuadProc

```
ACCB1 ASBool ACCB2 PDTextSelectEnumQuadProc (void* procObj,  
ASInt32 page, ASFixedQuad* quad);
```

Description

Callback for [PDTextSelectEnumQuads](#). Called once for each quad in a text selection.

Parameters

<code>procObj</code>	User-supplied data that was passed in the call to PDTextSelectEnumQuads .
<code>page</code>	The page on which the text selection is located.
<code>quad</code>	The quad being enumerated.

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

`PDExpT.h`

Related Methods

[PDTextSelectEnumQuads](#)

PDTextSelectEnumTextProc

```
ACCB1 ASBool ACCB2 PDTextSelectEnumTextProc (void* procObj,  
PDFont font, ASFixed size, PDColorValue color, char* text,  
ASInt32 textLen);
```

Description

Callback for [PDTextSelectEnumText](#) and [PDTextSelectEnumTextUCS](#). Called once for each text run (text in the same font, size, color, and on the same line) in a text selection.

Parameters

<code>procObj</code>	User-supplied data that was passed in the call to PDTextSelectEnumText or PDTextSelectEnumTextUCS .
<code>font</code>	The text's font.
<code>size</code>	The text's size, in points.
<code>color</code>	The text's color.
<code>text</code>	The text in the current run. NOTE: This string is not necessarily null-terminated.
<code>textLen</code>	The number of bytes in <code>text</code> .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

`PDExpT.h`

Related Methods

[PDTextSelectEnumText](#)
[PDTextSelectEnumTextUCS](#)

PDThumbCreationDrawThumbProc

```
ACCB1 void ACCB2 PDThumbCreationDrawThumbProc (PDThumb thumb,  
void* clientData);
```

Description

(Optional) Callback for [PDThumbCreationServer](#). Called after [PDThumbCreationGetThumbDataProc](#) and after a **PDThumb** has been created. Gives the server a chance to draw the thumbnail image in a status window. May be **NULL**.

Parameters

thumb	The thumbnail image to draw.
clientData	User-supplied data that was passed in the call to PDDocCreateThumbs .

Return Value

None

Header File

PDExpT.h

Related Callbacks

[PDThumbCreationGetThumbDataProc](#)

Related Methods

[PDDocCreateThumbs](#)

PDThumbCreationGetThumbDataProc

```
ACCB1 ASBool ACCB2 PDThumbCreationGetThumbDataProc
(PDPage page, ASFixed thumbScale, ASInt32 width,
ASInt32 height, void* thumbData, void* clientData);
```

Description

(Optional) Callback for [PDThumbCreationServer](#). Called for each page that does not currently contain a thumbnail image. May be **NULL**. If it is **NULL**, the thumbnail data is generated by the default thumbnail generator.

Parameters

page	The page for which to create a thumbnail image.
thumbScale	The scale to map from the page size to the thumbnail size—the thumbnail size is either 1/8 of the page size, or is limited to MAX_THUMBPAGE_WIDTH and MAX_THUMBPAGE_HEIGHT , whichever is smaller.
width	The width of the thumbnail image to create.
height	The height of the thumbnail image to create.
thumbData	A buffer into which the thumbnail data is copied. This buffer has the size: $\text{rowBytes} = (\text{width} * \text{bitsPerPixel} + 7) / 8;$ $\text{size} = \text{rowBytes} * \text{height};$ where bitsPerPixel is specified as numComponents x bitsPerComponent . numComponents is dependent upon the color space. For DeviceRGB , numComponents is 3. For an indexed color space, numComponents is 1.
clientData	User-supplied data that was passed in the call to PDDocCreateThumbs .

Return Value

true to continue thumbnail image creation, **false** to halt thumbnail image creation.

Header File

PDExpT.h

Related Methods

[PDDocCreateThumbs](#)

PDThumbCreationNotifyPageProc

```
ACCB1 ASBool ACCB2 PDThumbCreationNotifyPageProc  
(ASInt32 pageNum, void* clientData);
```

Description

(Optional) Callback for [PDThumbCreationServer](#). Called before processing each page. May be **NULL**.

Parameters

pageNum	The page for which to create a thumbnail image.
clientData	User-supplied data that was passed in the call to PDDocCreateThumbs .

Return Value

true to continue thumbnail image creation, **false** to halt thumbnail image creation.

Header File

PDExpT.h

Related Methods

[PDDocCreateThumbs](#)



PDWordProc

```
ACCB1 ASBool ACCB2 PDWordProc (PDWordFinder wObj,  
PDWord wInfo, ASInt32 pgNum, void* clientData);
```

Description

Callback for [PDWordFinderEnumWords](#). Called once for each word.

Parameters

wObj	The word finder.
wInfo	The current word in the enumeration.
pgNum	The page number on which wInfo is located.
clientData	User-supplied data that was passed in the call to PDWordFinderEnumWords .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDWordFinderEnumWords](#)

PDXObjectFilterEnumProc

```
ACCB1 ASBool ACCB2 PDXObjectFilterEnumProc (char* filter,  
CosObj decodeParms, void* clientData);
```

Description

Callback for [PDXObjectEnumFilters](#). Called once for each filter that has been applied to an **XObject**'s data.

Parameters

filter	The filter's name.
decodeParms	The dictionary Cos object containing the filter's decode parameters.
clientData	User-supplied data that was passed in the call to PDXObjectEnumFilters .

Return Value

true to continue enumeration, **false** to halt enumeration.

Header File

PDExpT.h

Related Methods

[PDXObjectEnumFilters](#)

PIExportHFTsProcType

ACCB1 [ASBool](#) ACCB2 [PIExportHFTsProcType](#) (void);

Description

(Optional) Callback for [PIHandshake](#). This handshaking function is called by the viewer during initialization to allow a client to export one or more HFTs to other clients.

This function should not do anything other than export HFTs. Clients should not change the user interface at this time, for instance; do that in the [PIInitProcType](#) callback.

If this callback returns **false**, the [PIUnloadProcType](#) callback is called. The client *must* remove and release *all* menu items and other user interface elements, HFTs, HFTServers, release any memory or any other resources allocated, and so on, in its [PIUnloadProcType](#) callback.

Parameters

None

Return Value

true if the HFT was exported successfully, **false** otherwise.

Header File

PIVersn.h

Related Callbacks

[PIHandshake](#)

[PIImportReplaceAndRegisterProcType](#)

PIHandshake

```
ACCB1 ASBool ACCB2 PIHandshake (ASInt32 handshakeVersion,  
void* handshakeData);
```

Description

This handshaking function is called by the viewer after a client is loaded into memory. The client should fill out **handshakeData** to tell the viewer its name and how to call it back to export an HFT, import an HFT, perform initialization, and perform any sort of cleanup needed when the client unloads.

This function should not do anything other than set up these callbacks. Clients should not change the user interface at this time, for instance; do that in the [PIInitProcType](#) callback.

Every client must provide this function.

Parameters

handshakeVersion	Version of the handshakeData structure.
handshakeData	Data indicating the client's name and lifecycle callbacks for initialization and termination. Corresponds to PIHandshakeData_V0200 .

Return Value

true if the handshake version number is valid, **false** if the handshake version number is unknown. If **false**, the client is not initialized, and any capability it would add is not available.

Header File

PICommon.h

Related Callbacks

[PIExportHFTsProcType](#)
[PIImportReplaceAndRegisterProcType](#)
[PIInitProcType](#)
[PIUnloadProcType](#)

PIImportReplaceAndRegisterProcType

ACCB1 [ASBool](#) ACCB2 [PIImportReplaceAndRegisterProcType](#) (void);

Description

(Optional) Callback for [PIHandshake](#). This handshaking function is called by the viewer during initialization to allow a client to import an HFT to another client, replace an API method, or register for a notification.

This is the *only* place that a client may replace a function. You may call utility functions, but do not attempt to use Cos, PDModel or AcroView-level methods here. Clients should not change the user interface at this time, for instance; do that in the [PIInitProcType](#) callback.

If this callback returns **false**, the [PIUnloadProcType](#) callback is called. The client *must* remove and release *all* menu items and other user interface elements, HFT's, HFTServers, release any memory or any other resources allocated, and so on, in its [PIUnloadProcType](#) callback.

Parameters

None

Return Value

true if the HFT was imported successfully, **false** otherwise.

Header File

PIVersn.h

Related Callbacks

[PIExportHFTsProcType](#)

[PIHandshake](#)



PIInitProcType

```
ACCB1 ASBool ACCB2 PIInitProcType (void);
```

Description

(Required) Callback for [PIHandshake](#). This handshaking function is called by the viewer during initialization to allow a client to do any sort of initialization it requires, such as adding user interface elements like menu items.

It is *not* safe to assume that all other clients have initialized at this point.

If you want to do something after *all* clients have been loaded and after the user has dismissed the open file dialog (if any), register for the notification

AVAppDidInitialize and perform the action in the callback you register. Or register for an idle proc with **AVAppRegisterIdleProc** and perform the action in the callback you register. An example of such a case is adding a menu item to a menu or submenu added by another client.

If this callback returns **false**, the [PIUnloadProcType](#) callback is called. The client *must* remove and release *all* menu items and other user interface elements, HFT's, HFTServers, release any memory or any other resources allocated, and so on, in its [PIUnloadProcType](#) callback.

Parameters

None

Return Value

true if the client initialized successfully, **false** otherwise.

Header File

PIVersn.h

Related Callbacks

[PIHandshake](#)

PIUnloadProcType

```
ACCB1 ASBool ACCB2 PIUnloadProcType (void);
```

Description

(Optional) Callback for [PIHandshake](#). This handshaking function is called by the viewer when the client unloads to allow it to perform any sort of cleanup needed. Use this routine to release any system resources you may have allocated.

This is called for every client when the viewer terminates or when any of the other [PIHandshake](#) callbacks return **false**.

The client *must* remove and release *all* menu items and other user interface elements, HFT's, HFTServers, release any memory or any other resources allocated, and so on, in its [PIUnloadProcType](#) callback.

Parameters

None

Return Value

true if the client finished its cleanup successfully, **false** otherwise.

Header File

PIVersn.h

Related Callbacks

[PIHandshake](#)

PMBeginOperationProc

```
ACCB1 void ACCB2 PMBeginOperationProc (void* clientData);
```

Description

Callback used in [ProgressMonitor](#). Initialize the progress monitor and displays it with a current value of zero. This method must be called first when the progress monitor is used.

Parameters

clientData	User-supplied data that was passed in the call to whatever API method required the progress monitor.

Return Value

None

Header File

ASExpT.h

Related Callbacks

[PMEndOperationProc](#)

PMEndOperationProc

```
ACCB1 void ACCB2 PMEndOperationProc (void* clientData);
```

Description

Callback used in [ProgressMonitor](#). Draws the progress monitor with its current value set to the progress monitor's duration (a full progress monitor), then removes the progress monitor from the display.

Parameters

clientData	User-supplied data that was passed in the call to whatever API method required the progress monitor.

Return Value

None

Header File

ASExpT.h

Related Callbacks

[PMBeginOperationProc](#)

PMGetCurrValueProc

ACCB1 [ASInt32](#) ACCB2 PMGetCurrValueProc (void* clientData);

Description

Callback used in [ProgressMonitor](#). Gets the progress monitor's duration, set by the most recent call the progress monitor's [PMSetCurrValueProc](#).

Parameters

clientData	User-supplied data that was passed in the call to whatever API method required the progress monitor.

Return Value

None

Header File

[ASExpT.h](#)

Related Callbacks

[PMSetCurrValueProc](#)

PMGetDurationProc

```
ACCB1 ASInt32 ACCB2 PMGetDurationProc (void* clientData);
```

Description

Callback used in [ProgressMonitor](#). Gets the progress monitor's duration, set by the most recent call the progress monitor's [PMSetDurationProc](#).

Parameters

clientData	User-supplied data that was passed in the call to whatever API method required the progress monitor.

Return Value

The progress monitor's maximum value.

Header File

`ASExpT.h`

Related Callbacks

[PMSetDurationProc](#)

PMSetCurrValueProc

```
ACCB1 void ACCB2 PMSetCurrValueProc (ASInt32 currValue,  
void* clientData);
```

Description

Callback used in [ProgressMonitor](#). Sets the current value of the progress monitor and updates the display. The allowed value ranges from 0 (empty) to the value passed to **setDuration**. For example, if the progress monitor's duration is 10, the current value must be between 0 and 10, inclusive.

Parameters

currValue	The progress monitor's current value.
clientData	User-supplied data that was passed in the call to whatever API method required the progress monitor.

Return Value

None

Header File

`ASExpT.h`

Related Callbacks

[PMGetCurrValueProc](#)



PMSetDurationProc

```
ACCB1 void ACCB2 PMSetDurationProc (ASInt32 duration,
void* clientData);
```

Description

Callback used in [ProgressMonitor](#). Sets the value that corresponds to a full progress monitor display. The progress monitor is subsequently filled in by setting its current value. This method must be called before you can set the progress monitor's current value.

Parameters

duration	The maximum value the progress monitor will be allowed to have.
clientData	User-supplied data that was passed in the call to whatever API method required the progress monitor.

Return Value

None

Header File

ASExpT.h

Related Callbacks

[PMGetDurationProc](#)



PMSetTextProc

```
ACCB1 void ACCB2 PMSetTextProc (AStext text,
void* clientData);
```

Description

Callback within [ASProgressMonitorRec](#) that sets the text string that is displayed by the progress monitor.

The built-in document progress monitor (see [AVAppGetDocProgressMonitor](#)) makes a copy of **text**. As such, it is the client's responsibility to destroy it.

Parameters

text	The string to display.
clientData	A pointer to the data associated with the progress monitor (which should be passed to you with the progress monitor).

Return Value

None

Header File

AExpT.h

Related Callbacks

None



Declarations

ASArraySize

```
typedef ASUns32 ASArraySize;
```

Description

An array size value for use in callback procedures.

Header File

`ASExpT.h`

Related Methods

[AVDocSelectionAcquireQuadsProc](#)
[CosObjOffsetProc](#)



ASBool

Description

2-byte unsigned short with two values—true (1) or false (0)

Header Files

CoreExpT.h

Related Methods

Numerous



ASByteCount

```
typedef ASUns32 ASByteCount;
```

Description

A byte count value for use in [ASProcStmRdExHandler](#) and [ASFileSysItemProps](#).

Header File

`ASExpT.h`

Related Methods

[ASUUIDGenFromName](#)

ASCabEntryRec

```
typedef struct _t_ASCabEntryRec {
    const char * keyName;
    ASCabValueType type;
    ASInt32 intVal;
    const void * ptrVal;
    double doubleVal;
} ASCabEntryRec;
```

Description

Data structure representing a cabinet entry. The first entry in each descriptor specifies the name of the key; the second field contains the type; the following fields contain the values. The entry list must end with a descriptor containing **NULL** for the key name. It can be used as shown below to construct an [ASCab](#):

```
ASCabEntryRec cabData[] = {{"key1", kASValueInteger, 1},
                            {"key2", kASValueInteger, -1},
                            {"key3", kASValueBool, false},
                            {NULL}};

ASCab CreateDefaultCab()
{
    return ASCabFromEntryList (cabData);
}
```

This example uses just three values for each record. However, more may be required—for example, for a double:

```
{"keyDouble", kASValueDouble, 0, NULL, doub}
```

For a string:

```
{"keyString", kASValueString, 0, (void*)string}
```

Header File

```
ASExtraExpT.h
```

Related Methods

```
ASCabFromEntryList
```

Members

keyName	The name of the key.

type	The supported ASCabValueTypes are: kASValueBool—intVal contains the value. kASValueInteger—intVal contains the value. kASValueAtom—intVal contains the value. kASValueDouble—doubleVal contains the value. kASValueString—ptrVal points to a null-terminated C string. kASValueText—ptrVal points to a null-terminated string containing script text, intVal specifies the ASScript code for the text. kASValueBinary—ptrVal points to the binary data, intVal specifies the size of the data. kASValueNull—Creates an entry with a NULL value. No other types are supported (specifically kASValueCabinet and kASValuePointer). You can build nested cabinets using the “key:key” syntax for the keyNames.
intVal	Contains the value for boolean, integer, and atom types, the ASScript code for text, and the size of binary data.
ptrVal	Contains a pointer to the value for string, text, and binary data types.
doubleVal	Contains a double numeric value.

ASCabMungeAction

```
typedef ASEnum16 ASCabMungeAction;
```

Description

A constant value that determines the actions to be taken when [ASCabMunge](#) is called. **keyCab** is the [ASCab](#) that provides the keys determining how **theCab** is to be changed. During an [ASCabMunge](#) operation the key cab will not be altered.

Header File

ASExtraExpT.h

Related Methods

[ASCabMunge](#)

Values

kASMungeRemove	Any keys in keyCab are removed from theCab and their values destroyed.
kASMungeRemoveUnknown	Any keys in theCab which are not also in keyCab are removed and their values destroyed.
kASMungeRemoveDefaults	Any keys in keyCab which are also in theCab and have the same value in theCab are removed and their values destroyed.
kASMungeRemoveBadValues	Any keys in theCab which are also in keyCab but have different values are removed and their values destroyed.
kASMungeCopy	All key-value pairs in keyCab are copied into theCab , possibly overwriting existing key-value pairs in theCab .
kASMungeReplace	Any keys in theCab which are also in keyCab have their values replaced with copies of the values in keyCab .
kASMungeCopyMissing	Any keys in keyCab which are not in theCab are copied over to theCab , along with the corresponding values.
kASMungeRemoveNulls	Any keys in keyCab with a type of kASValueNull are removed from theCab , along with their corresponding values.

ASCabValueType

```
typedef ASEnum16 ASCabValueType;
```

Description

[ASCabs](#) can be used to store arbitrary key-value pairs. The keys are always null-terminated strings containing only low-ASCII alphanumeric characters. These constants specify the various types of values.

Header File

`ASExtraExpT.h`

Related Methods

[ASCabFromEntryList](#)

[ASCabGetType](#)

Values

<code>kASValueBool</code>	An ASBool .
<code>kASValueInteger</code>	An ASInt32 .
<code>kASValueAtom</code>	An ASAtom .
<code>kASValueDouble</code>	A double-precision floating point number.
<code>kASValueString</code>	A null-terminated, un-encoded string.
<code>kASValueText</code>	An ASText object.
<code>kASValueBinary</code>	A binary blob of any size.
<code>kASValuePointer</code>	A pointer to something outside the cabinet.
<code>kASValueCabinet</code>	Another cabinet.
<code>kASValueNull</code>	Key exists but has no useful value—that is, a placeholder.
<code>kASValueUns</code>	An ASUns32 .
<code>kASValueUnknown</code>	An invalid type.



ASCalendarTimeSpan

```
typedef struct _t_ASCalendarTimeSpan{
    ASUns32 Year;
    ASUns32 Month;
    ASUns32 Day;
    ASUns32 Hour;
    ASUns32 Minute;
    ASUns32 Second;
} ASCalendarTimeSpanRec, *ASCalendarTimeSpan;
```

Description

Represents a calendar time span, used to calculate ambiguous time spans such as 1 year and 3 months. A calendar time span cannot be negative.

Header File

ASExpT.h

Related Methods

[ASCalendarTimeSpanAddWithBase](#)
[ASCalendarTimeSpanCompare](#)
[ASCalendarTimeSpanDiff](#)



ASConstCab

```
typedef const struct _t_ASCabinet *ASConstCab;
```

Description

An opaque object holding a constant cabinet ([ASCab](#)) structure.

Header File

ASExpT.h

Related Methods

numerous



ASConstText

```
typedef const struct _t_ASTextRec *ASConstText;
```

Description

An opaque object holding constant encoded text.

Header File

ASExpT.h

Related Methods

numerous



ASCoord

```
#if !defined(ACRO_SDK_LEVEL) || (ACRO_SDK_LEVEL < 2)
#pragma message("Warning: Using older Acrobat SDK. Define
ACRO_SDK_LEVEL to 0x00060000")
typedef ASInt16 ASCoord;
#else
typedef ASInt32 ASCoord;
```

Description

A coordinate for a point in device space, for use in mouse-click callbacks. Values are conditionally compiled as 16-bit or 32-bit integers, depending on the Acrobat version level.

Header File

`ASExpT.h`

Related Structures

[AVSDKDependentInteger](#)
[Quad](#)



ASCount

```
typedef ASUns32 ASCount;
```

Description

A numeric count value.

Header File

`ASExpT.h`

Related Methods

[ASGetSecs](#)
[ASIsValidUTF8](#)
[AVAppCreateIconBundle6](#)
[AVDocGetNthPageView](#)
[AVDocGetNumPageViews](#)

ASCryptStm

ASCryptStmRec

```
typedef struct _t_ASCryptStmRec {
    ASInt32 count;
    char *currentPointer;
    char *basePointer;
    ASUns32 modeFlag;
    ASCryptStmProcs *procs;
    ASStm baseStm;
    ASInt32 nBytesWanted;
    void *clientData;
} ASCryptStmRec;
```

Description

An [ASStm](#) object cover used for a cryptographic filter’s stream callbacks.

Members

count	The number of characters remaining in the buffer.
currentPointer	The next character to get or put.
basePointer	The base of the buffer (if any).
modeFlag	Flag to indicate mode. ● ASCryptStmModeRead 0x0001 (decryption) ● ASCryptStmModeWrite 0x0002 (encryption) ● ASCryptStmModeEOF 0x0004 ● ASCryptStmModeError 0x0008
procs	Handlers for security stream access.
baseStm	The base ASStm object.
nBytesWanted	The number of bytes being requested for decryption.
clientData	A pointer to arbitrary user-defined data.

Related Callbacks

[PDCryptFilterStreamProc](#)

Header File

ASExpT.h

ASCryptStmProcs

```
typedef struct {  
    ASCryptStmFilBufProc EmptyBuff;  
    ASCryptStmFlsBufProc FullBuff;  
    ASCryptStmUnGetcProc UnGetCh;  
    ASCryptStmFFlushProc FlushBuff;  
    ASCryptStmFCloseProc Close;  
    ASCryptStmFResetProc Reset;  
    ASCryptStmFPutEOFProc PutEOF;  
} ASCryptStmProcs;
```

Description

A structure containing callback procedures for an [ASCryptStm](#). The first call to a procedure of the [PDCryptFilterStreamProc](#) type must fill out an [ASCryptStmRec](#) structure with pointers to callback routines for various types of stream access.

Leave unused callbacks uninitialized.

Members

EmptyBuff	Optional. Called by getc when buffer is empty. Used only for decryption.
FullBuff	Optional. Called by putc when buffer is full. Used only for encryption.
UnGetCh	Optional. Backs up the input stream. Used only for decryption.
FlushBuff	Required. Flushes a dirty buffer if necessary.
Close	Required. Closes the stream.
Reset	Optional. Resets the stream, discarding any buffered data. Used only for encryption.
PutEOF	Optional. Puts an EOF marker to the stream. Used only for encryption.

Header File

ASExpT.h

ASDateTimeFormat

```
enum {
    kASTimeNone=0,
    kASTimePDF,
    kASTimeUniversal,
    kASTimeUniversalH,
    kASTimeUTC_ASN1,
    kASTimeGENERALIZED_ASN1
};
typedef ASEnum8 ASDateTimeFormat;
```

Description

A constant indicating a string format for representing a date-time.

Members

kASTimePDF	PDF date format, as defined in the PDF Reference .
kASTimeUniversal kASTimeUniversalH	Slight variations of the time format expressed in ISO 8601.
kASTimeUTC_ASN1	UTC ASN1 format.
kASTimeGENERALIZED_ASN1	ASN1 format.

Examples

```
kASTimePDF = "D:20000911121643-08'00'"
kASTimeUniversal = "2000.09.11 13:30:20 -08'00' "
kASTimeUniversalH = "2000-09-11 13:30:20 -08'00' "
kASTimeUTC_ASN1 = "000911203020Z"
kASTimeGENERALIZED_ASN1 = "20000911203020Z"
```

Header File

```
ASExpT.h
```

Related Methods

```
ASDateGetTimeString
ASDateSetTimeFromString
```



AEnum8

Description

1-byte enumeration with values from 0 to 127, used in data structures.

Header Files

CoreExpT.h



AEnum16

Description

2-byte enumeration with values from 0 to 32,767, used in data structures.

Header Files

CoreExpT.h



ASharedLib.h

```
typedef ASInt32 ASharedLibError;
```

Description

An error code value for use in [ASharedLib](#) and [ASharedLibSys](#) methods and callbacks.
Introduced in Acrobat 6.0.

Header File

`ASharedLib.h`

Related Methods

numerous

ASFile Flags

Description

Flags that describe file transfers. Set by external file systems.

Header File

ASExpT.h

Related Methods

[ASFileSysOpenFile](#)

Values

kASFileSlowTransfer	Set if the file's data transfer rate is generally slow, for example, because it is on a floppy disk or being accessed via modem.
kASFileSlowConnect	Initiating each access to the file is slow, for example, because the file is served by an HTTP server that spawns a new process for each request.
kASFileUseMRead	Use multi-read commands to access the file.
kASFileDialUp	<i>(New for Acrobat 5.0)</i> Set if media/access is a dial up connection. This flag is only fully implemented on Windows. On the Macintosh, this flag is always conservatively set to true .

ASFileMode Flags

Description

Flags that describe file modes.

Header File

ASExpT.h

Related Methods

[ASFileRead](#)

[ASFileSetMode](#)

Values

kASFileModeDoNotYieldIfBytesNotReady	If set, ASFileRead does not yield if bytes are not ready (which raises the fileErrBytesNotReady exception).
kASFileModeDisableExplicitMReadRequests	If set, the file will be read all at once regardless of multiple read requests.
kASFileRaiseIfBytesNotReady	If set, ASFileRead will raise fileErrBytesNotReady when trying to read from a file with a cache for which the requested bytes are not yet present.
kASFileNoRequestIfBytesNotReady	If set, no read requests are issued if bytes are not ready (that is, in the cache).



ASFileOffset

```
typedef ASUns32 ASFileOffset;
```

Description

A file offset value for use in callback procedures.

Header File

`ASExpT.h`

Related Methods

[CosDocEnumEOFsProc](#)

ASFileMode

```
typedef ASUns32 ASFileMode;
```

Description

File access modes used to specify how a file can be used when it is open. Not all modes can be specified individually; **ASFILE_CREATE** can be used only in conjunction with **ASFILE_READ** or **ASFILE_WRITE**. In addition, it is acceptable to specify **ASFILE_READ** and **ASFILE_WRITE** together, by OR-ing the two constants.

ASFILE_SERIAL and **ASFILE_LOCAL** (present only in version 3.0 or later) are hints that help the Acrobat viewer optimize access to the file; they must be OR-ed with one or more of the other constants.

Values

ASFILE_READ	Open the file for reading.
ASFILE_WRITE	Open the file for writing.
ASFILE_CREATE	Create the file if it does not exist.
ASFILE_SERIAL	A hint indicating that the file will be accessed sequentially.
ASFILE_LOCAL	A hint indicating that a local copy of the file will be needed.

Header File

ASExpT.h

Related Methods

[ASFileSysOpenFile](#)

[ASFileReopen](#)



ASFilePos

```
typedef ASUns32 ASFilePos;
```

Description

A file position value for use in callback procedures.

Header File

`ASExpT.h`

Related Methods

[CosObjOffsetProc](#)



ASFileStatus Flags

Description

Constant status values returned by [ASFileSysGetStatusProc](#).

Header File

ASExpT.h

Related Methods

[ASFileRead](#)

Values

Mode	Description
kASFileOkay	The MDFile is in a valid state.
kASFileTerminating	The MDFile is being closed, for example, because the file is displayed in a Web browser's window and the user canceled downloading.

ASFileSysItemProps

ASFileSysItemPropsRec

```
typedef struct _t_ASFileSysItemProps {
    ASSize_t size;
    ASBool isThere;
    ASFileSysItemType type;
    ASBool isHidden;
    ASBool isReadOnly;
    ASBool creationDateKnown;
    ASTimeRec creationDate;
    ASBool modDateKnown;
    ASTimeRec modDate;
    ASByteCount fileSize;
    ASByteCount fileSizeHigh;
    ASTCount folderSize;
    ASUns32 creatorCode;
    ASUns32 typeCode;
} ASFileSysItemPropsRec, *ASFileSysItemProps;
```

Description

A list of properties for the object referenced by an [ASPathName](#). Used in [ASFileSysGetItemProps](#) and the folder enumeration routines.

Header File

ASExpT.h

Related Methods

[ASFileSysGetItemProps](#)
[ASFileSysFirstFolderItem](#)
[ASFileSysNextFolderItem](#)

Members

size	Size of the data structure. Must be set to sizeof (ASFileSysItemPropsRec) .
isThere	true if the object exists. If false none of the following fields are valid.
type	One of the ASFileSysItemTypes .
isHidden	true if the object's hidden bit is set.
isReadOnly	true if the object is read only.



creationDateKnown	true if the file system could determine the creation date of the object.
creationDate	The creation date of the object. Valid only if creationDateKnown is true .
modDateKnown	true if the file system could determine the last modification date of the object.
modDate	The modification date of the object. Valid only if modDateKnown is true .
fileSize, fileSizeHigh	If type is kASFileSysFile , these two fields hold the size of the file in bytes as a 64-bit integer.
folderSize	If type is kASFileSysFolder , this field specifies how many items are in the folder. If this value is -1 the file system was unable to easily determine the number of objects. You will need to explicitly enumerate the objects to determine how many are in the folder.
creatorCode	The Mac OS creator code for the file. For non-Mac OS file systems, this will be zero.
typeCode	The Mac OS type code for the file. For non-Mac OS file systems, this will be zero.



ASFileSysItemType

```
typedef ASEnum16 ASFileSysItemType;
```

Description

Constants that categorize an object associated with an [ASPathName](#).

Header File

ASExpT.h

Related Callbacks

[ASFileSysGetItemPropsProc](#)
[ASFileSysFirstFolderItemProc](#)
[ASFileSysNextFolderItemProc](#)

Related Methods

[ASFileSysGetItemProps](#)
[ASFileSysFirstFolderItem](#)
[ASFileSysNextFolderItem](#)

Values

kASFileSysFile	Object is associated with a file.
kASFileSysFolder	Object is associated with a folder.
kASFileSysUnknown	Object type is unknown (-1)



ASFileSys

ASFileSysRec

```
typedef struct _t_ASFileSysRec *ASFileSys;
typedef struct _t_ASFileSysRec {
    ASSize_t size;
    ASFileSysOpenProc open;
    ASFileSysCloseProc close;
    ASFileSysFlushProc flush;
    ASFileSysSetPosProc setpos;
    ASFileSysGetPosProc getpos;
    ASFileSysSetEofProc seteof;
    ASFileSysGetEofProc geteof;
    ASFileSysReadProc read;
    ASFileSysWriteProc write;
    ASFileSysRemoveProc remove;
    ASFileSysRenameProc rename;
    ASFileSysIsSameFileProc isSameFile;
    ASFileSysGetNameProc getName;
    ASFileSysGetTempPathNameProc getTempPathName;
    ASFileSysCopyPathNameProc copyPathName;
    ASFileSysDiPathFromPathProc diPathFromPath;
    ASFileSysPathFromDIPathProc pathFromDIPath;
    ASFileSysDisposePathNameProc disposePathName;
    ASFileSysGetFileSysNameProc getFileSysName;
    ASFileSysGetStorageFreeSpaceProc getStorageFreeSpace;
    ASFileSysFlushVolumeProc flushVolume;
    /* The following are present in version 3.0 or later */
    ASFileSysGetFileFlags getFileFlags;
    ASFileSysAsyncReadProc readAsync;
    ASFileSysAsyncWriteProc writeAsync;
    ASFileSysAsyncAbortProc abortAsync;
    ASFileSysYieldProc yield;
    ASFileSysMReadRequestProc mreadRequest;
    ASFileSysGetStatusProc getStatus;
    ASFileSysCreatePathNameProc createPathName;
    ASFileSysAcquireFileSysPathProc acquireFileSysPath;
    ASFileSysClearOutstandingMReadsProc
        clearOutstandingMReads;
    /* The following are present in version 5.0 or later */
    ASFileSysGetItemPropsProc getItemProps;
    ASFileSysFirstFolderItemProc firstFolderItem;
    ASFileSysNextFolderItemProc nextFolderItem;
    ASFileSysDestroyFolderIteratorProc destroyFolderIterator;
    ASFileSysSetModeProc setFileMode; /* do not use */
    ASFileSysURLFromPathProc urlFromPath;
    ASFileSysGetParentProc getParent;
}
```

```

ASFileSysCreateFolderProc createFolder;
ASFileSysRemoveFolderProc removeFolder;
ASFileSysDisplayStringFromPathProc displayStringFromPath;
ASFileSysSetTypeAndCreatorProc setTypeAndCreator;
ASFileSysGetTypeAndCreatorProc getTypeAndCreator;
/* present in version 6.0 or later *.
ASFileSysGetItemPropsAsCabProc getItemPropsAsCab;
ASFileSysCanPerformOpOnItemProc canPerformOpOnItem;
ASFileSysPerformOpOnItemProc performOpOnItem;
ASFileSysAcquirePlatformPathProc acquirePlatformPath;
ASFileSysReleasePlatformPathProc releasePlatformPath;
ASFileSysGetNameAsASTextProc getNameAsASText;
ASFileSysDisplayASTextFromPathProc displayASTextFromPath;
ASFileSysRangeArrivedProc rangeArrived;
ASFileSysCanSetEofProc canSetEof;
ASFileSysDIPathFromPathExProc diPathFromPathEx;
ASFileSysPathFromDIPathExProc pathFromDIPathEx;
} ASFileSysRec;

```

Description

Data structure containing callbacks that implement a file system.

Header File

ASExpT.h

Related Methods

Numerous. See [ASFileSys](#) object.

Members

size	Size of the data structure. Must be set to sizeof (ASFileSysRec) .

ASFixed

ASFixedP

```
typedef ASInt32 ASFixed, *ASFixedP;
```

Description

The **ASFixed** type is a 32-bit quantity representing a rational number with the high (low on little-endian machines) 16 bits representing the number's mantissa and the low (high) 16 bits representing the fractional part. The definition is platform-dependent.

ASFixedP is a pointer to an **ASFixed**.

Addition, subtraction, and negation with **ASFixed** types can be done with + and -, unless you care about overflow. (Overflow in **ASFixed**-value operations is indicated by the values **fixedPositiveInfinity** and **fixedNegativeInfinity**.)

Header Files

ASExpT.h

Related Methods

[ASFixedDiv](#)
[ASFixedMatrixConcat](#)
[ASFixedMatrixInvert](#)
[ASFixedMatrixTransform](#)
[ASFixedMatrixTransformRect](#)
[ASFixedMul](#)
[ASFixedToString](#)



ASFixedMatrix

ASFixedMatrixP

```
typedef struct _t_FixedMatrix {  
    ASFixed a;  
    ASFixed b;  
    ASFixed c;  
    ASFixed d;  
    ASFixed h;  
    ASFixed v;  
} ASFixedMatrix, *ASFixedMatrixP;
```

Description

Matrix containing fixed numbers.

Header File

ASExpT.h

Related Methods

Numerous



ASFixedPoint

ASFixedPointP

```
typedef struct _t_FixedPoint {  
    ASFixed h;  
    ASFixed v;  
} ASFixedPoint, *ASFixedPointP;
```

Description

Point (in two-dimensional space) represented by two fixed numbers.

Header File

ASExpT.h

Related Methods

Numerous



ASFixedQuad

ASFixedQuadP

```
typedef struct _t_ASFixedQuad {  
    ASFixedPoint tl, tr, bl, br;  
} ASFixedQuad, *ASFixedQuadP;
```

Description

Quadrilateral represented by four fixed points (one at each corner). In the Acrobat viewer, a quadrilateral differs from a rectangle in that the latter must always have horizontal and vertical sides, and opposite sides must be parallel.

Header File

PDExpT.h

Related Methods

Numerous



ASFixedRect

ASFixedRectP

```
typedef struct _t_ASFixedRect {  
    ASFixed left;  
    ASFixed top;  
    ASFixed right;  
    ASFixed bottom;  
} ASFixedRect, *ASFixedRectP;
```

Description

A rectangle represented by the coordinates of its four sides. In the Acrobat viewer, a rectangle differs from a quadrilateral in that the former must always have horizontal and vertical sides, and opposite sides must be parallel.

Header File

ASExpT.h

Related Methods

Numerous



ASFlagBits

```
typedef ASUns32 ASFlagBits;
```

Description

A flag-bits value for use .

Header File

`ASExpT.h`

Related Methods

[ASFileSetMode](#)
[CosDocCreate](#)
[CosDocSaveToFile](#)
[CosDocSaveWithParams](#)
[HFTReplaceEntry](#)
[HFTReplaceEntryEx](#)
[PDAnnotInfo](#)

Related Callbacks

[ASFileSysGetFileFlags](#)
[ASFileSysGetStatusProc](#)
[PDAnnotHandlerGetAnnotInfoFlagsProc](#)

ASFolderIterator

```
typedef struct _t_ASFolderIterator* ASFolderIterator;
```

Description

An opaque object used to iterate through the contents of a folder.

[ASFileSysFirstFolderItem](#) returns the first item in the folder along with an **ASFolderIterator** object for iterating through the rest of the items in the folder. Call [ASFileSysNextFolderItem](#) with this object to return the next object in the folder until the routine returns **false**. To discard the **ASFolderIterator** object, call [ASFileSysDestroyFolderIterator](#).

Header File

ASExpT.h

Related Callbacks

[ASFileSysFirstFolderItemProc](#)
[ASFileSysNextFolderItemProc](#)
[ASFileSysDestroyFolderIteratorProc](#)

Related Methods

[ASFileSysFirstFolderItem](#)
[ASFileSysNextFolderItem](#)
[ASFileSysDestroyFolderIterator](#)



ASHostEncoding

```
typedef ASInt32 AHostEncoding;
```

Description

Integer specifying the host encoding for text. On the Mac OS, it is a script code. On Windows, it is a CHARSET id. On UNIX, Acrobat currently only supports English, so the only valid **ASHostEncoding** is 0 (Roman). See [ASScript](#).

Header File

`ASExpT.h`

Related Methods

Numerous



ASInt8

Description

1-byte signed char

Header Files

CoreExpT.h

Related Methods

Numerous



ASInt16

Description

2-byte signed short numeric value.

Header Files

CoreExpT.h

Related Methods

Numerous



ASInt32

Description

4-byte signed long numeric value.

Header Files

CoreExpT.h

Related Methods

Numerous

ASIORequest

```
typedef struct _t_ASIORequestRec {
    ASMDFile mdFile;
    void* ptr;
    ASTFilePos offset;
    ASArraySize count;
    ASArraySize totalBytesCompleted;
    ASErrorCode pError;
    void* clientData;
    ASIODoneProc IODoneProc;
    void* IODoneProcData;
} ASIORequestRec, *ASIORequest;
```

Description

Data structure representing an I/O request.

Header File

ASExpT.h

Related Callbacks

ASFileSysAsyncReadProc
ASFileSysAsyncWriteProc
ASIODoneProc

Members

mdFile	The MDFile corresponding to the ASFile this request is for.
ptr	Pointer to data to write to or read from mdFile .
offset	Offset (specified in bytes) into mdFile of the first byte to read or write.
count	Number of bytes to read/write. Must be filled in before IODoneProc is called. If zero, the read was either terminated or did not complete.
totalBytesCompleted	Number of bytes actually read or written.
pError	Error code. This code is filled by the ASFileSys before IODoneProc is called. If nonzero, the read was either terminated or did not complete.
clientData	User-supplied data that the ASFileSys can use for anything it wishes.

IODoneProc	User-supplied callback to call by the ASFileSys when the operation has completed. If non- NULL , it points to a procedure that <i>must</i> be called either when the request has terminated due to error or other condition, or when all of the bytes have been received for this request. NOTE: This callback may be called at interrupt time.
IODoneProcData	User-supplied that is available for IODoneProc to use.



ASMDFile

```
#define MDFile ASMDFile
typedef void *ASMDFile;
```

Description

NOTE: NOTE: **ASMDFile** replaces **MDFile**. **MDFile** is an obsolete name for this data type for backward compatibility.

An **MDFile** is an opaque representation of a file instance for a particular file system. File system implementors may choose any convenient representation for an **MDFile**; in particular, file systems need not worry about **MDFile**-space conflicts; the **ASFile** object exported by the common implementation is guaranteed to be unique across all open files, and the common implementation maps calls to **ASFile** methods to calls to **ASFileSystem** callbacks with the corresponding **MDFile**.

Header File

ASExpT.h

Methods affected by this name change

[ASFileFromMDFile](#)
[ASFileGetMDFile](#)

Structures affected by this name change

[ASIORequest](#)

Callbacks affected by this name change

[ASFileSysAsyncAbortProc](#)
[ASFileSysGetFileFlags](#)
[ASFileSysYieldProc](#)
[ASFileSysMReadRequestProc](#)
[ASFileSysClearOutstandingMReadsProc](#)
[ASFileSysGetStatusProc](#)
[ASFileSysOpenProc](#)
[ASFileSysCloseProc](#)
[ASFileSysFlushProc](#)
[ASFileSysSetPosProc](#)
[ASFileSysGetPosProc](#)
[ASFileSysSetEofProc](#)
[ASFileSysGetEofProc](#)
[ASFileSysReadProc](#)
[ASFileSysWriteProc](#)
[ASFileSysRenameProc](#)
[ASFileSysIsSameFileProc](#)



ASPathName

A file system-specific named location for a particular type of device. Uses the [ASFileSys](#) structure pointers for callback. An [ASPathName](#) is specific to a given [ASFileSys](#).

Related Methods

- [ASFileAcquirePathName](#)
- [ASFileSysAcquireParent](#)
- [ASFileSysCreatePathName](#)
- [ASFileSysPathFromDIPath](#)
- [ASPathFromPlatformPath](#)
- [PDFFileSpecAcquireASPath](#)
- [ASFileSysReleasePath](#)
- [ASFileSysDIPathFromPath](#)

ASPlatformPrinterSpec

ASPlatformPrinterSpecRec

```

/* In Mac OS */
typedef struct _t_ASPlatformPrinterSpec
*ASPlatformPrinterSpec;
typedef struct _t_ASPlatformPrinterSpec {
    ASSize_t size;
    void* cGrafPtr;
    short hRes, vRes;
} ASPlatformPrinterSpecRec;

/* In UNIX */
typedef struct _t_ASPlatformPrinterSpec
*ASPlatformPrinterSpec;
typedef struct _t_ASPlatformPrinterSpec {
    ASSize_t size;
    char* printerName;
    ASUns8* baseAddr;
    ASUns32 rowBytes;
    ASUns32 depth;
    AVRect32 bounds;
} ASPlatformPrinterSpecRec;

/* In Windows */
#define kPrinterSpecNameLen 64 /* room for 32 Unicode chars */
typedef struct _t_ASPlatformPrinterSpec
*ASPlatformPrinterSpec;
typedef struct _t_ASPlatformPrinterSpec {
    ASSize_t size;
    char driverName[kPrinterSpecNameLen];
    char printerName[kPrinterSpecNameLen];
    char portName[kPrinterSpecNameLen];
    ASBool createMetaFile;
    char metaFileName[260];
    ASInt32 win16Hdc;
    ASInt32 hRes;
    ASInt32 vRes;
    ASInt32 colorDepth;
    ASBool isPostScript;
} ASPlatformPrinterSpecRec;

```

Description

Data structure representing a platform specification for a printer. Used in [AVDocPrintParams](#).

Header File

AVExpT.h

Related Methods

[AVDocPrintPagesWithParams](#)

Members

For all platforms	
size	Size of the data structure. Must be set to <code>sizeof (ASPlatformPrinterSpecRec)</code> .
In Mac OS	
cGrafPtr	Port to print to.
hRes, vRes	Best known resolution of current printer.
In UNIX	
printerName	Print command, such as "lp -dMyPrinter -n4". If printerName is NULL , a default print command is used. The Acrobat viewer's built-in default is "lp" on most UNIX systems, and "lpr" on SunOS. This should print to the system's default printer. Some UNIX systems also look at the environment variable LPDEST or PRINTER . See the documentation for your platform to determine whether this is the case.
baseAddr	Currently unused.
rowBytes	Currently unused.
depth	Currently unused.
bounds	Currently unused.
In Windows	
The following items should be provided for full, non-interactive printing.	
driverName	See <code>Windows.h</code> <code>DEVNAMES</code> for a description of these fields.

In Windows	
printerName	Name of the printer. For example, "HPPCL," "HP LaserJet 4," or "LPT1."
portName	Port to print to.
For embedded printing, createMetaFile , metaFileName , win16Hdc , hRes , vRes , colorDepth , and isPostScript must be provided.	
createMetaFile	Must be true if Windows 32-bit platforms; optional for Windows 16-bit platforms.
metaFileName	Pathname for the metafile. Only required if createMetaFile is true .
win16Hdc	Device context for 16- or 32-bit platforms.
hRes	Horizontal resolution of printer; 300 dpi, for example.
vRes	Vertical resolution of printer; 300 dpi, for example.
colorDepth	Color depth of device; typically 1, 8, or 24. This determines the depth of images created for the printer. You may specify 24 when printing to a monochrome printer. The driver is expected to convert to the printer depth. Not used if isPostScript is true .
isPostScript	Set to true if printing to a PostScript printer.



ASPortRef

```
/* ASPortRef */
#if MAC_PLATFORM
typedef CGrafPtr ASPortRef;
#elif WIN_PLATFORM
typedef void* ASPortRef;
#elif UNIX_PLATFORM
typedef void* ASPortRef;
#endif
```

Description

Provides access to a port. **ASPortRef** is the same as a Windows HDC.

Header File

ASExpT.h

ASProcStmRdExHandler

```
typedef struct _s_ASProcStmRdExHandler
{
    ASByteCount size; /* sizeof(ASProcStmRdExHandlerRec) */
    ASStmProc readProc;
    ASProcStmDestroyProc destroyProc;
} ASProcStmRdExHandlerRec, *ASProcStmRdExHandler;
```

Description

For use by [ASProcStmRdOpenEx](#). The supplied handlers are called when the client of the stream attempts to read data from it, and when the client closes it.

Header File

`ASExpT.h`

Related Methods

[ASProcStmRdOpenEx](#)

ASProgressMonitor

ASProgressMonitorRec

```
typedef struct _t_ProgressMonitor {
    ASSize_t size;
    PMBeginOperationProc beginOperation;
    PMEndOperationProc endOperation;
    PMSetDurationProc setDuration;
    PMSetCurrValueProc setCurrValue;
    PMGetDurationProc getDuration;
    PMGetCurrValueProc getCurrValue;
    PMSetTextProc setText;
} ASProgressMonitorRec, *ASProgressMonitor;
```

Description

This type replaces the [ProgressMonitorRec](#).

Header File

AVExpT.h

Methods Affected by this Change

[AVCommandGetProgressMonitor](#)
[AVAppGetDocProgressMonitor](#)
[PDDocImportCosDocNotes](#)
[PDDocExportNotes](#)
[PDDocImportNotes](#)
[PDDocCreateThumbs](#)
[PDDocDeleteThumbs](#)
[PDDocSave](#)
[PDDocDeletePages](#)
[PDDocInsertPages](#)
[PDDocReplacePages](#)
[PDDocEnumFonts](#)

Structs Affected by this Change

[ProgressMonitorRec](#)
[CosDocSaveParamsRec](#)
[PDDocSaveParamsRec](#)
[PDDocCopyParamsRec](#)



ASReportType

```
typedef ASEnum16 ASReportType;
```

Description

Constants used in an [ASReportProc](#) to indicate what kind of information is reported.

Header File

ASExtraExpT.h

Related Callbacks

[ASReportProc](#)

Values

kASReportNote	A note.
kASReportWarning	A warning.
kASReportError	An error.

ASScript

```
typedef ASInt32 ASScript;
enum {
    kASRomanScript,
    kASJapaneseScript,
    kASTraditionalChineseScript,
    kASKoreanScript,
    kASArabicScript,
    kASHebrewScript,
    kASGreekScript,
    kASCyrillicScript,
    kASRightLeftScript,
    kASDevanagariScript,
    kASGurmukhiScript,
    kASGujaratiScript,
    kASOriyaScript,
    kASBengaliScript,
    kASTamilScript,
    kASTeluguScript,
    kASKannadaScript,
    kASMalayalamScript,
    kASSinhaleseScript,
    kASBurmeseScript,
    kASKhmerScript,
    kASThaiScript,
    kASLaotianScript,
    kASGeorgianScript,
    kASArmenianScript,
    kASSimplifiedChineseScript,
    kASTibetanScript,
    kASMongolianScript,
    kASGeezScript,
    kASEastEuropeanRomanScript,
    kASVietnameseScript,
    kASExtendedArabicScript,
    kASEUnicodeScript,
    kASDontKnowScript = -1
};
```

Description

An enumeration of writing scripts. Not all of these scripts are supported on all platforms.

**Header File**

AExpT.h

Related Methods

Numerous



ASize_t

Description

4-byte canonical type for sizes of objects in bytes (as in `size_t`).

Header Files

`CoreExpT.h`

Related Methods

Numerous



ASTArraySize

```
typedef ASInt32 ASTArraySize;
```

Description

A numeric array size value for use in AS and Cos-level I/O methods and data structures.

Header File

`ASExpT.h`

Related Methods

numerous

Related Callbacks

[ASFileCompletionProc](#)
[ASFileSysGetNameProc](#)
[ASFileSysMReadRequestProc](#)
[ASStmProc](#)



ASTCount

```
typedef ASInt32 ASTCount;
```

Description

A numeric count value for use in stream methods.

Header File

`ASExpT.h`

Related Methods

[ASIsValidUTF8](#)
[ASStmFlush](#)
[ASStmRead](#)
[ASStmWrite](#)
[CosCopyStringValue](#)
[CosDocGetID](#)
[CosStreamPos](#)
[CosStringValue](#)
[CosStringValueSafe](#)
[HFTNew](#)

Related Callbacks

[ASStmProc](#)

ASTextFilterType

```
typedef AEnum16 ATextFilterType;
```

Description

Constants that specify filter types used to modify text objects.

Header File

`ASExtraExpT.h`

Related Methods

[ASTextFilter](#)

Values

<code>kASTextFilterIdentity</code>	Does nothing.
<code>kASTextFilterLineEndings</code>	Normalizes line endings. Equivalent to ASTextNormalizeEndOfLine .
<code>kASTextFilterUpperCase</code>	Makes all text upper case.
<code>kASTextFilterXXXDebug</code>	Changes any ASText to “XXX” (for debugging).
<code>kASTextFilterUpperCaseDebug</code>	Makes all text except scanf format strings upper case.
<code>kASTextFilterRemoveAmpersands</code>	Removes stand-alone ampersands, changes '&&' to '&'.
<code>kASTextFilterRsvd1</code>	Reserved, do not use.
<code>kASTextFilterUnknown</code>	An invalid filter type.



ASTFilePos

```
typedef ASInt32 ASTFilePos;
```

Description

A numeric count value for use in I/O methods and data structures.

Header File

`ASExpT.h`

Related Methods

[ASFileGetEOF](#)
[ASFileGetPos](#)
[ASFilePushData](#)
[ASFileSetPos](#)

Related Callbacks

[ASFileCompletionProc](#)
[ASFileSysMReadRequestProc](#)



ASTimeRec

ASTimeRecP

```
typedef struct _t_ASTimeRec {  
    ASInt16 year;  
    ASInt16 month;  
    ASInt16 date;  
    ASInt16 hour;  
    ASInt16 minute;  
    ASInt16 second;  
    ASInt16 millisecond;  
    ASInt16 day;  
    ASInt16 gmtoffset;  
} ASTimeRec, *ASTimeRecP;
```

Description

Time/Date structure.

The **millisecond** field is currently unused.

Header File

ASExpT.h

Related Methods

[ASDateGetLocalTime](#)
[ASDateGetUTCTime](#)
[ASDateSetTimeFromRec](#)
[PDAnnotGetDate](#)
[PDAnnotSetDate](#)



ASTVersion

```
typedef ASUns32 ASTVersion;
```

Description

Cryptographic version number.

Header File

`ASExpT.h`

Related Methods

[CosCryptGetVersion](#)

[CosDecryptGetMaxKeyBytes](#)

[CosEncryptGetMaxKeyBytes](#)



ASUnicodeChar

```
typedef ASUns32 ASUnicodeChar;
```

Description

Used to pass individual Unicode characters around. Large enough to hold any arbitrary Unicode character (at least 21 bits wide).

Header File

```
AExpT.h
```

ASUnicodeFormat

```
typedef AEnum16 ASUnicodeFormat;
```

Description

Constants that describe the various Unicode formats you can pour into and read out of an [ASText](#) object.

Header File

```
AExtraExpT.h
```

Related Methods

```
ASIsValidUTF8  
ASTextFromUnicode  
ASTextFromSizedUnicode  
ASTextSetUnicode  
ASTextSetSizedUnicode  
ASTextGetUnicodeCopy
```

Values

kUTF16BigEndian	Always returns the bytes in big-endian order.
kUTF16HostEndian	Returns the bytes in the host’s native endian, whatever is natural for an ASUns32 .
kUTF8	Endian neutral.



ASUns8

Description

1-byte unsigned char.

Header Files

CoreExpT.h

Related Methods

Numerous



ASUns16

Description

2-byte unsigned short numeric value.

Header Files

CoreExpT.h

Related Methods

Numerous



ASUns32

Description

4-byte unsigned long numeric value.

Header Files

CoreExpT.h

Related Methods

Numerous



ASUTF16Val

```
typedef ASUns16 ASUTF16Val;
```

Description

Holds a single 16-bit value from a UTF-16 encoded Unicode string. Typically used to point to the beginning of an UTF-16 string, for example:

```
ASUTF16Val *utf16String = ...
```

This datatype is not large enough to hold any arbitrary Unicode character. Use [ASUnicodeChar](#) to pass individual Unicode characters around.

Header File

```
AExpT.h
```

Related Methods

[ASTextGetUnicode](#)

[ASTextGetUnicodeCopy](#)



ASUUID

```
typedef struct
{
    ASUns32 timeLow;
    ASUns16 timeMid;
    ASUns16 timeHiAndVersion;
    ASUns8 clockSeqHiAndReserved;
    ASUns8 clockSeqLow;
    ASUns8 node[6];
} ASUUID;
```

Description

A structure representing a universal unique identifier (UUID) for the current user or the current session.

Header File

ASExpT.h

Related Methods

[AVAppGetUUID](#)
[ASUUIDFromCString](#)
[ASUUIDGenFromHash](#)
[ASUUIDGenFromName](#)
[ASUUIDGenUnique](#)
[ASUUIDToCString](#)



ASUUIDMaxStringLen

```
#define ASUUIDMaxStringLen 40
```

Description

A constant for the maximum string length of a unique identifier (UUID).

Header File

```
ASExpT.h
```

Related Methods

[ASUUIDToCString](#)



ASVersion

```
typedef ASUns32 ASVersion;
```

Description

An HFT version number.

Header File

`ASExpT.h`

Related Methods

[ASExtensionMgrGetHFT](#)
[HFTServerProvideHFTProc](#)
[HFTGetVersion](#)



ASWindowRef

```
/* ASWindowRef */
#if MAC_PLATFORM
typedef WindowRef ASWindowRef;
#elif WIN_PLATFORM
typedef void* ASWindowRef;
#elif UNIX_PLATFORM
typedef void* ASWindowRef;
#endif
```

Description

A platform dependent window handle corresponding to a **WindowPtr** in Mac OS, an **HWND** in Windows and a **Widget** in Unix.

Header File

ASExpT.h

Related Methods

[AVSweetPeaProcessADMEvent](#)

AVAccessColorPolicy

typedef [ASEnum8](#) AVAccessColorPolicy;

Description

Constants that specify the manner in which the background and text colors are chosen when viewing a document.

Header File

AVExpT.h

Related Methods

[AVAppSetPreference](#)

Values

kAVAccessUseDocumentColors	Use the colors specified within the document.
kAVAccessUseSystemColors	Use the colors specified by the OS preferences.
kAVAccessUsePreferenceColors	Use the colors specified by the Acrobat preferences.

AVActionContext

AVActionContextRec

```
typedef struct _AVActionContextRec {
    ASSize_t size;
    CosObj co;
    ASAtom asaType;
    ASAtom asaKey;
} AVActionContextRec, *AVActionContext;
```

Description

This structure gives the action handler some context in terms of its execution. It specifies the "parent" object, which initiated the action, and the trigger type of the action. Trigger names should correspond to the key used in the **AA** dictionary of the file.

Header File

AVExpT.h

Related Callbacks

[AVActionPerformExProc](#)

Members

size	Size of this record. Must be set to <code>sizeof(AVActionContextRec)</code> .
co	Object that is performing the action—for example, bookmark or annotation.
asaType	Type of the object, using these <code>defines</code> : #define AVTRIGGERTYPE_DOC ASAtomFromString("Doc") #define AVTRIGGERTYPE_DEST ASAtomFromString("Dest") #define AVTRIGGERTYPE_PAGE ASAtomFromString("Page") #define AVTRIGGERTYPE_LINK ASAtomFromString("Link") #define AVTRIGGERTYPE_ANNOT ASAtomFromString("Annot") #define AVTRIGGERTYPE_BOOKMARK ASAtomFromString("Bookmark")
asaKey	Trigger name for the object, using these <code>defines</code> : #define AVTRIGGER_MOUSEUP ASAtomFromString("Mouse Up") #define AVTRIGGER_OPEN ASAtomFromString("Open") #define AVTRIGGER_CLOSE ASAtomFromString("Close")

AVActionHandlerProcs

```
typedef struct _t_AVActionHandlerProcs {
    ASSize_t size;
    AVActionPerformProc Perform;
    AVActionDoPropertiesProc DoProperties;
    AVActionFillActionDictProc FillActionDict;
    AVActionGetInstructionsProc GetInstructions;
    AVActionGetButtonTextProc GetButtonText;
    AVActionGetStringOneTextProc GetStringOneText;
    AVActionGetStringTwoTextProc GetStringTwoText;
    /* New for Acrobat 3.01 */
    AVActionCopyProc Copy;
    /* New for Acrobat 6.0 */
    AVActionPerformExProc PerformEx;
    AVActionDoPropertiesExProc DoPropertiesEx;
    AVActionGetDetailsProc GetDetails;
} AVActionHandlerProcsRec, *AVActionHandlerProcs;
```

Description

Data structure containing callbacks that implement an action handler. The callbacks implement the action handler functions. For example, display user interface text, request the action's properties from the user, perform the action.

Header File

AVExpT.h

Related Methods

[AVAppRegisterActionHandler](#)
[AVActionHandlerGetProcs](#)
[AVDocCopyAction](#)

Members

size	Size of the data structure. Must be set to sizeof (AVActionHandlerProcsRec) .

AVAdjustCursorParams

```
typedef struct _t_AVAdjustCursorParams
{
    ASSize_t size;
    AVDevCoord xHit;
    AVDevCoord yHit;
    ASAtom toolType;
} AVAdjustCursorParamsRec, *AVAdjustCursorParams;
```

Description

Parameters that describe where and how a cursor event occurred, for the use of cursor handling callback procedures.

Header File

```
AVExpT.h
```

Related Methods

```
AVAnnotHandlerAdjustCursorExProc
```

Members

size	The size of this structure.
xHit	The x-coordinate of the cursor.
yHit	The y-coordinate of the cursor.
toolType	The tool type that received the cursor event.

AAlertButtonInfo

```
typedef struct AAlertButtonInfo {
    ASBool show;
    ASText title;
} AAlertButtonInfo;
```

Description

Data structure containing information about a button used in an Alert dialog.

Header File

AVExpT.h

Related Methods

[AAlertWithParams](#)

Members

show	Pass true to show the button.
title	If non- NULL this text is used as the button caption, otherwise the default is used. The default values for button1 , button2 and button3 are "OK", "Cancel", and "" respectively.

AAlertCheckBoxInfo

```
typedef struct AAlertCheckBoxInfo {
    ASBool show;
    ASText title;
    ASBool value;
} AAlertCheckBoxInfo;
```

Description

Data structure containing information about a checkbox used in an Alert dialog.

Header File

AVExpT.h

Related Methods

[AAlertWithParams](#)

Members

show	Pass true to show the button.
title	If non- NULL this text is used as the checkbox caption, otherwise the default is used. The default value for the checkbox is "Do not show this message again".
value	Pass true to initially check the box. The chosen value is returned in this parameter.



AAlert Icons

Description

Standard icons used in alert boxes.

Header File

AVExpT.h

Related Methods

[AAlert](#)
[AAlertConfirm](#)
[AAlertNote](#)

Icon Type	Icon Displayed (Windows)	Icon Displayed (Macintosh)
ALERT_NOICON	None.	None.
ALERT_CAUTION		
ALERT_NOTE		
ALERT_QUESTION		
ALERT_STOP		

- NOTE:** The **AAlert** method displays the **ALERT_QUESTION** icon when the **ALERT_NOTE** constant is specified. Because of this, **AAlertNote** displays the **ALERT_QUESTION** icon.
- NOTE:** The **ALERT_QUESTION** value is not recognized by **AAlert()**. It can only be used with the **AAlertWithParams** method.



AAlertParams

AAlertParamsRec

```
typedef struct _t_AAlertParams {
    ASSize_t size;
    AVDoc parentDoc;
    AText windowTitle;
    ASInt32 iconType;
    AText message;
    ASBool beep;
    AAlertButtonInfo button1;
    AAlertButtonInfo button2;
    AAlertButtonInfo button3;
    AAlertCheckBoxInfo checkbox;
    /* new in Acrobat 6.0 */
    AAlertType type;
} AAlertParamsRec, *AAlertParams;
```

Description

Data structure containing information about the format of an Alert dialog.



Members

size	The size of the structure. Must be set to sizeof (AAlertParamsRec) .
parentDoc	The AVDoc that is the modal parent of the alert dialog. May be NULL . The parentDoc can be NULL , in which case the alert dialog is parented off of the currently active doc, if there is one. The parentDoc is a no-op on the Mac.
windowTitle	The title of the dialog. May be NULL , in which case the default title, "Adobe Acrobat", is used.
iconType	The icon to display. Must be one of the AAlert Icons .
message	The message to display.
beep	Set to true to trigger a beep when the dialog is shown.
button1, button2, button3	AAlertButtonInfo structures describing the dialog's buttons. Any or all may be NULL . If all are NULL , the dialog is shown with an OK button.
checkBox	AAlertCheckBoxInfo structure describing the dialog's checkbox. May be NULL .
type	The type of the alert.

Header File

AVExpT.h

Related Methods

[AAlertWithParams](#)



AAlertType

```
enum {
    kAAlertTypeNone = 0,
    kAAlertTypeOk,
    kAAlertTypeOkCancel,
    kAAlertTypeYesNo,
    kAAlertTypeYesNoCancel
};
typedef AEnum16 AAlertType;
```

Description

Constant values for alert types, used in [AAlertParams](#).

Header File

AVExpT.h

Related Methods

[AAlertWithParams](#)

AVAnnot Appearance Flags

```
#define kAVAppearanceDrawNow 0x01
#define kAVAppearanceNoZoom 0x02
#define kAVAppearanceNoRotate 0x04
#define kAVAppearanceSmooth 0x08
```

Description

An OR of these bit flags is passed in the flags parameter of the [AVAnnotHandlerGetAppearanceProc](#) callback. These flags determine how the annotation's appearance is drawn on return from the callback.

Header File

AVExpT.h

Related Callbacks

[AVAnnotHandlerGetAppearanceProc](#)

Values

kAVAppearanceDrawNow	The appearance is not drawn at intervals.
kAVAppearanceNoZoom	The appearance is not zoomed. Initialized to the current NoZoom value on entry to the callback.
kAVAppearanceNoRotate	The appearance is not rotated. Initialized to the current NoRotate value on entry to the callback
kAVAppearanceSmooth	The appearance is smoothed (anti-aliased) when drawn.

AVAnnotHandler

AVAnnotHandlerRec

```
typedef struct _t_AVAnnotHandler *AVAnnotHandler;
typedef struct _t_AVAnnotHandler {
    ASSize_t size;
    ASUns32 flags;
    AVAnnotHandlerDoClickProc DoClick; /*deprecated in Acrobat 6*/
    AVAnnotHandlerAdjustCursorProc AdjustCursor; /*deprecated in
Acrobat 6*/
    AVAnnotHandlerPtInAnnotViewBBoxProc PtInAnnotViewBBox;
    AVAnnotHandlerGetAnnotViewBBoxProc GetAnnotViewBBox;
    AVAnnotHandlerNotifyAnnotRemovedFromSelectionProc
        NotifyAnnotRemovedFromSelection
    AVAnnotHandlerNotifyAnnotAddedToSelectionProc
        NotifyAnnotAddedToSelection;
    AVAnnotHandlerDrawProc Draw; /*deprecated in Acrobat 6*/
    AVAnnotHandlerNewProc New;
    AVAnnotHandlerGetTypeProc GetType;
    AVAnnotHandlerNotifyDestroyProc NotifyDestroy;
    AVAnnotHandlerDoPropertiesProc DoProperties;
    AVAnnotHandlerDoKeyDownProc DoKeyDown;
    AVAnnotHandlerGetLayerProc GetLayer; /*deprecated in Acrobat
6*/
    /* New callbacks in Acrobat 3.0 */
    AVAnnotHandlerCursorEnterProc CursorEnter;
    AVAnnotHandlerCursorExitProc CursorExit;
    /* New callbacks in Acrobat 3.01 */
    AVAnnotHandlerCopyProc Copy;
    /* New callbacks in Acrobat 4.0 */
    AVAnnotHandlerDoClickProc DoRightClick; /*deprecated in
Acrobat 6*/
    AVAnnotHandlerGetInfoProc GetInfo;
    AVAnnotHandlerDeleteInfoProc DeleteInfo;
    /* New callbacks in Acrobat 5.0 */
    AVAnnotHandlerCanPerformOpProc CanPerformOp;
    AVAnnotHandlerPerformOpProc PerformOp;
    AVAnnotHandlerDoKeyDownExProc DoKeyDownEx;
    AVAnnotHandlerDrawExProc DrawEx;
    /* New callbacks in Acrobat 6.0 */
    AVAnnotHandlerGetFlagsProc GetFlags;
    AVAnnotHandlerGetAppearanceProc GetAppearance;
    AVAnnotHandlerAppearanceDrawingProc BeginAppearanceDrawing;
    AVAnnotHandlerAppearanceDrawingProc FinishAppearanceDrawing;
    AVAnnotHandlerAppearanceDrawingProc CancelAppearanceDrawing;
    AVAnnotHandlerDoPropertiesExProc DoPropertiesEx;
    AVAnnotHandlerGetLayerExProc GetLayerEx;
```

```

AVAnnotHandlerDoClickExProc DoClickEx;
AVAnnotHandlerDoClickExProc DoRightClickEx;
AVAnnotHandlerAdjustCursorExProc AdjustCursorEx;
} AVAnnotHandlerRec;

```

Description

Data structure containing callbacks that implement an annotation handler. The callbacks implement the annotation handler functions. For example, draw the annotation, highlight the annotation when it is selected, and the data specifies properties of the annotation (for example, text selection behavior).

Appearance drawing is new in Acrobat 6.0. If the handler implements the **GetAppearance** callback, the new drawing style is used. If the **GetAppearance** callback returns **false** or is not implemented, the viewer calls the drawing procedure (**DrawEx** if present, otherwise **Draw**) to draw the annotation using Acrobat 5.0-style drawing.

For the new style of appearance drawing, the viewer calls the handler's **BeginAppearanceDrawing** callback, then draws the appearance object. If the drawing operation completes without errors, it call the the handler's **BeginAppearanceDrawing** callback; if an error occurs, or the operation is canceled, it calls the **CancelAppearanceDrawing** callback. This can happen if, for example, the user switches to another page before the drawing is completed.

The handler must not destroy or modify the appearance object while it is being drawn. It can do so only after the **EndAppearanceDrawing** or **CancelAppearanceDrawing** callback is exercised.

NOTE: Types of numeric values in callbacks have changed in Acrobat 6.0.

Header File

AVEXPT.h

Related Callbacks

```

AVAnnotHandlerEnumProc
PDAnnotHandlerDeleteAnnotInfoProc
PDAnnotHandlerGetAnnotInfoFlagsProc
PDAnnotHandlerGetAnnotInfoProc
PDAnnotHandlerGetTypeProc

```

Related Methods

```

AVAppRegisterAnnotHandler
AVAppGetAnnotHandlerByName
AVDocCopyAnnot

```

Members

size	Size of the data structure. Must be set to sizeof (AVAnnotHandlerRec) .



flags	Bit flags that affect the annotation's behavior. The flags may be OR-ed together. Possible flag values are: ANNOT_CLIP_TEXT_SELECTION ● If this flag is set, selecting text in a document never highlights the text that is underneath an open annotation pop-up window. ● If this flag is not set, selecting text highlights all the text, including the text underneath an annotation pop-up window. ANNOT_WANTS_SHIFT_KEY ● Acrobat's normal behavior is that an annotation does not receive a mouse click event when the Shift key is held down. If this flag is set, the annotation does receive the mouse click.

AVAnnotHandlerInfo

AVAnnotHandlerInfoRec

```
typedef struct _t_AVAnnotHandlerInfoRec {  
    ASSize_t size;  
    unsigned char* cName;  
    void* vBitmap;  
} AVAnnotHandlerInfoRec, *AVAnnotHandlerInfo;
```

Description

Structure used to describe information for a particular annotation type.

Header File

AVExpT.h

Related Callbacks

[AVAnnotHandlerDeleteInfoProc](#)
[AVAnnotHandlerGetInfoProc](#)

Related Methods

[AVAnnotHandlerGetInfo](#)

Members

size	Size of the data structure. Must be set to sizeof (AVAnnotHandlerInfoRec) .
cName	User interface name of annotation type in the host encoding.
vBitmap	Platform-dependent bitmap used as the annotation icon. If NULL , the annotation manager uses the unknown annotation icon for the annotation.

AVAnnotOp

```
typedef AEnum16 AVAnnotOp;
enum {
    kAVAnnotAcceptFocus,
    kAVAnnotLostFocus,
    kAVAnnotDefaultAction,
    kAVAnnotShowMenu,
    kAVAnnotSuspendFocus,
    kAVAnnotResumeFocus,
    kAVAnnotDoPageHasOpened,
    kAVAnnotDoPageHasClosed,
    kAVAnnotDoPageIsInView,
    kAVAnnotDoPageIsNotInView
};
```

Description

Constants that specify operations that you can ask an annotation to react to.

Header File

AVExpT.h

Related Methods

[AVPageViewFocusAnnotPerformOp](#)

Values

kAVAnnotAcceptFocus	Accept input focus.
kAVAnnotLostFocus	Lose the input focus.
kAVAnnotDefaultAction	The user presses Enter while the annotation has focus.
kAVAnnotShowMenu	Show a context-menu for the annot.
kAVAnnotSuspendFocus	Focus is temporarily lost; you may get it back.
kAVAnnotResumeFocus	Focus has been restored.
kAVAnnotDoPageHasOpened	The page containing the annotation has become the current page.
kAVAnnotDoPageHasClosed	The page containing the annotation is no longer the current page.
kAVAnnotDoPageIsInView	The page containing the annotation has become visible to the user.



kAVAnnotDoPageIsNotInView	The page containing the annotation is no longer visible to the user.

AVAnnotOpData

AVAnnotOpDataRec

```
typedef struct _t_AVAnnotOpData {
    ASSize_t size;
    AVDevCoord x;
    AVDevCoord y;
    void * clientData;
    /* new in Acrobat 6 */
    AVAnnotOpReason reason;
    ASInt32 pageNum;
} AVAnnotOpDataRec, *AVAnnotOpData;
```

Description

Additional information passed to the annotation when performing an operation. For some operations a **NULL** will be passed; in others a pointer to an **AVAnnotOpData** structure will be passed.

NOTE: The coordinate numeric type has changed in Acrobat 6.0.

Header File

AVExpT.h

Related Methods

[AVPageViewSetFocusAnnot](#)

Members

size	Set by Acrobat to the size of this record.
x	If the operation is kAVAnnotShowMenu , provides the default location of the menu in AV device coordinates.
y	
clientData	Used by Forms and Annots to determine when an annotation is getting focus via a mouse click.
reason	The user action that triggered this operation request.
pageNum	The page number of the page containing the annotation.

AVAnnotOpReason

```
typedef enum {
    kAVAnnotUnknown = 0,
    kAVAnnotClick,
    kAVAnnotTab
} AVAnnotOpReason;
```

Description

Constants that specify how an annotation operation request was triggered.

Header File

```
AVExpT.h
```

Related Methods

```
AVPageViewFocusAnnotPerformOp
```

Values

kAVAnnotClick	Operation triggered by a mouse click.
kAVAnnotTab	Operation triggered by a Tab key press.



AVAppLanguageFormat

```
enum {  
    kAVAppLanguage_RFC1766,  
    kAVAppLanguage_LCID,  
    kAVAppLanguage_ISO4Char /* en-us */  
}; typedef ASEnum16 AVAppLanguageFormat;
```

Description

Constants that specify language format values for use in [AVAppLanguageParams](#).

Header File

AVExpT.h

Related Methods

[AVAppGetLanguageWithParams](#)

AVAppLanguageParams

```
typedef struct _AVAppLanguageParamsRec {
    ASSType\_t size;
    AVAppLanguageFormat kLangFormat;
    char szAVAppLanguage[kMaxLanguageNameLen];
} AVAppLanguageParamsRec, *AVAppLanguageParams;
```

Description

Data structure containing language format information, in which to return the language in use for an application.

Header File

```
AVExpT.h
```

Related Methods

```
AVAppGetLanguageWithParams
```

Members

size	Size of the data structure. Must be set to sizeof (AVAppLanguageParamsRec) .
kLangFormat	The format in which to specify the language.
szAVAppLanguage	The returned language value in the specified format. For details of language values, see Language Codes .



AVAppUUIDType

```
typedef ASEnum8 AVAppUUIDType;
```

Description

Constants that specify universal unique identifier (UUID) type values. The UUID is represented by an [ASUUID](#) structure.

Header File

```
AVExpT.h
```

Related Methods

```
AVAppGetUUID
```

Values

<code>kAVAppUserUUID</code>	The UUID for this user for this installation.
<code>kAVAppSessionUUID</code>	The UUID for the currently executing session.



AVArraySize

```
typedef ASUns16 AVArraySize;
```

Description

An array size value for AV methods.

Header File

AVExpT.h

Related Methods

[AVAppChooseFolderDialog](#)
[AVAppOpenDialog](#)
[AVAppSaveDialog](#)
[AVExtensionAcquireInfo](#)
[AVExtensionGetNumPlugIns](#)

Related Structures

[AVFileFilterRec](#)
[AVConversionFromPDFHandler](#)
[AVConversionToPDFHandler](#)
[AVOpenSaveDialogParams](#)

AVAuxDataHandler

AVAuxDataHandlerRec

```
typedef struct _t_AVAuxDataHandler {
    ASSize_t size;
    AVAuxDataPerformProc PerformProc;
} AVAuxDataHandlerRec, *AVAuxDataHandler;
```

Description

Data structure containing callbacks and data representing an auxiliary data handler.

Header File

AVExpT.h

Related Methods

- [AVDocSendAuxData](#)
- [AVHasAuxDataHandler](#)
- [AVRegisterAuxDataHandler](#)

Members

size	Size of the data structure. Must be set to <code>sizeof(AVAuxDataHandlerRec)</code> .
PerformProc	Called with auxiliary data when a client calls AVDocSendAuxData . This proc should perform whatever action it needs to do for the auxiliary data.



AVBatchContext

```
typedef struct _t_AVBatchContext *AVBatchContext;
```

Description

Placeholder only. Not currently implemented.

AVClickParams

```
typedef struct _t_AVClickParams
{
    ASSize_t size;
    AVDevCoord xHit;
    AVDevCoord yHit;
    AVFlagBits16 flags;
    AVTCount clickNo;
    ASAtom toolType;
} AVClickParamsRec, *AVClickParams;
```

Description

Parameters that describe where and how a mouse click occurred, for the use of click handling callback procedures.

Header File

AVExpT.h

Related Methods

[AVAnnotHandlerDoClickExProc](#)

Members

size	The size of this structure.
xHit	The x-coordinate of the mouse click.
yHit	The y-coordinate of the mouse click.
flags	Indicates which modifier keys are pressed. Must be an OR of the Modifier Keys values.
clickNo	1 if this is a single click, 2 if a double click, 3 if triple click.
toolType	The tool type that received the click event.



AVColorForcing

```
enum {  
    kAVForceColorNone = 0,  
    kAVForceColorToGrayscale,  
    kAVForceColorToMonochrome  
};  
typedef ASEnum16 AVColorForcing;
```

Description

Constants that specify color forcing values for [AVDocPrintOverrideData](#), used in [AVDocPrintParams](#).

Use **AVForceColor** when a print driver incorrectly handles conversion of color to grayscale or monochrome. This overrides the Windows **printerspec colorDepth** value. As of 1/1/02, only WinFax is known to need this setting since it incorrectly converts all colors to black.

Header File

AVExpT.h

Related Methods

[AVDocPrintPagesWithParams](#)

AVCommandHandler

AVCommandHandlerRec

```
typedef AVCommandHandlerRec *AVCommandHandler;
typedef struct _t_AVCommandHandler {
    ASSize_t size;
    AVCmdHandlerInitProc Initialize;
    AVCmdHandlerTermProc Terminate;
    AVRegisterCommandsProc RegisterCommands;
    AVCommandCreatedProc Created;
    AVCommandDestroyProc Destroy;
    AVCommandSetProc SetParams;
    AVCommandGetProc GetParams;
    AVCommandGetProc GetProps;
    AVCommandShowDialogProc ShowDialog;
    AVCommandWorkProc Work;
    AVCommandCancelProc Cancel;
    AVCommandResetProc Reset;
    AVCommandPreflightSequenceProc PreflightSequence;
    AVCommandPreflightFileProc PreflightFile;
    AVCommandPostflightFileProc PostflightFile;
    AVCommandPostflightSequenceProc PostflightSequence;
} AVCommandHandlerRec;
```

Description

A set of callbacks that perform the actions required of a command.

Header File

AVExpT.h

Related Methods

[AVCommandNew](#)

Members

size	Size of the data structure. Must be set to <code>sizeof (AVCommandHandlerRec)</code> .
Initialize	Called once for each handler registered.
Terminate	Called once for each handler registered when Acrobat shuts down. Called before clients are unloaded

RegisterCommands	The application maintains a global list of commands that you can choose from when building your own command. During the initialization sequence, all registered command handlers are asked to build and register all the commands that the handler wants included in the global command list.
Created	Called after a command is created. The command handler can establish default parameters, and so forth, for the newly created command.
Destroy	Called before a command is destroyed. The command handler should free any memory allocated by the command, and so forth.
SetParams	Called to set the command's parameters.
GetParams	Called to retrieve the command's parameters.
GetProps	Called to retrieve command properties.
ShowDialog	Displays the command's parameter setting dialog and allows the user to alter the parameters.
Work	Do some work. If the command does not finish the work, return kAVCommandWorking . If the command finishes the work, return kAVCommandDone . If the user cancels the operation, return kAVCommandCanceled . If an error occurs, return kAVCommandInError . In most cases the work procedure will be called until you return kAVCommandDone , but in some cases, you may be called on to Cancel or Reset before the work is done.
Cancel	Stop working and pretend you reached a point where all work was done.
Reset	Stop working, clear any errors, and try to get yourself back into a Ready state. For many commands this is equivalent to canceling.
PreflightSequence	See sequence description in callback reference.
PreflightFile	See sequence description in callback reference.
PostflightFile	See sequence description in callback reference.
PostflightSequence	See sequence description in callback reference.



AVCommandStatus

```
typedef AEnum16 AVCommandStatus;
```

Description

Constant status codes that can be returned by various [AVCommand](#) methods.

Header File

AVExpT.h

Related Methods

[AVCommandGetStatus](#)

Values

<code>kAVCommandReady</code>	Not working, but ready to.
<code>kAVCommandWorking</code>	Still working.
<code>kAVCommandDone</code>	Done working.
<code>kAVCommandCanceled</code>	Canceled.
<code>kAVCommandInError</code>	In error.

AVCommandUIPolicy

```
typedef enum {
    kAVCommandUIInteractive = 0,
    kAVCommandUISemiInteractive = 1,
    kAVCommandUIErrorsOnly = 2,
    kAVCommandUISilent = 3
} AVCommandUIPolicy;
```

Description

Constants that specify how the command is expected to interact with the user.

Header File

AVExpT.h

Related Methods

[AVCommandGetUIPolicy](#)

Values

kAVCommandUIInteractive	Fully interactive. Gather parameters based on currently active document at the time the dialog goes up. Display errors and warnings.
kAVCommandUISemiInteractive	Interactive but under the control of the sequencing user interface. When showing a dialog, use the parameters passed in rather than the parameters gathered from the document.
kAVCommandUIErrorsOnly	Display errors but no other dialogs.
kAVCommandUISilent	Never put up a dialog.

AVConversionClientData

```
typedef struct _t_AVConversionClientData  
*AVConversionClientData;
```

Description

The user-defined data that is supplied when a conversion handler is registered with the conversion server. This data is provided to all **AVConversionHandler** callbacks.

Header File

AVExpT.h

Related Callbacks

[AVConversionDefaultSettingsProc](#)
[AVConversionParamDescProc](#)
[AVConversionSettingsDialogProc](#)
[AVConversionConvertFromPDFProc](#)
[AVConversionConvertToPDFProc](#)

AVConversionEnumProcData

```
typedef struct _t_AVConversionEnumProcData  
*AVConversionEnumProcData;
```

Description

The user-defined data that is supplied to either of the conversion handler enumeration routines.

Header File

AVExpT.h

Related Callbacks

[AVConversionFromPDFEnumProc](#)
[AVConversionToPDFEnumProc](#)

Related Methods

[AVConversionEnumFromPDFConverters](#)
[AVConversionEnumToPDFConverters](#)

AVConversionFlags

```
typedef ASUns32 AVConversionFlags;
```

Description

Constant flags that can be passed to a PDF conversion callback to allow non-default behavior.

Header File

AVExpT.h

Related Callbacks

[AVConversionConvertFromPDFProc](#)
[AVConversionConvertStreamToPDFProc](#)

Related Methods

[AVConversion](#) methods

Values

kAVConversionNoFlags	No flags.
kAVConversionAsyncOkay	Asynchronous conversion is allowed.
kAVConversionPopSettingsDialog	Pop the settings dialog, if one is provided for this conversion handler.
kAVConversionInteractive	Interactive mode. Indicates converter can pop additional dialogs if necessary.
kAVConversionDontOverwrite	Do not overwrite the existing files EXCEPT for the source file. This flag is only used in batch.

AVConversionFromPDFHandler

AVConversionFromPDFHandlerRec

```
typedef struct _t_AVConversionFromPDFHandler {
    AVFileFilterRec convFilter;
    ASSize_t size;
    char uniqueID[256];
    ASBool canDoSync;
    ASInt16 priority;
    AVConversionDefaultSettingsProc defaultSettings;
    AVConversionParamDescProc parameterDescription;
    AVConversionSettingsDialogProc settingsDialog;
    AVConversionConvertFromPDFProc convert;
    AVConversionClientData clientData;
    /*new in Acrobat 6.0*.
    AVConversionMimeTypeString *streamMimeTypes;
    AVArraySize numStreamMimeTypes;
    AVConversionConvertStreamToPDFProc convertStream;
    AVConversionConvertStreamFromStructNodeProc
        convertStructNode;
} AVConversionFromPDFHandlerRec, *AVConversionFromPDFHandler;
```

Description

Data structure containing callbacks that implement the “FromPDF” handler’s functionality and data that describes the handler’s conversion capabilities.

Header File

AVExpT.h

Related Callbacks

[AVConversionFromPDFEnumProc](#)

Related Methods

[AVAppRegisterFromPDFHandler](#)

Members

convFilter	An AVFileFilterRec that describes the types of files that this filter can convert. See description of AVFileFilterRec and AVFileDescRec for more details.

size	Size of the data structure. Must be set to sizeof (AVConversionFromPDFHandlerRec) .
uniqueID	Unique identifier for the conversion handler. Should be of the form com.companyname.productname.type . See PDF_FILEFILTERREC_UNIQUEID and FDF_FILEFILTERREC_UNIQUEID in AVExpT.h .
canDoSync	true if the converter can perform synchronous conversion, false if the converter only does asynchronous conversion. This capability is required for the handler to be accessible from the batch framework.
priority	An integer that indicates where converter should be registered in list of FromPDF handlers. A high priority number indicates that this converter is checked earlier than other converters. All Acrobat 5.0 converters have a priority of 0.
defaultSettings	AVConversionDefaultSettingsProc that is called when the handler is registered with the conversion server. Can be NULL .
parameterDescription	AVConversionParamDescProc that is called when a parameter description of this handler is requested. Can be NULL .
settingsDialog	AVConversionSettingsDialogProc that is called when the batch framework or the open dialog requests a settings dialog for this handler. Can be NULL .
convert	AVConversionConvertFromPDFProc that is called to perform the conversion operation.
clientData	Provided to all AVConversion callbacks.
streamMimeTypes	A string containing Mime types that can be handled by the convertStream callback.
numStreamMimeTypes	The number of Mime types in streamMimeTypes .
convertStream	The stream-conversion handler procedure.
convertStructNode	The structure-node-to-stream conversion handler procedure.

AVConversionMimeTypeString

```
typedef char AVConversionMimeTypeString[256];
```

Description

A Mime-type string for PDF conversion handlers.

Header File

AVExpT.h

Related Structures

[AVConversionFromPDFHandler](#)
[AVConversionToPDFHandler](#)

Related Methods

[AVAppRegisterFromPDFHandler](#)
[AVAppRegisterToPDFHandler](#)

AVConversionStatus

```
typedef ASEnum16 AVConversionStatus;
```

Description

Constants that describe the status of a conversion operation.

Header File

AVExpT.h

Related Callbacks

[ASFileSysGetItemPropsProc](#)
[ASFileSysFirstFolderItemProc](#)
[ASFileSysNextFolderItemProc](#)

Related Methods

[ASFileSysGetItemProps](#)
[ASFileSysFirstFolderItem](#)
[ASFileSysNextFolderItem](#)
[AVConversion](#) methods

Values

kAVConversionFailed	The conversion failed.
kAVConversionSuccess	The conversion succeeded.
kAVConversionSuccessAsync	The conversion will continue asynchronously.
kAVConversionCancelled	The conversion was cancelled.

AVConversionToPDFHandler

AVConversionToPDFHandlerRec

```
typedef struct _t_AVConversionToPDFHandler {
    AVFileFilterRec convFilter;
    ASSize_t size;
    char uniqueID[256];
    ASBool canDoSync;
    ASInt16 priority;
    AVConversionDefaultSettingsProc defaultSettings;
    AVConversionParamDescProc parameterDescription;
    AVConversionSettingsDialogProc settingsDialog;
    AVConversionConvertToPDFProc convert;
    AVConversionClientData clientData;
    /*new in Acrobat 6.0*.
    AVConversionMimeTypeString *streamMimeTypes;
    AVArraySize numStreamMimeTypes;
} AVConversionToPDFHandlerRec, *AVConversionToPDFHandler;
```

Description

Data structure containing callbacks that implement the “ToPDF” handler’s functionality and data that describes the handler’s conversion capabilities.

Header File

AVCalls.h

Related Callbacks

[AVConversionToPDFEnumProc](#)

Related Methods

[AVAppRegisterToPDFHandler](#)

Members

convFilter	An AVFileFilterRec that describes the types of files that this filter can convert. See description of AVFileFilterRec and AVFileDescRec for more details.
size	Size of the data structure. Must be set to <code>sizeof (AVConversionFromPDFHandlerRec)</code> .

uniqueID	Unique identifier for the conversion handler. Should be of the form com.companyname.productname.type . See PDF_FILEFILTERREC_UNIQUEID and FDF_FILEFILTERREC_UNIQUEID in AVExpT.h .
canDoSync	true if the converter can perform synchronous conversion, false if the converter only does asynchronous conversion. This capability is required for the handler to be accessible from the batch framework.
priority	An integer that indicates where converter should be registered in list of "FromPDF" handlers. A high priority number indicates that this converter is checked earlier than other converters. All Acrobat 5.0 converters have a priority of 0.
defaultSettings	AVConversionDefaultSettingsProc that is called when the handler is registered with the conversion server. Can be NULL .
parameterDescription	AVConversionParamDescProc that is called when a parameter description of this handler is requested. Can be NULL .
settingsDialog	AVConversionSettingsDialogProc that is called when the batch framework or the open dialog requests a settings dialog for this handler. Can be NULL .
convert	AVConversionConvertToPDFProc that is called to perform the conversion operation.
clientData	Provided to all AVConversion callbacks.
streamMimeTypes	A string containing Mime types that can be handled by the convertStream callback.
numStreamMimeTypes	The number of Mime types in streamMimeTypes .



AVCursor

```
typedef struct _t_AVCursor *AVCursor;
```

Description

Data structure representing the cursor. See [Predefined Cursors](#) for a list of already defined cursor shapes.

Header File

AVExpT.h

Related Methods

[AVSysGetCursor](#)

[AVSysGetStandardCursor](#)

[AVSysSetCursor](#)

AVDestInfo

AVDestInfoRec

```
typedef struct _t_AVDestInfo {
    ASSize\_t size;
    const char* namedDest;
    AVTArraySize nameLength;
    PDPageNumber pageNum;
    ASAtom fitType;
    ASFixedRect destRect;
    ASFixed zoom;
} AVDestInfoRec, *AVDestInfo;
```

Description

Data structure representing a destination in a PDF document. An **AVDestInfo** carries all the information that a [PDViewDestination](#) can. Used for ensuring that cross-document links in external windows act as expected, so a client can go to a destination without building it via [PDViewDestCreate](#), which does not work on read-only documents.

Header File

AVExpT.h

Related Methods

[AVPageViewToDestInfo](#)
[AVPageViewUseDestInfo](#)
[AVDestInfoDestroy](#)

Members

size	Size of the data structure. Must be set to sizeof (AVDestInfo) .
namedDest	The named destination associated with this destination. If this is non- NULL , the other attributes are ignored. This destination may contain multi-byte characters.
nameLength	Length of namedDest , in bytes.
pageNum	The page number of the destination view.
fitType	The fit type of the destination view. One of View Destination Fit Types .
destRect	A rectangle enclosing the destination view.
zoom	The zoom factor of the destination view. Use zero to inherit the zoom.



AVDevCoord

```
typedef AVSDKDependentInteger AVDevCoord;
```

Description

An x or y coordinate in the page view's device space.

Header File

AVExpT.h

Related Methods and Structures

numerous

AVDevRect

```
typedef AVRect AVDevRect;
```

Description

Data structure representing a rectangle (a quadrilateral having only horizontal and vertical sides) in a page view's device space.

Header File

```
AVExpT.h
```

Related Methods

```
AVPageViewDragOutNewRect  
AVPageViewDragOutNewRectSnapped  
AVPageViewDragRect  
AVPageViewDragRectSnapped  
AVPageViewDragRectSnappedEx  
AVPageViewDrawAnnotSequence  
AVPageViewDrawRect  
AVPageViewDrawRectOutline  
AVPageViewDrawRectOutlineWithHandles  
AVPageViewGetAnnotRect  
AVPageViewGetAperture  
AVPageViewInvalidateRect  
AVPageViewInvertRect  
AVPageViewInvertRectOutline  
AVPageViewScrollToRect  
AVPageViewSnapRect  
AVRectHandleHitTest
```

Related Callbacks

```
AVAnnotHandlerDrawExProc  
AVAnnotHandlerGetAnnotViewBBoxProc
```

Related Structures

```
AVScreenRect  
AVWindowRect  
AVDragRectParams
```



AVDevSize

```
typedef AVSDKDependentInteger AVDevSize;
```

Description

A size in the page view's device space.

Header File

`AVExpT.h`

Related Methods

[AVPageViewDrawRectOutline](#)
[AVPageViewScrollToRect](#)

Related Structures

[AVLine](#)

AVDocOpenParams

AVDocOpenParamsRec

```
typedef struct _t_AVDocOpenParams {
    ASSize\_t size;
    ASBool useFrame;
    AVScreenRect frame;
    ASBool useVisible;
    ASBool visible;
    /* Available only in or after Acrobat 3.0 */
    ASBool useServerType;
    const char* serverType;
    void* serverCreationData;
    ASBool useSourceDoc;
    AVDoc sourceDoc;
    ASBool useReadOnly;
    ASBool readOnly;
    ASBool useViewType;
    const char* viewType;
    ASBool useViewDef;
    AVDocViewDef viewDef;
    /* New in Acrobat 5.0 */
    ASBool usePermReqProc;
    AVDocPermReqProc permReqProc;
    /* New in Acrobat 6.0 */
    ASCab openActions;
    ASBool suppressDialogs;
} AVDocOpenParamsRec, *AVDocOpenParams;
```

Description

Parameters used when opening a file.

In UNIX, it is not possible to set the frame of the **NULL** document (that is, the window to show when no document is open) using this data structure.

Header File

AVExpT.h

Related Methods

[AVAppOpenHelpFileWithParams](#)
[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromASFileWithParams](#)
[AVDocOpenFromPDDocWithParams](#)

Members

size	Size of the data structure. Must be set to sizeof (AVDocOpenParamsRec) .
useFrame	If true , frame is used to specify the size and location of the window into which the document is opened. If false , frame is ignored and the default frame is used instead. See also visible .
frame	An AVRect specifying the size and location of the window into which the document is opened. In Mac OS, the coordinates are global screen coordinates. In Windows, the coordinates are MDI client coordinates. See also visible .
useVisible	If true , visible is used to determine whether the window is visible after the document is opened. If false , visible is ignored. See also visible .
visible	The window's visibility. If visible is false and useVisible is true , frame is ignored—regardless of the setting of useFrame . In Mac OS, if true , the document is opened into a visible window. If false , the document is opened into a hidden window. In Windows, if true , the document is opened in a visible window. If false , the document is opened in a minimized window.
useServerType	Whether the serverType and <i>serverCreationData</i> fields are used.
serverType	Name of AVDoc server for this AVDoc : "EXTERNAL" — the AVDoc server for an external window
serverCreationData	Platform-dependent server data to associate with the AVDoc server. For a <i>serverType</i> of "EXTERNAL", must be of type ExternalDocServerCreationData .
useSourceDoc	Whether the sourceDoc field is used.
sourceDoc	AVDoc whose window will be taken over by new document. sourceDoc will be closed at the same time.
useReadOnly	Whether the readOnly field is used.
readOnly	If true , open the document in read-only mode.
useViewType	Indicates whether the viewType field is used.

viewType	Type of view to open on document. Permissible values are: "AVPageView" — Displays only the AVPageView , that is, the window that displays the PDF file. Does not display scrollbars, the toolbar, and bookmark or thumbnails pane. Annotations, such as links, are active. "AVDocView" — Displays the AVPageView , scrollbars, and bookmark or thumbnails pane. Annotations, such as links, are active. "AVExternalView" — Displays the AVPageView , scrollbars, toolbar, and bookmark or thumbnails pane. Annotations, such as links, are active. "AVEmbeddedView" — Embeds the PDF file in an external document, an HTML file, for example. Show the first page of the PDF file; no scrollbars, the toolbar, and bookmark or thumbnails pane are visible. Annotations, such as links, are neither displayed nor active.
useViewDef	Whether the viewDef field is used.
viewDef	Initial view with which to open the document. Must be an AVDocViewDef .
usePermReqProc	Whether the permReqProc field be used.
permReqProc	Return PDPermReqDenied to deny a permission, or PDPermReqGranted to grant one.
openActions	Expanded and more flexible version of the viewDef field. If both are specified, the older AVDocViewDef takes precedence.
suppressDialogs	When true , suppress any non-alert dialogs that may be triggered by opening the document.

AVDocPrintOverrideData

```
typedef struct _t_AVDocPrintOverrideData {
    ASSize_t size;
    AVColorForcing colorOverride;
    AVUseValue usePrinterCRD;
    AVUseValue useT1Conversion;
} AVDocPrintOverrideDataRec, *AVDocPrintOverrideData;
```

Description

Structure specifying override parameters used by [AVDocPrintParams](#).

Header File

AVExpT.h

Related Methods

[AVDocPrintPagesWithParams](#)

Members

size	Size of the data structure. Must be set to <code>sizeof (AVDocPrintOverrideDataRec)</code> .
colorOverride	Whether to use color forcing. Default is AVColorForceNone . Can be used to send 1 bit or 8 bit data to printer instead of 24 bit. Ignored on Mac OS.
usePrinterCRD	Whether to allow the viewer to decide if it should use the printer's PostScript Color Rendering Dictionary. Default is kAVuseAuto , which is normally true . but is false where the printer is listed as broken, as in these cases: - When printer-based color management is used, printers whose CRD is incorrect produce light gray or peach for white expressed in an ICC color space. - As of 1/1/02 the CRDs for Tektronix Phaser 740 and Xerox Docuprint N2125 are known to be incorrect, and this may be so for others, especially in these printer families. It is possible to change the printer defaults, to list additional printers, for example, or to remove a printer from the broken list if the driver if fixed. Contact Adobe technical support for details.

useT1Conversion

Whether to allow the viewer to decide if it should use T1 color conversion (that is, convert Type1 fonts to more efficient font representations) for better performance . Ignored on Mac OS.

Default is **kAVUseAuto**, which is normally **true**, except in cases where the printer cannot handle optimized fonts and needs full, slow, Type1 fonts. As of 1/1/02 only the Windows 16 bit HP LJ 4siMX driver is known to be in this category.

It is possible to change the printer defaults, to list additional printers, for example, or to remove a printer from the broken list if the driver is fixed. Contact Adobe technical support for details.

AVDocPrintParams

AVDocPrintParamsRec

```
typedef struct _t_AVDocPrintParams *AVDocPrintParams;
typedef struct _t_AVDocPrintParams {
    ASSize_t size;
    ASBool interactive;
    ASBool cancelDialog;
    ASInt32 firstPage;
    ASInt32 lastPage;
    ASInt32 psLevel;
    ASBool binaryOK;
    ASBool shrinkToFit;
    ASAtom fileSysName;
    ASPathName filePathName;
    ASPlatformPrinterSpec printerSpec;
    ASBool embedded;
    AVRect embeddedRect; /* cosmetic type change in 6.0 */
    ASBool emitToPrinter;
    ASBool emitToFile;
    ASBool doColorSeparations;
    ASEnum8 emitFileOption;
    ASEnum8 emitFontOption;
    ASUns32 emitFlags;
    /* New in Acrobat 4.0 */
    PDPageRange* ranges;
    AVTSmallArraySize numRanges; /* cosmetic type change in 6.0 */
    ASBool TTasT42;
    ASBool printAsImage;
    ASBool printerHasFarEastFonts;
    ASBool reverse;
    ASInt32 pageSpec; /* Updated in 5.0 */
    /* New in Acrobat 5.0 */
    ASInt32 transparencyLevel;
    char destProfile[256];
    /* New in Acrobat 6.0 */
    AVPageSize pageSize;
    AVDocPrintTileData tileData;
    AVDocPrintRasterizeData rasterData;
    AVDocPrintOverrideData overrideData;
    ASFixedRect *selectRect;
    PDOContext ocContext;
    ASUTF16Val userNote[kPrintUserNoteLen];
    AVResourcePolicy resPolicy;
    ASUns32 marksFlags;
} AVDocPrintParamsRec;
```


Description

Structure that specifies how to print a document.

Header File

AVExpT.h

Related Types

[Emit Font Options](#)
[AV Print Mark Flags](#)
[Emit Flags](#)

Related Methods

[AVDocPrintPagesWithParams](#)

Members

size	Size of the data structure. Must be set to <code>sizeof (AVDocPrintParamsRec)</code> .
One and only one of the following booleans must be set:	● interactive — Displays a Print dialog box and print status window while printing. ● cancelDialog — Displays a Cancel dialog box while printing. ● embedded — Renders one page scaled to the size specified by embeddedRect. ● emitToPrinter — Non-interactive output to a printer without a Print dialog box or status window. ● emitToFile — Non-interactive output to a file. Used to emit color separations or EPS. This flag <i>cannot</i> be used with the Adobe Reader.

Fields used only if emitToPrinter or emitToFile is true :	● firstPage — First page to print. The first page of the document is 0. A value of -1 prints from the first page. ● lastPage — Last page to print. A value of -1 prints to the last page. ● psLevel — If printing to PostScript, 1 means emit as level 1, 2 means level 2, and 3 means level 3. ● binaryOK — true if a binary channel to the printer is supported, false otherwise. ● shrinkToFit — true if the page is scaled to fit the printer page size, false otherwise. Ignored if pageSize is specified. ● pageSpec — (Updated for Acrobat 5.0) Indicates odd, even, or all pages to be printed within the range—only meaningful when firstPage and lastPage parameters are used. See Page Specification. ● numRanges — Number of page ranges to print. If not 0, then ranges supercedes firstpage/lastpage/pagespec. ● ranges — An array of pages ranges to print if numRanges is not 0.
FileSysName	Used only if emitToPrinter or emitToFile is true. The file system name or ASAtomNull; see filePathName. If emitToPrinter is true and filePathName is specified, fileSysName must be the name of the default file system. You can get the file system's name from the ASFileSysGetFileSysNameProc callback in the ASFileSysRec of the file system you are using.
FilePathName	An ASPathName of the default ASFileSys. Used only if emitToPrinter or emitToFile is true. If emitToPrinter is true and filePathName is non-NULL, the system printer driver is used to emit the output stream to the file. Implemented for Windows only.
printerSpec	Optionally used if interactive, embedded, or emitToPrinter is true. If NULL, a default system printer is used. If non-NULL, printerSpec is a platform-specific value. Must be an ASPlatformPrinterSpec.
embedded	true if an embedded view of a page to print, false otherwise. ● firstPage and lastPage must be the same. ● embeddedRect specifies the location on the page where the view of the page is to appear. ● The printer must be specified as an HDC or CGrafPtr.

embeddedRect	Location on the page where the view of the page is to appear, specified in device coordinates for the current printer.
emitToPrinter	If true, use the system printer driver for output. If filePathName is specified, uses the driver to create the file. Raises genErrBadParm if an invalid parameter is provided (for example, printing to PDFWriter, Distiller, or to a printer that has been un-installed).
The following parameters are for emission of PostScript Level 1 Color Separations and EPS—or standard PostScript. Creates and writes to filePathName (may not be NULL). Does not use the system printer driver. Only has partial font emitting capabilities on some platforms: • Macintosh: embedded and system Type 1 fonts only; no TrueType or substitution fonts • UNIX: all fonts • Windows: embedded and system Type 1 fonts only; no TrueType or substitution fonts	
emitToFile	If true , emit non-interactive output to a file. Used to emit color separations or EPS. This flag cannot be used with the Adobe Reader.
doColorSeparations	Perform level 1 color separations. See <i>Color Separation Conventions for PostScript Language Programs</i> , Technical Note #5044.
emitFileOption	File output options: PostScript or EPS, with or without a preview. Must be one of the following values: kAVEmitFileEPS — PostScript file kAVEmitFileEPSNoPrev — EPS file with no preview kAVEmitFileEPSMacStdPrev — EPS file with standard preview kAVEmitFileEPSMacExtPrev — EPS file with extended preview
emitFontOption	Font output options. Must be one of Emit Font Options .
emitFlags	Additional emit options. Must be one of Emit Flags .
The following parameters control TrueType-to-Type 1 conversion for PostScript printing.	
TTasT42	If true , send TrueType fonts as TrueType fonts (level 3 and most level 2 PostScript printers. If false , convert TrueType to Type 1. This is typically <i>only</i> desirable for Level 1 PostScript where no TrueType handling is present.
printAsImage	If true , print pages as an image.
printerHasFarEastFonts	If true , do not down load Far East fonts to printer.

reverse	Print from lastPage to firstPage , or, if numRanges is not 0, print the specified ranges in reverse order, and also reverse the page order within each range.
transparencyLevel <i>(New in Acrobat 5.0)</i> <i>(Obsolete in Acrobat 6.0—use transparency in AVDocPrintRasterizeData instead.)</i>	Transparency level reflects the pull-down on the Advanced print dialog for controlling how much rasterization should be performed when flattening transparent objects during printing. 1 = The entire page will be rasterized. Use this setting for printing or exporting complex pages with many transparent objects. Ideal for fast output at low resolution; higher resolution will yield higher quality but increase processing time. Size of saved files or print spool files may be large. 2 = Maintains simpler vector objects, but rasterizes more complex areas involving transparency. Ideal for artwork with only a few transparent objects. Some printers may yield rough transitions between bordering vector and raster objects and make hairlines appear thicker. Appropriate for low-memory systems. 3 = Maintains most objects as vector data, but rasterizes very complex transparent regions. Generally the best setting for printing and exporting most pages. With some printers, improves transition issues between bordering vector and raster objects. 4 = Maintains most of the page content as vectors, rasterizing only extremely complex areas. Produces high quality output that is generally resolution-independent. Higher occurrences of transparent regions will increase processing time. With some printers improves transition issues between bordering vector and raster objects. 5 = The entire page is printed or exported as vector data, to the greatest extent possible. This produces the highest quality resolution-independent output. Processing of complex pages may be very time and memory intensive.
destProfile	Represents the name of the destination profile to use when doing host-based color management.
pageSize	The page size. If specified, shrinkToFit value is ignored.
tileData	The tile data. If NULL , uses default values.
rasterData	The raster data. If NULL , uses default values.
overrideData	The override data. If NULL , uses default values.

selectRect	Area of page(s) to print, in points. If NULL , print whole page.
ocContext	The optional-content context to use. If NULL , use current on-screen optional-content state, but apply print-specific AutoState changes from the default optional-content configuration's AS entry.
userNote [kPrintUserNoteLen]	A short note to user that appears in print dialog.
resPolicy	The font and resource policy.
marksFlags	An OR of constant bit flag values defined for AV Print Mark Flags , that controls what marks are emitted. NOTE: In Adobe Acrobat 6.0 Standard, these flags are ignored.

AVDocPrintRasterizeData

```
typedef struct _t_AVDocPrintRasterizeData {  
    ASSize_t size;  
    AVRasterizeFlags flags;  
    ASInt32 transparency;  
    ASUns32 bitmapResolution;  
    ASUns32 gradientResolution;  
} AVDocPrintRasterizeDataRec, *AVDocPrintRasterizeData;
```

Description

Structure specifying rasterization parameters used by [AVDocPrintParams](#). Constant DPI values are defined in `AVExpT.h`.

Header File

`AVExpT.h`

Related Methods

[AVDocPrintPagesWithParams](#)

Members

size	Size of the data structure. Must be set to sizeof (AVDocPrintRasterizeData) .
flags	An OR of bit flags that control the rasterization method to use for printing.
transparency	Degree of transparency, in the range 1 to 100, where 1 means all raster, and 100 means preserve as much transparency data as possible.
bitmapResolution	DPI for bitmaps. Default is 300. This value is also used when printAsImage is true .
gradientResolution	DPI for gradients interior to the object (not edges). Can generally be lower than the bitmapResolution . Default is 150.

AVDocPrintSepsParams

```
typedef struct _t_AVDocPrintSepsParams
{
    ASSize_t size;
    AVDoc avDoc;
    ASInt32 firstPage;
    ASInt32 lastPage;
    ASAtom fileSysName;
    ASPathName filePathName;
    ASEnum8 emitFileOption;
    ASUns32 emitFlags;
    AVPageSize pageSize;
    AVDocPrintRasterizeData rasterData;
    AVDocPrintOverrideData overrideData;
    PDHostSepsSpec sepsSpec;
}
AVDocPrintSepsParamsRec;
```

Description

Structure specifying color separation printing parameters used by [AVDocPrintSeparations](#).

Header File

AVExpT.h

Related Methods

[AVDocPrintSeparations](#)

Members

size	Size of the data structure. Must be set to sizeof (AVDocPrintSepsParamsRec) .
avDoc	The document for which to print color separations.
firstPage	First page to be printed. First page number is 0.
lastPage	Last page to be printed.

fileSysName, filePathName	The method creates an output file for each plate for each page, using the fileSysName for the file system, and the filePathName as the root file name for each file. It appends the page number, then the colorant name to each output file name. If filePathName is not a valid pathname to a file Acrobat can open and write to, the method raises an exception using the value returned by ASFileSysOpenFile.
emitFileOption	Not used.
emitFlags	Additional emit options. Must be one of the Emit Flags .
pageSize	Not used.
rasterData	NULL uses default values.
overrideData	NULL uses default values.
sepsSpec	The separations specification parameter structure.

AVDocPrintTileData

```
typedef struct _t_AVDocPrintTileData {
    ASSize_t size;
    ASUns32 overlap;
    ASFixed scale;
    ASBool label;
    AVTileMark markType;
} AVDocPrintTileDataRec, *AVDocPrintTileData;
```

Description

Structure specifying print tiling parameters used by [AVDocPrintParams](#).

Header File

AVExpT.h

Related Methods

[AVDocPrintPagesWithParams](#)

Members

size	Size of the data structure. Must be set to sizeof (AVDocPrintTileData) .
overlap	Number of points by which tiled pages overlap.
scale	A scaling value. 1.0 indicates normal scaling.
label	When true , each tiled sheet has a label that contains the file name, the page number (page x of y, where y is the total number of pages), and the date and time.
markType	The tile marking type.

AVDocSaveParams

AVDocSaveParamsRec

```
typedef struct _t_AVDocSaveParams {  
    ASSize\_t size;  
    ASBool useSaveDialog;  
    /* New in Acrobat 5.0 */  
    ASBool dontAllowConversions;  
} AVDocSaveParamsRec, *AVDocSaveParams;
```

Description

Structure used by [AVDocDoSaveAsWithParams](#) containing parameters that a client wishing to save a file might want to specify. It is passed in by address to [AVDocDoSaveAsWithParams](#) with a size field so that current clients replacing [AVDocDoSaveAsWithParams](#) won't break in the future if new open specifications are provided.

Header File

AVExpT.h

Related Methods

[AVDocDoSaveAsWithParams](#)

Members

size	Size of the data structure. Must be set to <code>sizeof (AVDocSaveParamsRec)</code> .
useSaveDialog	Use the standard file Save dialog box.
dontAllowConversions (New in Acrobat 5.0)	Don't use convert from PDF handlers.

AVDocSelectionServer

AVDocSelectionServerRec

```
typedef struct _t_AVDocSelectionServer {
    ASSize_t size;
    AVDocSelectionGetTypeProc GetType;
    AVDocSelectionGettingSelectionProc GettingSelection;
    AVDocSelectionAddedToSelectionProc AddedToSelection;
    AVDocSelectionLosingSelectionProc LosingSelection;
    AVDocSelectionRemovedFromSelectionProc RemovedFromSelection;
    AVDocSelectionCanSelectAllProc CanSelectAll;
    AVDocSelectionSelectAllProc SelectAll;
    AVDocSelectionCanPropertiesProc CanProperties;
    AVDocSelectionPropertiesProc Properties;
    AVDocSelectionCanDeleteProc CanDelete;
    AVDocSelectionDeleteProc Delete;
    AVDocSelectionCanCopyProc CanCopy;
    AVDocSelectionCopyProc Copy;
    AVDocSelectionEnumSelectionProc EnumSelection;
    AVDocSelectionShowSelectionProc ShowSelection;
    AVDocSelectionCanCutProc CanCut;
    AVDocSelectionCutProc Cut;
    AVDocSelectionCanPasteProc CanPaste;
    AVDocSelectionPasteProc Paste;
    AVDocSelectionKeyDownProc KeyDown;
    /* Renamed in Acrobat 4.0 */
    AVDocSelectionHighlightSelectionProc HighlightSelection;
    /* New in Acrobat 4.0 */
    AVDocSelectionGetSelectionTypeProc GetSelectionType;
    AVDocSelectionEnumPageRangesProc EnumPageRanges;
    /* New in Acrobat 5.0 */
    AVDocSelectionGetAVRectProc GetAVRect;
    AVDocSelectionShowMenuProc ShowMenu;
    /* New in Acrobat 6.0 */
    AVDocSelectionHighlightSelectionExProc HighlightSelectionEx;
    AVDocSelectionAcquireQuadsProc AcquireQuads;
} AVDocSelectionServerRec, *AVDocSelectionServer;
```

Description

Data structure containing callbacks that implement a selection server. The callbacks implement the selection server functions. For example, add an item to the selection, remove an item from the selection, copy the current selection to the clipboard.

Header File

AVExpT.h



Related Methods

[AVDocRegisterSelectionServer](#)
[AVDocGetSelectionServerByType](#)

Members

size	Size of the data structure. Must be set to sizeof (AVDocSelectionServerRec) .

AVDocServerType

```
typedef enum _t_AVDocServerType {
    AVDocServerUnknown,
    AVDocServerDefault,
    AVDocServerExternal,
    AVDocServerHelp
} AVDocServerType;
```

Description

Constant values indicating a type of document server being used for a document.

Header File

AVExpT.h

Related Methods

[AVDocGetServerType](#)

Members

AVDocServerUnknown	Unknown server type.
AVDocServerDefault	The default document server, used for most documents.
AVDocServerInternal	The same as the default document server.
AVDocServerExternal	A server used for external documents shown in a Web browser.
AVDocServerHelp	A server used for documents displayed in the help window.

AVDocViewDef

AVDocViewDefRec

```
typedef struct _t_AVDocViewDef {
    ASSize\_t size;
    ASBool bringToFront;
    ASBool usePageViewInfo; /* page view info */
    PDLayoutMode pageViewLayoutMode;
    PDPageNumber pageViewPageNum;
    AVZoomType pageViewZoomType;
    ASFixed pageViewZoom;
    AVDevCoord pageViewX;
    AVDevCoord pageViewY;
    ASBool pageViewStartThread;
    AVPageIndex pageViewThreadIndex;
    PDBead pageViewBead;
    ASBool useOverViewInfo; /* overview info */
    PDPageMode overViewMode;
    AVPixelOffset overViewPos;
    ASInt32 overViewX;
    ASInt32 overViewY;
    ASBool useWindowInfo; /* window info */
    AVRect windowFrame;
    ASBool unused1;
    const char* unused2;
} AVDocViewDefRec, *AVDocViewDef;
```

Description

Structure that defines a view of a document, including page, zoom, and so on.

NOTE: Numeric types have changed in Acrobat 6.0, and the [AVDocGetViewDef](#) and [AVDocSetViewDef](#) methods have been superseded by [AVDocGetViewDefEx](#) and [AVDocSetViewDefEx](#).

Header File

AVExpT.h

Related Methods

[AVDocGetViewDef](#)

[AVDocSetViewDef](#)

Members

size	Size of the data structure. Must be set to <code>sizeof (AVDocViewDef)</code> .

bringToFront	If true , bring window to front; if false , do not bring window to front.
usePageViewInfo	If true , use the next 6 page view fields. If false , it does not use them.
pageViewLayoutMode	Page layout mode; must be one of PDLayouMode .
pageViewPageNum	Page number.
pageViewZoomType	Zoom type; must be one of AVZoomType .
pageViewZoom	Zoom factor; used if pageViewZoomType is AVZoomNoVary . Use zero to inherit zoom.
pageViewX	The x-coordinate to scroll to.
pageViewY	The y-coordinate to scroll to.
pageViewStartThread	If true , use the next two article thread fields. If false , it does not use them.
pageViewThreadIndex	Current thread index.
pageViewBead	Current PDBead .
useOverViewInfo	If true , use the next four view fields. If false , it does not use them.
overViewMode	The PDPageMode to use.
overViewPos	Position of splitter.
overViewX	The x-coordinate to scroll to in bookmark or thumbnail pane.
overViewY	The y-coordinate to scroll to in bookmark or thumbnail pane.
useWindowInfo	If true , use the <i>windowFrame</i> field. If false , it does not use them.
windowFrame	New window frame in which to display the document.
unused1, unused2	Unused.

AVDragRectParams

```
typedef struct _t_AVDragRectParams {
    ASSize_t size;
    AVPageView pageView;
    ASInt32 xStart;
    ASInt32 yStart;
    ASFixedRect *startRect;
    ASFixedRect *resultRect;
    AVDragType dragType;
    AVDevRect *extrema;
    AVCursor *cursorArray;
    ASInt32 nCursors;
    AVPageViewDrawProc drawProc;
} AVDragRectParamsRec, *AVDragRectParams;
```

Description

The parameters for [AVPageViewDragRectSnappedEx](#), which supercedes [AVPageViewDragRectSnapped](#) in Acrobat 6.0. In addition to the parameters allowed by the earlier method, the new version allows you to specify your own drawing procedure.

Header File

AVExpT.h

Related Method

[AVPageViewDragRectSnappedEx](#)

Members

size	The size of the array. Must be set to sizeof (AVDragRectParamsRec) .
pageView	The page view where the drag occurs.
xStart	Starting x-coordinate in device space.
yStart	Starting y-coordinate in device space.
startRect	Initial rectangle in user space.
resultRect	Resulting rectangle in user space
dragType	Desired drag type, typically the result of AVRectHandleHitTest .
extrema	The device-space drag bounds.
cursorArray	The drag cursors, or NULL for default cursors.
nCursors	Number of cursors in cursorArray .



drawProc	The client-defined drawing procedure, or NULL for the default Acrobat drawing procedure.

AVDragType

```
typedef ASEnum8 AVDragType;
```

Description

Constants that specify the commands for moving and changing the size of a rectangle.

Header File

AVExpT.h

Related Methods

[AVRectHandleHitTest](#)

Values

<code>kAVDragRect</code>	Move the whole rectangle.
<code>kAVDragTopLeft</code>	Top left corner.
<code>kAVDragTopRight</code>	Top right corner.
<code>kAVDragBottomRight</code>	Bottom right corner.
<code>kAVDragBottomLeft</code>	Bottom left corner.
<code>kAVDragTopMiddle</code>	Top middle.
<code>kAVDragRightMiddle</code>	Right middle.
<code>kAVDragBottomMiddle</code>	Bottom middle.
<code>kAVDragLeftMiddle</code>	Left middle.
<code>kAVDragSnapToTopLeft</code>	Snap to top left.
<code>kAVDragSnapToTop</code>	Snap to top.
<code>kAVDragSnapToTopRight</code>	Snap to top right.
<code>kAVDragSnapToRight</code>	Snap to right.
<code>kAVDragSnapToBottomRight</code>	Snap to bottom right.
<code>kAVDragSnapToBottom</code>	Snap to bottom.
<code>kAVDragSnapToBottomLeft</code>	Snap to bottom left.
<code>kAVDragSnapToLeft</code>	Snap to left.

AVExtensionInfo

AVExtensionInfoRec

```
typedef struct _AVExtensionInfoRec {
    ASAtom  asaName;
    ASBool  bLoaded;
    ASBool  bCertified;
    ASUns16 nMajorVersion, nMinorVersion;
    char *cDate;
    ASPathName aspFile;
    char *cDescription;
    char *cLegal;
    char *cDependencies;
} AVExtensionInfoRec, *AVExtensionInfo;
```

Description

Data structure containing information about a client loaded by the viewer.

NOTE: For third-party (non-Adobe) clients, only **asaName**, **bLoaded**, and **bCertified** are valid.

Header File

AVExpT.h

Related Methods

[AVExtensionAcquireInfo](#)

[AVExtensionReleaseInfo](#)

Members

asaName	The registered name of the client.
bLoaded	Always true indicating that the client was loaded.
bCertified	true if the client is certified, false otherwise.
nMajorVersion, nMinorVersion	The major and minor versions of the client.
cDate	The creation timestamp on the client.
aspFile	The path to the client.
cDescription	A description of the client. May be NULL .
cLegal	Legal text associated with the client. May be NULL .



cDependencies	The dependencies of the client.

AVFileDescRec

```
typedef struct _t_AVFileDescRec {
    char extension[32];
    ASUns32 macFileType;
    ASUns32 macFileCreator;
} AVFileDescRec;
```

Description

Structure to handle file types and/or extensions in open and save dialogs.

Header File

AVExpT.h

Related Methods

- [AVAppChooseFolderDialog](#)
- [AVAppOpenDialog](#)
- [AVAppSaveDialog](#)

Members

extension	Up to 32-character string for file extension. Use \0 on Windows for do not care (ignored on Windows only if \0 is used).
macFileType	File type; used on Macintosh only. Use 0 for do not care.
macFileCreator	File creator; used on Macintosh only. Use 0 for do not care.

AVFileFilterRec

```
typedef struct _t_AVFileFilterRec {
    ASText filterDescription;
    AVFileDescRec* fileDescs;
    AVArraySize numFileDescs;
} AVFileFilterRec;
```

Description

Structure to hold a series of file type descriptors that form a file filter for an open or save dialog.

Header File

`AVExpT.h`

Related Callbacks

Various

Related Methods

- [AVAppChooseFolderDialog](#)
- [AVAppOpenDialog](#)
- [AVAppSaveDialog](#)

Members

filterDescription	Localized string describing this filter. It is the name that appears in the open or save dialog.
fileDescs	An array of AVFileDescRec . A single AVFileFilterRec can have as many AVFileDescRecs as needed. On Windows, the filename is concatenated with the extension string of the relevant AVFileDescRec in the Open and Save dialogs. On the Macintosh, the fileDescription string is used in the File Open and Save dialogs, and the AVFileDescRecs are used to filter which files are displayed when that AVFileFilterRec is selected.
numFileDescs	The number of AVFileDescRecs in fileDescs .



AVFilterIndex

```
typedef ASInt16 AVFilterIndex;
```

Description

A filter index value for AV methods.

Header File

AVExpT.h

Related Methods

[AVAppOpenDialog](#)

[AVAppSaveDialog](#)



AVFlagBits16

```
typedef ASInt16 AVFlagBits16;
```

Description

A flag-bits value for use in callback procedures.

Header File

`AVExpT.h`

Related Callbacks

[AVAnnotHandlerDoClickProc](#)
[AVAnnotHandlerDoKeyDownExProc](#)
[AVAnnotHandlerDoKeyDownProc](#)
[AVDocSelectionKeyDownProc](#)
[AVPageViewClickProc](#)
[AVPageViewKeyDownProc](#)

Related Structures

[DoClickProcType](#)
[DoKeyDownProcType](#)



AVFlagBits32

```
typedef ASUns32 AVFlagBits32;  
#define AV_ANNOT_POPUPISREADONLY 0x0001  
#define AV_ANNOT_SHOW_OFFSCREEN_INDICATOR 0x0002  
#define AV_ANNOT_SUPPORTS_REPLIES 0x0004
```

Description

A flag-bits value for use in callback procedures.

Header File

AVExpT.h

Related Callbacks

[AVAnnotHandlerGetFlagsProc](#)
[AVIconHandlerGetFlagsProc](#)

Related Methods

[AVPageViewAppearanceGetAVMatrix](#)
[AVSysGetModifiers](#)
[AVWindowNew](#)
[AVWindowNewFromPlatformThing](#)

AVFullScreenMonitor

```
typedef ASEnum8 AVFullScreenMonitor;
```

Description

Constants that describe the preferred monitor to use when going full-screen on a multi-monitor system.

Header File

```
AVExpT.h
```

Members

kAVFullScreenLargestIntersection	Use the monitor with the largest intersection.
kAVFullScreenMostColors	Use the monitor with the most colors.
kAVFullScreenWidest	Use the monitor with the widest screen.
kAVFullScreenTallest	Use the monitor with the tallest screen.
kAVFullScreenLargest	Use the monitor with the largest screen.
kAVFullScreenMain	Use the monitor with the main screen.



AVIcon

```
#define AVIcon void*
```

Description

An icon on a menu item or toolbar button.

Header File

AVCalls.h

Related Methods

[AVMenuItemNew](#)
[AVSysGetIconFromFilename](#)
[AVSysGetIconFromMimeType](#)
[AVSysGetIconFromTypeAndCreator](#)
[AVToolButtonGetIcon](#)
[AVToolButtonNew](#)
[AVToolButtonSetIcon](#)

AVIconBundle

AVIconBundleRec

```
#if WIN_PLATFORM
typedef HICON AVIconBundleIconRef;
#elif MAC_PLATFORM
typedef IconSuiteRef AVIconBundleIconRef;
#else
typedef void* AVIconBundleIconRef;
#endif /* WIN_PLATFORM */
typedef struct _t_AVIconBundleRec {
    ASUns32 tag1;
    ASUns32 tag2;
    ASInt32 version;
    AVIconBundleIconRef grayIcon;
    AVIconBundleIconRef colorIcon;
} AVIconBundleRec, *AVIconBundle;
```

Description

An icon bundle allows you to gather up multiple icons and present them to Acrobat as a single **AVIcon**. For example, when creating a toolbar button you can pass in an icon bundle specifying both gray and color icons; the gray icon will be used to draw the button in its normal state, the color icon will be used to draw the button when the pointer is over it.

The format for icon bundles is platform-specific (primarily since the format for **AVIcons** is platform-specific). On Windows the icons can be specified using HICONs, not HBITMAPs. On the Mac, they are **IconSuiteRef**, never SICN resources. Both platforms support the PNG format.

The tags at the front are there so the implementation can determine beyond a shadow of a doubt that the information passed in is an icon bundle and not an Acrobat 4-compatible **AVIcon**.

Header File

AVExpT.h

Members

tag1	Set to AVIC (for example, bundle.tag1 = 'AVIC')
tag2	Set to ONBU (for example, bundle.tag2 = 'ONBU')
version	Set to version of app (for example, 0x00050000 for Acrobat 5.0)
grayIcon	Defined according to the typedef shown above.
colorIcon	Defined according to the typedef shown above.

AVIconBundle6

AVIconBundleRec6

```
struct _t_AVIconBundleRec6 {
    ASUns32 tag1;
    ASUns32 tag2;
    ASInt32 version;
    AVIconHandler handler;
    void *iconData;
};
```

Description

An icon bundle allows you to gather up multiple icons and present them to Acrobat as a single **AVIcon**. For example, when creating a toolbar button you can pass in an icon bundle specifying both gray and color icons; the gray icon will be used to draw the button in its normal state, the color icon will be used to draw the button when the pointer is over it. The format for icon bundles is platform-specific (primarily since the format for **AVIcons** is platform-specific). On Windows the icons are specified using HICONs, not HBITMAPs. On the Mac, they are **IconSuiteRef**, never SICN resources. The tags at the front are there so the implementation can determine beyond a shadow of a doubt that the information passed in is an icon bundle and not an Acrobat 4-compatible **AVIcon**.

NOTE: This form of the icon bundle is new in Acrobat 6.0.

Header File

AVExpT.h

Members

tag1	Set to AVIC (for example, bundle.tag1 = 'AVIC')
tag2	Set to ONBU (for example, bundle.tag2 = 'ONBU')
version	Set to version of app (for example, 0x00060000 for Acrobat 6.0)
handle	The icon handler.
iconData	A pointer to the icon data record structure.



AVIconColorFormat

```
typedef ASEnum16 AVIconColorFormat;
```

Description

A color format for an [AVIconBundle6](#).

Header File

AVExpT.h

Related Methods

[AVAppCreateIconBundle6](#)

[AVIconHandlerOpenStmProc](#)



AVIconData

AVIconDataRec

```
typedef struct _t_AVIconData {  
    ASStm dataStm;  
    AVIconColorFormat eColorFormat;  
} AVIconDataRec, *AVIconData;
```

Description

A data record for an [AVIconBundle6](#).

Header File

AVExpT.h

Related Methods

[AVAppCreateIconBundle6](#)

Members

dataStm	A data stream containing the binary-encoded data.
eColorFormat	The color this icon represents.



AVIconDataFormat

```
enum
{
#ifdef MAC_PLATFORM
    kAVIconMacIconRef16, // 16x16 icon ref
    kAVIconMacIconRef32, // 32x32 icon ref
    kAVIconMacIconRef128, // 128x128 icon ref
#endif
    kAVIconPNG,
    kAVIconLayered
}; typedef ASEnum16 AVIconDataFormat;
```

Description

Constants that specify a data format for an [AVIconBundle6](#).

Header File

AVExpT.h

Related Methods

[AVAppCreateIconBundle6](#)

AVIconHandler

AVIconHandlerRec

```
typedef struct _t_AVIconHandlerRec {
    ASInt32 size;
    void* clientData;
    AVIconHandlerMeasureProc Measure;
    AVIconHandlerOpenStmProc OpenStm;
    AVIconHandlerGetFlagsProc GetFlags;
    AVIconHandlerReleaseProc Release;
} AVIconHandlerRec, *AVIconHandler;
```

Description

An icon handler for an [AVIconBundle6](#).

Header File

AVExpT.h

Related Methods

[AVAppCreateIconBundle6](#)

Members

size	The size of the record.
clientData	A pointer to user-defined data passed to the callback.
Measure	The callback procedure for getting the height and width of icons in the set.
OpenStm	The callback procedure for opening a stream for reading the icon data.
GetFlags	The callback procedure for retrieving flag values.
Release	The callback procedure for releasing the object.



AVIdentity

```
typedef enum {  
    kAVILoginName,  
    kAVIName,  
    kAVICorporation,  
    kAVIEMail,  
    kAVILast  
} AVIdentity;
```

Description

Constants that identify the properties of a user's identity.

Header File

AVExpT.h

Related Methods

[AVIdentityGetText](#)
[AVIdentitySetText](#)

AVInfoPanelUpdateType

```
typedef ASEnum8 AVInfoPanelUpdateType;
```

Description

Constants for use with the [AVPageViewUpdateInfoPanel](#).

Header File

```
AVExpT.h
```

Related Methods

```
AVPageViewUpdateInfoPanel
```

Members

kAVInfoPanelLock	Client wishes to assume control over the output of the info panel.
kAVInfoPanelUnlock	Client is transferring control back to Acrobat to update the info panel.
kAVInfoPanelRect	Client is passing the values to Acrobat that should be displayed in the info panel.



AVInkValue

```
typedef struct AVInkValue
{
    ASAtom inkName;
    float value;
} AVInkValue;
```

Description

An ink value for use in color separation methods.

Header File

AVExpT.h

Related Methods

[AVPageViewGetPixelInformationAtPoint](#)



AVKeyCode

```
typedef ASUns16 AVKeyCode;
```

Description

A key code value for use in key-down callback procedures.

Header File

`AVExpT.h`

Related Callbacks

[AVAnnotHandlerDoKeyDownExProc](#)
[AVAnnotHandlerDoKeyDownProc](#)
[AVDocSelectionKeyDownProc](#)
[AVPageViewKeyDownProc](#)

Related Structures

[DoKeyDownProcType](#)



AVCoord

```
typedef ASInt32 AVCoord;
```

Description

A coordinate for a point in device space that supports 32-bit calls prior to Acrobat 6.0.

Unless you must support prior versions and are using `ACRO_SDK_LEVEL=1`, use [AVDevCoord](#) instead.

Header File

`ASExpT.h`

Related Methods

[AVPageViewDeviceToInfo](#)

[AVPageViewInfoToDevice](#)

[AVWindowGetCursorAtPoint](#)

AVLine

```
typedef struct _tag_AVLine {  
    AVDevCoord x1;  
    AVDevCoord y1;  
    AVDevCoord x2;  
    AVDevCoord y2;  
    AVDevSize width;  
    ASFixedP dashArray;  
    AVTSmallArraySize arrayLen;  
} AVLineRec, *AVLine;
```

Description

A structure that defines a line for a page view.

Header File

ASExpT.h

Related Methods

Members

x1	Starting x coordinate.
y1	Starting y coordinate.
x2	Ending x coordinate.
y2	Ending x coordinate.
width	The width of the line.
dashArray	An array of dash lengths, or NULL if the line is solid.
arrayLen	The size of dashArray , if it is not NULL .



AVMenuIndex

```
typedef ASInt32 AVMenuIndex;
```

Description

A menu index value that indicates a user's choice in a pop up menu.

Header File

AVExpT.h

Related Methods

- [AVMenubarAcquireMenuByIndex](#)
- [AVMenubarGetMenuIndex](#)
- [AVMenubarAddMenu](#)
- [AVMenuAcquireMenuItemByIndex](#)
- [AVMenuGetMenuItemIndex](#)
- [AVMenuAddMenuItem](#)
- [AVPageViewDoPopupMenu](#)

AVOpenSaveDialogFlags

```
typedef ASUns32 AVOpenSaveDialogFlags;
enum {
    kAVOpenSaveAllowAllFlag = 1 << 0,
    kAVOpenSaveAllowMultiple = 1 << 1,
    kAVOpenSaveAllowForeignFileSystems = 1 << 2,
    kAVOpenSaveAllowSettingsButton = 1 << 3
};
```

Description

Bit flags specifying open and save dialog settings.

Header File

AVExpT.h

Members

kAVOpenSaveAllowAllFlag	Use "All Files (*.*)" file filter for dialog. Meaningful only for an open dialog.
kAVOpenSaveAllowMultiple	Allow multiple files to be opened through this dialog. Meaningful only for an open dialog.
kAVOpenSaveAllowForeignFileSystems	Allow file systems other than the default to be used to open the file(s).
kAVOpenSaveAllowSettingsButton	Adds a Settings button to the dialog. Meaningful for open and save dialogs.

AVOpenSaveDialogParams

AVOpenSaveDialogParamsRec

```
typedef struct {
    ASSize_t size;
    AOpenSaveDialogFlags flags;
    AVWindow parentWindow;
    AText windowTitle;
    AText actionButtonTitle;
    AText cancelButtonTitle;
    AFileSys initialFileSys;
    APathName initialPathName;
    AText initialFileName;
    AVFileFilterRec** fileFilters;
    AVArraySize numFileFilters;
    AOpenSaveDialogSettingsComputeEnabledProc
        settingsComputeEnabledProc;
    AOpenSaveDialogSettingsExecuteProc settingsExecuteProc;
    void* settingsProcData;
} AOpenSaveDialogParamsRec, *AOpenSaveDialogParams;
```

Description

A structure defining the properties and callbacks related to a file open/save dialog.

Header File

AVExpT.h

Related Methods

- AVAppChooseFolderDialog
- AVAppOpenDialog
- AVAppSaveDialog

Members

size	The size of this structure. Set to sizeof (AVOpenSaveDialogParamsRec).
flags	A bitwise OR of the AOpenSaveDialogFlags.
parentWindow	Parent window of dialog (ignored on the Macintosh platform). May be NULL.
windowTitle	Title of dialog that is used for the prompt. May be NULL for default title.

actionButtonTitle	Title of action button (Open , Save , or Choose). May be NULL for default title.
cancelButtonTitle	Title of cancel button. May be NULL for default title
initialFileSys	May be NULL if flags does not contain kAVOpenSaveAllowForeignFileSystems .
initialPathName	Used to specify initial location or selection. May be NULL if default location or selection is acceptable.
initialFileName	Ignored (may be NULL) for Open and ChooseFolder . For Save , filename portion is used for edit field. May be NULL on Windows, but is required on the Mac.
fileFilters	Array of AVFileFilterRecs . Ignored (may be NULL) for ChooseFolder . May be NULL for Open ONLY if kAVOpenSaveAllowAllFlag is set.
numFileFilters	Number of AVFileFilterRecs in fileFilters .
settingsComputeEnabledProc	(Optional) Called to determine whether the Settings button should be enabled. May be NULL . Ignored if kAVOpenSaveAllowSettingsButton is not set.
settingsExecuteProc	Called when the user clicks on the (enabled) Settings button. May be NULL . Ignored if kAVOpenSaveAllowSettingsButton is not set.
settingsProcData	Data that is passed to the settingsExecuteProc callback. Ignored if kAVOpenSaveAllowSettingsButton is not set.



AVPageIndex

```
typedef ASInt32 AVPageIndex;
```

Description

A page index value for use in an **AVDoc** view definition.

Header File

`AVExpT.h`

Related Methods

[AVDocGetViewDef](#)



AVPageSize

```
enum {
    kAVPageSizeUninitialized = 0,
    kAVPageSizeNone, /*do not adjust for size, pages cropped
by printer */
    kAVPageSizeFitToPaper,
    kAVPageSizeShrinkLargePages,
    kAVPageSizeTileLargePages,
    kAVPageSizeTileAllPages
};
typedef ASEnum16 AVPageSize;
```

Description

Constants that specify page size types for [AVDocPrintParams](#), introduced in Acrobat 6.0. If the **pageSize** field is not set, the old field of **shrinkToFit** is used instead.

Header File

AVExpT.h

Related Methods

[AVDocPrintPagesWithParams](#)

AVPageViewControlID

```
typedef ASEnum16 AVPageViewControlID;
```

Description

Constants used with [AVPageViewShowControl](#) to allow a client author to turn on and off the controls shown in the status area at the bottom of a page view.

Header File

AVExpT.h

Related Methods

[AVPageViewShowControl](#)

Values

kAVPageViewZoomControl	The zoom control.
kAVPageViewPageFlipControls	The page flip control.
kAVPageViewPageNumberControl	The page number control.
kAVPageViewPageSizeControl	The page size control.
kAVPageViewSplitterBar	The splitter bar control.
kAVPageViewHorizontalScrollBar	The horizontal scroll bar control.
kAVPageViewVerticalScrollBar	The vertical scroll bar control.
kAVPageViewGrayBorder	(New in Acrobat 5.0) The gray border control.



AVPixelFormat

```
typedef ASInt16 AVPixelFormat;
```

Description

A pixel offset value for use in an **AVDoc** view definition.

Header File

AVExpT.h

Related Methods

[AVDocGetViewDef](#)



AVPrefsType

Description

A structure containing values that specify the Acrobat viewer's preferences settings. Can contain up to 255 entries.

Header File

AVExpT.h

Related Methods

[AVAppGetPreference](#)

[AVAppSetPreference](#)

Members

avpAllowByteRangeRequests	ASBool	If true (default), Acrobat, in conjunction with your browser, will use HTTP byte range requests in order to download specific parts of the PDF file that have been requested. Byte range requests allow for a better user experience of large PDF files in the browser. <i>(Present only in version 5.0 and higher)</i>
avpAllowOpenFile	ASBool	If true , links or actions that might launch another application are allowed. If false , such links do nothing. Corresponds to the "Allow Open File" item in the General Preferences dialog box. See avpSecureOpenFile .
avpAntialiasGraphics	ASBool	Turns on or off anti-aliasing of graphics. Users can control the anti-aliasing of images and text from General Preferences with the Smooth Text & Images checkbox
avpAntialiasLevel	ASInt16	Point size above which to use anti-aliasing. <i>(Present only in version 3.0 and higher)</i>

avpAntialiasText	ASBool	Turns on or off anti-aliasing, that is, text smoothing. <i>(Present only in version 3.0 and higher)</i>
avpASExtensionDigCert		Currently unused.
avpBookmarkShowLocation	ASBool	If true , the bookmarks corresponding to the portion of the document currently being viewed are displayed in bold. If false , it is not.
avpBrowserCheck	ASBool	If true , the application will run the browser self-healing code at initialization time. This code determines if the system is properly configured to run Acrobat in the browser, and if it is not, the code will ask the user whether she would like to automatically correct the problems that were found. If false , browser self-healing code is skipped. <i>(Present only in version 5.0 and higher)</i>
avpBrowserIntegration	ASBool	If true , the PDFViewer client is used to view PDF files opened inside a browser window. If false , the viewer is launched as a helper application to view the file. <i>(Windows only)</i>
avpCacheFormData	ASBool	If true , causes Acrobat to temporarily remember values entered in form fields by writing the data to temporarily to disk. If false , Acrobat will not remember values. <i>(Present only in version 5.1 and higher)</i>
avpCaseSensitive	ASBool	If true , the Acrobat viewer's Find command (not the Search client) performs case-sensitive searches. If false , searches are not case-sensitive.

avpCurrCMM	char*	Currently unsupported. Tracks what the current CMM is, for example, Apple ColorSync, Kodak, and so forth. Users can control this with the Color Manager part of the General Preferences dialog box.
avpDefaultOverviewType	ASInt32	Whether thumbnail images, bookmarks, or neither should be shown along with documents by default. Must be one of the PDPageMode values.
avpDefaultSplitterPos	ASInt32	The default width (in pixels) of the portion of the document window in which bookmarks and thumbnail images are displayed. The actual width can be changed by the user or programmatically using AVDocSetSplitterPosition .
avpDefaultZoomScale	ASFixed	Default magnification when a document is opened.
avpDefaultZoomType	AVZoomType	Default zoom type for a page view. Must be one of the AVZoomType values.
avpDestFitType	char*	Currently unsupported. Default destination fit type used for bookmark and link creation. Must be one of the View Destination Fit Types .
avpDestZoomInherit	ASBool	If true , the default destination fit type for creating new links and bookmarks is "Inherit Zoom." If false , it is not.
avpDisableAcrobatUpdate	ASBool	Indicates whether the automatic Acrobat update feature is enabled (true) or disabled (false). (Present only in version 5.0 and higher)

avpDisableWebServicesUpdate	ASBool	Indicates whether the automatic Acrobat Web Services update feature is enabled (true) or disabled (false). (Present only in version 5.0 and higher)
avpDisplayRightsDialog	ASBool	If true , ubiquity rights dialog is shown (if document has rights). If false , dialog is not shown. (Present only in version 5.0 and higher)
avpDoCalibratedColor	ASBool	If true , the Acrobat viewer renders using calibrated color. The chromaticity and gammas used are those set using PDPrefSetColorCal . If false , do not.
avpDoUpdate	ASBool	If true , then Acrobat will do an automatic check for Acrobat Updates or Web Service updates if the last update value is larger than 1 week or 1 month, depending on the current setting for avpUpdateFrequency . If false , then Acrobat will not automatically check for Acrobat Updates or Web Service Updates. (Present only in version 5.0 and higher)
avpDownloadEntireFile	ASBool	If true , download entire PDF files in browser windows. If false , do not. (Present only in version 3.0 and higher)
avpDrawMissingThumbs	ASBool	If true , Acrobat will draw thumbnails for pages when no thumbnails are embedded in the document. If false , Acrobat will display gray rectangles for pages without embedded thumbnails. (Present only in version 5.0 and higher)

avpEmitHalftones	ASBool	If true , use halftones found in PDF files. If false , do not. (Present only in version 3.0 and higher)
avpEnablePageCache	ASBool	true if the following page is rendered offscreen while the current page is viewed, improving performance when a document is viewed sequentially, false if no draw-ahead is used.
avpFullScreenChangeTimeDelay	ASInt32	Time (in seconds) to show each page when using automatic page changing in full-screen mode.
avpFullScreenClick	ASBool	true if the page advances on a mouse click in full-screen mode, false otherwise. (Present only in version 3.0 and higher)
avpFullScreenColor	PDColorValue	The background color to use when the Acrobat viewer is in full-screen mode.
avpFullScreenCursor	ASInt16	Cursor behavior in full-screen mode (0 = visible, 1 = hidden, 2 = hidden after delay). (Present only in version 3.0 and higher)
avpFullScreenEscape	ASBool	true if the Escape key exits full-screen mode, false otherwise. (Present only in version 3.0 and higher)
avpFullScreenLoop	ASBool	true if the document's pages are displayed in a loop (rather than just once) when using full-screen mode, false otherwise.
avpFullScreenTransitionType	char*	Default transition name. (Present only in version 3.0 and higher)

avpFullScreenUsePageTiming	ASBool	true if page timings are used to advance pages in full-screen mode, false otherwise. (Present only in version 3.0 and higher)
avpFullScreenUseTimer	ASBool	true if the page advances automatically in full-screen mode, false otherwise. (Present only in version 3.0 and higher)
avpGreekLevel	ASInt32	Size, in points, below which text is greeked if avpGreekText is true .
avpGreekText	ASBool	If true , text smaller than avpGreekLevel is greeked (displayed as gray boxes). If false , all text is drawn, regardless of its size.
avpHideTabsCompletely	ASBool	If true , hide viewer tabs completely. If false , do not. (Present only in version 5.0 and higher)
avpHighlightColor	PDColorValue	Currently unsupported.
avpHighlightMode	ASInt32	Specifies the way in which highlighted text is displayed. Must be one of the following: HIGHLIGHT_PAINT_XOR — Paint highlight color in XOR mode HIGHLIGHT_INVERT_MAC — Invert in special Macintosh highlight mode HIGHLIGHT_INVERT_XOR — Invert in XOR mode This preference exists because the standard Macintosh highlighting generally works quite well, but can occasionally become invisible when text is on a colored background. (Used on Mac OS only)

avpHoveringPopups	ASBool	If true, enables the new “automatically open pop-ups on mouse-over” feature. This means that moving the mouse over an annotation will cause any popup associated with that annotation to appear. When the mouse exits the annotation, the popup will disappear. <i>(Present only in version 5.0 and higher)</i>
avpIgnorePageClip	ASBool	If true, the viewer ignores clipping paths that would cause a thin white border to appear around the edges of the page. If false, Acrobat honors all clipping paths.
avpLastAcrobatUpdateCheck	ASUns32	Indicates an arbitrary time stamp for the last Acrobat Update check. If the update check frequency is every week or every month and the last update check was more than 1 week or 1 month ago, Acrobat will automatically check for updates. A value of 1 indicates that Acrobat should check for updates at the next launch. <i>(Present only in version 5.0 and higher)</i>
avpLastWebServicesUpdateCheck	ASUns32	Indicates an arbitrary time stamp for the last Acrobat Web Services check. If the update check frequency is every week or every month and the last update check was more than 1 week or 1 month ago, Acrobat will automatically check for Web Service updates. A value of 1 indicates that Acrobat should check for updates at the next launch. <i>(Present only in version 5.0 and higher)</i>

avpMarkHiddenPages	ASBool	Mark hidden (for Presenter plug-in) pages with the appropriate mark. (Present only in version 3.0 and higher)
avpMaxCosDocCache	ASInt32	The maximum zoom factor at which pages will be cached. Pages viewed at a higher zoom factor will be not cached. A value of 1.0 corresponds to a zoom factor of 100%. NOTE: A platform-dependent minimum value for the cos cache is enforced. It is 2M for Windows and Unix and 500K for Mac. If a client attempts to set the maxCosDocCache below the minimum, the plugin's setting is ignored and the minimum is used instead.
avpMaxOpenDocuments	ASInt32	Maximum number of open documents allowed. (Present only in version 3.0 and higher)
avpMaxPageCacheBytes	ASInt32	The maximum number of bytes the page cache is allowed to occupy.
avpMaxPageCacheZoom	ASFixed	The maximum zoom factor at which pages will be cached. Pages viewed at a higher zoom factor will not be cached. A value of 1.0 corresponds to a zoom factor of 100%.
avpMaxThreadZoom	ASFixed	The maximum zoom factor that is automatically used when the Acrobat viewer enters "Follow Article" mode. A value of 1.0 corresponds to a zoom factor of 100%.
avpMinimizeBookmarks	ASBool	If true , hide or minimize the Bookmarks panel after the user has selected a bookmark. If false , do not.

avpMinPageCacheTicks	ASInt32	The minimum number of ticks needed to redraw a page before it will be cached. Pages that can be redrawn in less time will not be cached. A tick is 1/60 of a second.
avpNoteColor	PDColorValue	Default color to use for new notes.
avpNoteFontName	char*	Currently unsupported. Font name for text notes. <i>(Present only in version 3.0 and higher)</i>
avpNoteFontSize	ASInt32	Font size for text notes.
avpNoteLabel	char*	Currently unsupported. Default label to use for new notes.
avpNoteLabelEncoding	void*	Indicates the script code (Mac OS) or charset (Windows) that the avpNoteLabel is encoded in. In both Mac OS and Windows, these codes are integers even though the preference is typed as void* .
avpOpenAsPDFFilterIndex	ASInt32	Indicates which filter will be used when the "Open As Adobe PDF" dialog is popped again. Use -1 for All files. <i>(Present only in version 5.0 and higher)</i>
avpOpenDialogAtStartup	ASBool	If true , the File Open dialog box is displayed when the Acrobat viewer is launched without a document to open. If false , it is not.
avpOpenInPlace	ASBool	If true , cross-document links should open in the same window. If false , open the file in another window.

avpOpenNewDocument	ASBool	Controls the default selection for the radio button in the “Open As Adobe PDF” dialog. This dialog appears after you select, for example, a .txt file to be opened as a PDF. If true , the default selection will be “Create New Document”. (Present only in version 5.0 and higher)
avpOpenSaveAsFilterIndex	ASInt32	Indicates which filter will be used when the “Save As” dialog is popped again. Use -1 for All files. (Present only in version 5.0 and higher)
avpOverrideAcrobatUpdateURL	char*	NULL -terminated char* that overrides the default Acrobat Update website that is used to check for updates. This is probably only useful for enterprise installations that want to run their own Acrobat Update website. (Present only in version 5.0 and higher)
avpPageUnits	PageUnitsType	Current units.
avpPageViewLayoutMode	PDLayoutMode	Default layout mode.
avpPersistentCacheFolder	ASPathName	Location of the Acrobat cache, which is used while viewing PDF files on a slow file system.
avpPersistentCacheSize	ASInt32	Size of the Acrobat cache, which is used while viewing PDF files on a slow file system. (Present only in version 3.0 and higher)
avpPrefsVersion	ASInt32	The preferences file format version number. Read only.

avpPrintAnnots	ASBool	If true , prints annotations on the page (useful for summarize annotations that the user can control from the Print dialog box). If false , non-Forms annotations with appearances (that is, mark-up annotations) will never print.
avpPrintUsingWorkingSpaces	ASBool	Whether the user last printed with the "Apply Color Working Spaces" checkbox set. (Present only in version 5.0 and higher)
avpPSLevel — ASInt32	ASInt32	The PostScript language level to use when printing to a PostScript printer. Valid values are 1, 2, and 3.
avpRecentFile1 , avpRecentFile2 , avpRecentFile3 , avpRecentFile4	char*	Currently unsupported. Paths for most recently used files in recent file menu. (Present only in version 3.0 and higher)
avpRememberDialogs	ASBool	If true , the Acrobat viewer remembers the location of certain dialog boxes (such as the Find dialog box) and displays them in their previous location. If false , they are displayed in a default location.
avpSaveAsLinearized	ASBool	If true , save as linearized file when saving. If false , do not save as linearized.
avpSecureOpenFile	ASBool	If true , links or actions that might launch another application are not allowed. A client can set this preference to true to override the avpAllowOpenFile preference (that the user can control from the General Preferences dialog box). If false , such links are allowed.

avpSendFarEastFonts	ASBool	If true , Type0 fonts used in the PDF file are included in the PostScript file that is generated. If false , the fonts are not be downloaded and the printer must already have the appropriate font.
avpShortMenus	ASBool	If true , the Acrobat viewer displays short menus. If false , it displays long menus. See AVMenuItemNew for information on specifying whether a menu item should be visible only when long menus are shown. <i>(Ignored in Acrobat 3.0 or later)</i>
avpShowAnnotSequence	ASBool	If true , displays and prints annotation sequence numbers on annotations (useful for summarize annotations that the user can control from the Annotations Preferences dialog box). If false , do not.
avpShowHiddenAnnots	ASBool	If true , shows the annotations marked as hidden. If false , such annotations are not displayed. <i>(Present only in version 3.0 and higher)</i>
avpShowLargeImages	ASBool	If true , the Acrobat viewer displays large images. If false , gray boxes are shown in place of large images, reducing rendering time for pages with large images.
avpShowLeftToolBar	ASBool	Indicates whether the left toolbar is visible or invisible. In Windows, this controls the visibility of the toolbar at application launch. In Mac OS, this controls the visibility of the toolbar at the time a document is opened.
avpShowMenuBar	ASBool	Indicates whether the menubar is visible or invisible when the application first launches.

avpShowOverprint	ASBool	Indicates whether the Overprint Preview mode is on or off. When it is on, the viewer simulates the way overlapping inks will look when printed. <i>(Present only in version 6.0 and higher)</i>
avpShowSplashAtStartup	ASBool	If true , the Acrobat viewer splash screen is shown when the Acrobat viewer is launched. If false , it is not.
avpShowToolBar	ASBool	If true , the Acrobat viewer's toolbar is displayed. If false , it is not. The toolbar can also be shown and hidden by the user.
avpShowUpdateDialog	ASBool	Indicates whether the user wants to see the automatic update dialog at startup. Setting this to false means the dialog is not displayed—the update check will still occur silently in the background. <i>(Present only in version 5.0 and higher)</i>
avpShrinkToFit	ASBool	If true , pages are shrunk to fit the image-able area of the printer when printed. If false , pages are not shrunk to fit.
avpSkipWarnings	ASBool	If true , a warning dialog box is not displayed when a user deletes notes, bookmarks, links, pages, or thumbnails. If false , the dialog box is displayed.

<code>avpSubstituteFontType</code>	ASInt32	Determines whether sans serif, serif, or both substitution fonts are available when printing. Using only one substitution font type generally reduces the quality of font substitution, but may allow some files that require font substitution to print on PostScript printers that have very little memory. Valid values are: 0 — Use both sans serif and serif 1 — Use sans serif only 2 — Use serif only
<code>avpSuppressCSA</code>	ASBool	Used by the PostScript print code to determine whether to emit color space arrays for 4- or 1-component ICCBased color spaces. If true, do not emit CSAs, instead emit DefaultCMYK or DefaultGray, respectively. The user may control this in Acrobat via File - > DocInfo -> Prepress.
<code>avpThumbViewScale</code>	ASFixed	The scale at which thumbnail images are created. The Acrobat viewer's default is fixedEighth, creating thumbnail images whose linear dimensions are one-eighth those of the actual page.
<code>avpThumbViewTimeout</code>	ASInt32	Currently unsupported.
<code>avpTrustedMode</code>	ASInt32	The level of trust required by Acrobat. When set to a non-zero value, Acrobat only loads clients certified as "trusted", that is, clients that respect the security settings of the document. Passing in <code>avpTrustedMode</code> to AVAppGetPreference always returns the trust level in effect when Acrobat was launched; it does not reflect the last value set by AVAppSetPreference, which will take effect when Acrobat is next launched.

avpTrustedModeOverride	ASInt32	When set to a non-zero value, this becomes the trust level the next time Acrobat is launched. This trust setting does not persist across sessions.
avpUpdateFrequency	ASUns32	Whether Acrobat checks for updates every week, every month, or never. Valid values for this preference are: <pre>#define CheckEveryWeek 1 #define CheckEveryMonth 2 #define CheckManually 3</pre> The default is CheckEveryMonth
avpUseHostFont	ASBool	Currently unsupported.
avpUseLocalFonts	ASBool	If true , the viewer uses fonts installed on the user's system when displaying and printing documents. If false , the viewer only uses its local copy of the base 14 fonts when displaying documents. All other fonts will be fauxed or substituted.
avpUseLogicalPageNumbers	ASBool	If true , the page numbering information encoded in the document is used. If false , the viewer numbers pages using decimal numbers starting at 1.
avpUsingAreaBoxNames	ASBool	Not currently used. Should always have a value of false . Provides a way of honoring the TrimBox, Bleedbox, and ArtBox fields in the page directory of a PDF file. <i>(Present only in version 5.0 and higher)</i>
avpWholeWords	ASBool	If true , the Acrobat viewer's Find command (not the Search client) matches only whole words. If false , the partial words are also matched.

AV Print Mark Flags

Description

Bit-flag [ASUns32](#) constants for use with [AVDocPrintParams](#).

Header File

AVExpT.h

Values

kAVCropMarks	Emit crop marks.
kAVTrimMarks	Emit trim marks.
kAVBleedMarks	Emit bleed marks.
kAVRegMarks	Emit registration marks.
kAVColorBarMarks	Emit color bar marks.
kAVPageInfoMarks	Emit page information marks.
kAVEasternStyleMarks	Emit Eastern style marks (default is Western style).



AVRasterizeFlags

```
enum {
    kAVRasterizeNoFlags = 0,
    kAVRasterizeAllTextToOutlines = 1 << 0,
    kAVRasterizeAllStrokesToOutlines = 1 << 1,
    kAVRasterizeAllowComplexClipRegions = 1 << 2,
    kAVRasterizePreserveOverprint = 1 << 3
};
typedef ASUns32 AVRasterizeFlags;
```

Description

Constants that specify rasterization methods to use for printing. Used in [AVDocPrintRasterizeData](#), which is used in [AVDocPrintParams](#).

Header File

AVExpT.h

Related Methods

[AVDocPrintPagesWithParams](#)

AVRect

AVRectP

```
typedef struct _t_AVRect {  
    AVSDKDependentInteger left;  
    AVSDKDependentInteger top;  
    AVSDKDependentInteger right;  
    AVSDKDependentInteger bottom;  
} AVRect, *AVRectP;
```

Description

Data structure representing a rectangle (a quadrilateral having only horizontal and vertical sides).

The AcroView coordinate system is defined so that (0,0) is at the top, **x** increases to the right, and **y** increases down (the same as GDI and QuickDraw but opposite to the PostScript language). An **AVRect** is defined so that its **top** is above its **bottom**, but this means that $0 < \text{top} < \text{bottom}$.

NOTE: Types of numeric coordinate values have changed in Acrobat 6. These types are conditionally compiled as **ASInt16** or **ASInt32** depending on the Acrobat version level.

Header File

AVExpT.h

Related Methods

Numerous

Related Structures

[AVDevRect](#)

AVRect32

AVRect32P

```
typedef struct _t_AVRect32 {  
    ASInt32 left;  
    ASInt32 top;  
    ASInt32 right;  
    ASInt32 bottom;  
} AVRect32, *AVRect32P;
```

Description

Data structure representing a rectangle (a quadrilateral having only horizontal and vertical sides).

The AcroView coordinate system is defined so that (0,0) is at the top, x increases to the right, and y increases down (the same as GDI and QuickDraw but opposite to the PostScript language). An **AVRect32** is defined so that its top is above its bottom, but this means that $0 < \text{top} < \text{bottom}$.

NOTE: This structure supports 32-bit coordinates in versions prior to Acrobat 6.0. For version 6.0 and higher, **AVRect** is preferred.

Header File

AVExpT.h

Related Methods

Numerous



AVRectHandleType

```
typedef ASEnum8 AVRectHandleType;
enum {
    kAVRectHandleNone,
    kAVRectHandleTopLeft,
    kAVRectHandleTopRight,
    kAVRectHandleBottomRight,
    kAVRectHandleBottomLeft,
    kAVRectHandleTopMiddle,
    kAVRectHandleRightMiddle,
    kAVRectHandleBottomMiddle,
    kAVRectHandleLeftMiddle
};
```

Description

Constants that specify the types of **AVRect** handles.

Header File

AVExpT.h

Related Methods

Various



AVResourcePolicy

```
enum {
    kAVResPolicySendAtStart,
    kAVResPolicySendByRange,
    kAVResPolicySendPerPage
};typedef ASEnum16 AVResourcePolicy;
```

Description

Font and resource policy values for [AVDocPrintParams](#).

Header File

AVExpT.h

Related Methods

[AVDocPrintParams](#)

Values

kAVResPolicySendAtStart	Send fonts and resources used on more than one page at start of document. Type 0 fonts are still downloaded incrementally. Fonts and resources used on just one page are downloaded on just that page. Fonts are downloaded in global VM.
kAVResPolicySendByRange	Send fonts and resources on the first page they are used. Pages must be printed in order they are created. Fonts are downloaded in global VM. Fonts and resources are removed when they are no longer needed.
kAVResPolicySendPerPage	Send fonts and resources on each page they are used. Fonts are downloaded in local VM.



AVScreenCoord

```
typedef AVSDKDependentInteger AVScreenCoord;
```

Description

An x or y coordinate in the screen space. (0,0) is at the top left of the main monitor.

Header File

`AVExpT.h`

Related Methods



AVScreenRect

```
typedef AVRect AVScreenRect;
```

Description

Data structure representing a rectangle (a quadrilateral having only horizontal and vertical sides) in a screen's coordinate space.

Header File

`AVExpT.h`

Related Methods

[AVWindowGetFrame](#)

[AVWindowSetFrame](#)

Related Callbacks

[AVWindowDidResizeProc](#)

[AVWindowWillBeResizedProc](#)

Related Structures

[AVDevRect](#)

[AVWindowRect](#)

[AVDocOpenParams](#)

[AVToolBarPosition](#)



AVSDKDependentInteger

```
#if !defined(ACRO_SDK_LEVEL) || (ACRO_SDK_LEVEL < 0x00060000)
typedef ASInt16 AVSDKDependentInteger;
#else
typedef ASInt32 AVSDKDependentInteger;
#endif
```

Description

A conditionally-compiled numeric type. It is a 16-bit or 32-bit integer, depending on the version level.

Header File

AVExpT.h

Related Types

[AVDevCoord](#)
[AVDevSize](#)
[AVScreenCoord](#)
[AVWindowCoord](#)

AVSpecialCategory

```
typedef enum {
    kAVSCUser,
    kAVSCApp,
    kAVSCLast
} AVSpecialCategory;
```

Description

Categories of special folders on the system. Used with folder types to locate folders. Note that some combinations of **AVSpecialCategory** and [AVSpecialFolder](#) are not valid. See [AVSpecialError](#) for a list of valid combinations.

Header File

```
AVExpT.h
```

Related Methods

```
AVAcquireSpecialFilePathName
AVAcquireSpecialFolderPathName
```

Values

kAVSCUser	User folders
kAVSCApp	Application folders
kAVSCLast	Often used to tag the end of an enumeration with a specific value.

AVSpecialError

```
typedef enum {
    kAVSEOkay,
    kAVSEInvalidCombination,
    kAVSEDoesntExist,
    kAVSECouldntCreate,
    kAVSEError
} AVSpecialError;
```

Description

Operation status codes for the special folder methods.

Header File

AVExpT.h

Related Methods

[AVAcquireSpecialFilePathName](#)
[AVAcquireSpecialFolderPathName](#)

Values

kAVSEOkay	No error.
kAVSEInvalidCombination	Invalid category/folder combination. See Valid Combinations, below.
kAVSEDoesntExist	File or directory doesn't exist.
kAVSECouldntCreate	File system error: directory couldn't be created.
kAVSEError	Some other generic error.

Valid Combinations

kAVSCUser/kAVSFRoot
kAVSCUser/kAVSFBooks
kAVSCUser/kAVSFPreferences
kAVSCUser/kAVSFSequences
kAVSCUser/kAVSFMessages
kAVSCUser/kAVSFDocuments
kAVSCUser/kAVSFJavaScript
kAVSCUser/kAVSFStamps
kAVSCUser/kAVSFDictionaries
kAVSCApp/kAVSFRoot
kAVSCApp/kAVSFUpdate



kAVSCApp/kAVSFSequences
kAVSCApp/kAVSFJavaScript
kAVSCApp/kAVSFStamps
kAVSCApp/kAVSFDictionaries
kAVSCApp/kAVSFPlugIns
kAVSCApp/kAVSFSPPlugIns
kAVSCApp/kAVSFHelp
kAVSCApp/kAVSFMessages
kAVSCApp/kAVSFResource
kAVSCApp/kAVSFHelpLocale (Windows only)
kAVSCUser/kAVSFTemp
kAVSCUser/kAVSFHelpLocale (Windows only)



AVSpecialFolder

```
typedef enum {
    kAVSFRoot,
    kAVSFEBooks,
    kAVSFPreferences,
    kAVSFSequences,
    kAVSFDocuments,
    kAVSFJavaScript,
    kAVSFStamps,
    kAVSFDictionaries,
    kAVSFPlugIns,
    kAVSFSPPlugIns,
    kAVSFHelp,
    kAVSFTemp,
    kAVSFMessages,
    kAVSFResource,
    kAVSFUpdate,
    kAVSFHelpLocale,
    kAVSFAuthoring,
    kAVSFSecurity,
    kAVSFLocalRoot,
    kAVSFLocalCache,
    kAVSFTasks,
    kAVSFLast
} AVSpecialFolder;
```

Description

Special folder types on the system. Used with folder categories to locate folders. Note that some combination of [AVSpecialCategory](#) and **AVSpecialFolder** are not valid. See tables below for detailed information. See [AVSpecialError](#) for a list of valid combinations.

Header File

AVExpT.h

Related Methods

[AVAcquireSpecialFilePathName](#)
[AVAcquireSpecialFolderPathName](#)



Values

kAVSFRoot (see table below for which is used)	<i>User root</i> Windows: Documents and Settings/<user>/Application Data/Acrobat <i>Viewer root</i> Location of the application. Sometimes needed to sniff for DLLs and other files that live at this level. Windows: Program Files/Adobe/Acrobat 5.0/<viewer>
kAVSFEbooks	<i>User eBook license files (RMF)</i> License files used by WebBUY to indicate ownership/rights of particular PDF documents. Windows: Documents and Settings/<user>/Application Data/Acrobat/EBook
kAVSFPreferences	<i>User preferences folder</i> Should contain files full of preferences (for example, ini or Mac prefs files). Generally you don't want the user touching these. If a client wants to provide regular preferences it should use the <code>miIni.c</code> interface which would map to pref files in this folder on the Mac and the registry on Windows. Windows: Documents and Settings/<user>/Application Data/Acrobat/Preferences
kAVSFSequences (see table below for which is used)	<i>User-defined batch sequences</i> Custom batch scripts that the user has written/defined and saved. Windows: Documents and Settings/<user>/Application Data/Acrobat/Sequences <i>Application batch sequences</i> Batch sequences that are shipped with the product as examples. Windows: Program Files/Adobe/Acrobat 5.0/<viewer>/Sequences/<locale>

kAVSFDocuments	<i>User documents folder</i> The viewer and all clients should default to opening and saving in this folder. Windows: Documents and Settings/<user>/My Documents
kAVSFJavaScript (see table below for which is used)	<i>User JavaScripts folder</i> JavaScripts written by the user (.js files) that are loaded at application launch. These are editable directly by the user and as such are not usually found in the same folder as kAVSFEBooks, kAVSFPreferences, or kAVSFSequences. Windows: Documents and Settings/<user>/My Documents/Acrobat/JavaScript <i>Application JavaScripts folder</i> JavaScripts shipped with the product (AForm.js, AFString<lang>.js, Annots.js) Windows: Program Files/Adobe/Acrobat 5.0/<viewer>/clients/EScript/JavaScripts
kAVSFStamps (see table below for which is used)	<i>User stamps folder</i> Custom rubber stamps that the user creates are stored in this folder. Windows: Documents and Settings/<user>/My Documents/Acrobat/Stamps <i>Application stamps folder</i> Stamps that are shipped with the product. Windows: Program Files/Adobe/Acrobat 5.0/<viewer>/clients/Annotations/Stamps
kAVSFDictionaries (see table below for which is used)	<i>User-installed dictionaries</i> User-installed dictionaries for the spell checker. Windows: Documents and Settings/<user>/My Documents/Dictionaries <i>Application-installed dictionaries</i> Dictionaries that are shipped with the product. Windows: Program Files/Adobe/Acrobat 5.0/<viewer>/clients/Spelling/Dictionaries

kAVSFPlugIns	<i>Application clients folder</i> Where clients are stored. Windows: Program Files/Adobe/Acrobat 5.0/<viewer>/clients
kAVSFSPPlugIns	Suite Pea clients folder.
kAVSFHelp	<i>Help folder</i> Application help. Windows: Program Files/Adobe/Acrobat 5.0/Help
kAVSFTemp	<i>Temporary folder</i> Windows: Windows\Temp
kAVSFMessages (see table below for which is used)	<i>User messages folder</i> Windows: Documents and Settings/<user>/Application Data/Acrobat/Messages <i>Application messages folder</i> Windows: Program Files/Adobe/Acrobat 5.0/Messages
kAVSFResource	<i>Resources folder</i>
kAVSFUpdate	<i>Update folder</i>
kAVSFHelpLocale (see table below for which is used)	<i>Downloaded Adobe Reader Help folder</i> Used to store the full Adobe Reader help file on systems that lock out access to the application help folder. Windows: Documents and Settings/<user>/Application Data/Acrobat/Help/<Locale> <i>Application help locale folder</i> Windows: Program Files/Adobe/Acrobat 5.0/Help/<Locale>
kAVSFAuthoring	<i>Authoring folder</i>
kAVSFSecurity	<i>Security folder</i>
kAVSFLocalRoot	<i>Local root folder</i>
kAVSFLocalCache	<i>Local cache folder</i>
kAVSFTasks	<i>Tasks folder</i>

	kAVSCUser (see AVSpecialCategory)	kAVSCApp (see AVSpecialCategory)
kAVSFRoot	User root	Acrobat root
kAVSFEBooks	User eBook license files (RMF)	N/A
kAVSFPreferences	User preferences folder	N/A
kAVSFSequences	User-defined batch sequences	Application batch sequences
kAVSFDocuments	User documents folder	N/A
kAVSFJavaScript	User JavaScripts	Application JavaScripts
kAVSFStamps	User stamps folder	Application stamps folder
kAVSFDictionaries	User-installed dictionaries	Application-installed dictionaries
kAVSFPlugIns	N/A	Clients folder
kAVSFHelp	N/A	Help folder
kAVSFTemp	Temporary folder	N/A
kAVSFMessages	User messages folder	Default messages folder
kAVSFHelpLocale	Downloaded Adobe Reader Help folder	Application help locale folder

AVStatusMonitorProcs

AVStatusMonitorProcsRec

```
typedef struct _t_AVStatusMonitorProcs {
    ASSize\_t size;
    ASProgressMonitor progMon;
    void *progMonClientData;
    ASCancelProc cancelProc;
    void *cancelProcClientData;
    ASReportProc reportProc;
    void *reportProcClientData;
} AVStatusMonitorProcsRec, *AVStatusMonitorProcs;
```

Description

A structure that contains a progress monitor, cancel procedure, and error report procedure.

Header File

AVExpT.h

Related Methods

[AVConversionConvertToPDFWithHandler](#)
[AVConversionConvertFromPDFWithHandler](#)
[AVConversionConvertToPDF](#)

Members

size	The size of this structure. Set to sizeof (AVStatusMonitorProcsRec) .
progMon	(May be NULL) The progress monitor. In general, clients of this structure test members for NULL . If a member is found, they do not do anything.
progMonClientData	The progress monitor client data that was acquired with the progress monitor.
cancelProc	(May be NULL) The cancellation procedure. In general, clients of this structure test members for NULL . If a member is found, they do not do anything.
cancelProcClientData	The cancellation procedure client data that was acquired with the cancellation procedure.
reportProc	(May be NULL) The report procedure. In general, clients of this structure test members for NULL . If a member is found, they do not do anything.



reportProcClientData	The report procedure client data that was acquired with the report procedure.



AVStructNode

Description

An opaque object representing a node in a document structure tree.

Header File

AVExpT.h

Related Callbacks

[AVConversionConvertStreamFromStructNodeProc](#)

Related Methods

[AVConversionConvertStreamFromStructNodeWithHandler](#)



AVSysIconType

```
enum
{
    kAVSysSmallIcon,
    kAVSysLargeIcon
};
typedef ASEnum8 AVSysIconType;
```

Description

Constants that specify the type of an [AVIcon](#).

Header File

AVExpT.h

Related Methods

[AVSysGetIconFromFilename](#)
[AVSysGetIconFromMimeType](#)
[AVSysGetIconFromTypeAndCreator](#)



AVSystemFont

AVSystemFontRec

```
typedef struct _t_AVSystemFont {
    short fondID;
    short styleID;
    ASUns32 flags;
    ASAtom pdfFontName;
} AVSystemFontRec, *AVSystemFont;
```

Description

(Macintosh only, present only in version 3.0 or later) System font.

Header File

AVExpT.h

Related Methods

[AVAppEnumSystemFonts](#)

Members

fondID	Macintosh FOND id.
styleID	Macintosh style value: normal, italic, bold or bold italic.
flags	Font flags. Must be one of AVSystemFont Flags .
pdfFontName	PostScript name or TrueType styled name.



AVSystemFont Flags

Description

Constants that specify the attributes of an [AVSystemFont](#).

Header File

AVExpT.h

Related Methods

[AVAppEnumSystemFonts](#)

Values

kFontIsType1	Type 1 font.
kFontIsMMMaster	Multiple Master font.
kFontIsMMInstance	Multiple Master font (completely-specified instance).
kFontIsTrueType	True type font.
kFontIsCIDFontType0	Character ID Type 0 font.
kFontIsOCFType1	OCF Type 1 font.



AVTArraySize

```
typedef ASInt32 AVTArraySize;
```

Description

The number of items (not the number of bytes) in an array, for use in page view methods.

Header File

```
ASExpT.h
```

Related Methods

```
AVAppGetNumDocs  
AVDocGetClientName  
AVMenubarGetNumMenus  
AVMenuGetNumMenuItems  
AVMenuGetTitle  
AVMenuItemGetTitle  
AVPageViewDrawRectOutline  
AVPageViewDrawRectOutlineWithHandles  
AVPageViewGetThreadIndex  
AVToolBarGetNumButtons  
AVUtilGetBaseNameAndExtensionByPathName  
AVUtilGetBaseNameAndExtensionByString
```

Related Callbacks

```
AVActionGetInstructionsProc  
AVActionGetButtonTextProc  
AVActionGetStringOneTextProc  
AVActionGetStringTwoTextProc
```



AVTBufferSize

```
typedef ASInt32 AVTBufferSize;
```

Description

The number of bytes in a buffer, for use in page view methods.

Header File

`ASExpT.h`

Related Methods

[AVAppGetNumDocs](#)



AVTCount

```
typedef ASInt16 AVTCount;
```

Description

A click-number value for use in page-view callback procedures.

Header File

AVExpT.h

Related Methods

[AVPageViewClickProc](#)

[AVPageViewKeyDownProc](#)



AVTFlagBits

```
typedef ASInt32 AVTFlagBits;
```

Description

A flag-bits value for use in page-view methods.

Header File

AVExpT.h

Related Methods

[AVPageViewDeviceRectToPageRZ](#)
[AVPageViewDragOutNewRectSnapped](#)
[AVPageViewDragRectSnapped](#)
[AVPageViewDragRectSnappedEx](#)



AVTFlagBits16

```
typedef ASInt16 AVTFlagBits;
```

Description

A flag-bits value for use in tool button methods.

Header File

`AVExpT.h`

Related Methods

[AVToolButtonSetExternal](#)



AVTileMark

```
enum {  
    kAVTileMarkNone = 0,  
    kAVTileMarkWestern,  
    kAVTileMarkEastern  
};  
typedef ASEnum16 AVTileMark;
```

Description

Constants that specify tile marking styles for [AVDocPrintTileData](#), used in [AVDocPrintParams](#).

Header File

AVExpT.h

Related Methods

[AVDocPrintPagesWithParams](#)

AVTool

AVToolRec

```
typedef struct _t_AVTool {
    ASSize_t size;
    ActivateProcType Activate;
    DeactivateProcType Deactivate;
    DoClickProcType DoClick;
    AdjustCursorProcType AdjustCursor;
    DoKeyDownProcType DoKeyDown;
    GetTypeProcType GetType;
    IsPersistentProcType IsPersistent;
    ASInt32 cursorID;
    AVComputeEnabledProc ComputeEnabled;
    void* computeEnabledData;
    /* Added in Acrobat 4.0 */

    DoLeaveProcType DoLeave;
    GetSelectionServerProcType GetSelectionServer;
    /* Added in Acrobat 6.0 */
    DoClickProcType DoClick;
    AdjustCursorProcType AdjustCursor;
    DoClickProcType DoRightClick;
    void *clientData;
    AVToolDestroyProc Destroy;
    AVToolGetLabelProc GetLabel;
    AVToolGetLabelIconProc GetLabelIcon;
} AVToolRec, *AVTool;
```

Description

Data structure for a tool. It contains callbacks that implement the tool's functions (for example, handling mouse clicks and controlling the cursor shape).

Header File

AVExpT.h

Related Methods

[AVAppGetActiveTool](#)
[AVAppSetActiveTool](#)
[AVAppGetDefaultTool](#)
[AVAppGetLastActiveTool](#)
[AVAppGetToolByName](#)
[AVAppRegisterTool](#)
[AVToolGetType](#)
[AVToolIsPersistent](#)



Members

size	Size of the data structure. Must be set to sizeof (AVToolRec) .
cursorID	A default cursor, used if the tool does not have an AdjustCursorProcType callback.
computeEnabledData	User-supplied data to pass to the tool's AVComputeEnabledProc callback each time it is called.

AVToolBarDockPosition

```
typedef AEnum8 AVToolBarDockPosition;
enum {
    kAVToolBarDockTop,
    kAVToolBarDockBottom,
    kAVToolBarDockLeft,
    kAVToolBarDockRight,
    kAVToolBarFloating
};
```

Description

Constant values that specify toolbar positions for registering the preferred position of a toolbar.

Header File

AVExpT.h

Related Methods

[AVAppRegisterToolBarPosition](#)



AVToolBarLayout

```
typedef AEnum8 AVToolBarLayout;  
enum {  
    kAVToolBarHorizontal,  
    kAVToolBarVertical,  
    kAVToolBarTwoColumn  
};
```

Description

Constant values that specify toolbar layouts.

Header File

AVExpT.h

Related Methods

[AVAppRegisterToolBarPosition](#)

AVToolBarPosition

AVToolBarPositionRec

```
typedef struct _t_AVToolBarPosition {
    ASSize_t size;
    ASBool inDoc;
    AVToolBarDockPosition dockPosition;
    const char *floatingWindowName;
    ASInt32 stackNum;
    ASInt32 offset;
    ASInt32 order;
    AVScreenRect windowFrame;
    AVToolBarLayout layout;
    ASBool hidden;
    ASBool windowHidden;
} AVToolBarPositionRec, *AVToolBarPosition;
```

Description

A structure that describes the position of a toolbar.

NOTE: Numeric coordinate types have changed in Acrobat 6.0.

Header File

AVExpT.h

Related Methods

[AVAppRegisterToolBarPosition](#)

Members

size	The size of this structure. Set to sizeof (AVToolBarPositionRec) .
inDoc	Should always be set to false .
dockPosition	The edge of the document window or monitor to attach this toolbar to. (On Macintosh the toolbar can only be docked to the top edge of the main monitor.)
floatingWindowName	If the toolbar is to be floating you can group it with another toolbar by specifying a name for the floating window. You can set this to a constant string.

stackNum	Toolbars are arranged in stacks; that is rows for horizontal toolbars or columns for vertical toolbars. stackNum = <i>n</i> adds the toolbar to the end of the <i>n</i>th stack (where 0 is the topmost or leftmost stack). stackNum = -1 opens a new stack on the top or leftmost stack. stackNum = ASMAXInt32 opens a new stack on the rightmost or bottom stack.
offset	The number of pixels from the top or left edge of the stack from which to position the toolbar. If ASMAXInt32, the toolbar will be positioned snugly behind other toolbars on the stack. If -1, it will be positioned at the front.
order	If multiple positions specify an offset of -1 or ASMAXInt32, this field is used to further order them. It controls the order in which the bars will be placed, not the visual order on-screen. If, for example, two bars have an offset of -1, the one associated with the value in the lower order field will be positioned first at the front of the bar. Then the one associated by the value in the higher order field also will be positioned at the front of the bar, but pushing the first one to the right.
windowFrame	If the toolbar is not inDoc and dockPosition is floating, you may end up creating a new window. This specifies the frame of that window.
layout	If a new window is called for, the layout of that window.
hidden	true if the toolbar should be hidden by default.
windowHidden	true if the floating window in which the toolbar is located should be hidden by default.

AVToolButtonLabelPriority

```
enum {
    kAVButtonPriorityOffExtraLow = 100,
    kAVButtonPriorityOffLow = 200,
    kAVButtonPriorityOffNormal = 300,
    kAVButtonPriorityOffHigh = 400,
    kAVButtonPriorityOffExtraHigh = 500,
    kAVButtonPriorityOnExtraLow = 600,
    kAVButtonPriorityOnLow = 700,
    kAVButtonPriorityOnNormal = 800,
    kAVButtonPriorityOnHigh = 900,
    kAVButtonPriorityOnExtraHigh = 1000,
    kAVButtonPriorityAlwaysOn = 1100
};
typedef ASEnum16 AVToolButtonLabelPriority;
```

Description

A set of priority values for a tool bar button label text. The priority determines the preference order in which labels are shown when a toolbar is too short to hold all of the button labels.

Header File

AVExpT.h

Related Methods

[AVToolButtonGetLabelText](#)

[AVToolButtonSetLabelText](#)

AVTransHandler

AVTransHandlerRec

```
typedef struct _t_AVTransHandler {
    ASSize_t size;
    AVTransHandlerGetTypeProc GetType;
    AVTransHandlerExecuteProc Execute;
    AVTransHandlerGetUINameProc GetUIName;
    AVTransHandlerGetItemUINameProc GetItemUIName;
    AVTransHandlerInitTransDictProc InitTransDict;
    AVTransHandlerCompleteTransDictProc CompleteTransDict;
    /* The following callbacks are for non-standard clients
    that have additional information */
    AVTransHandlerDoPropertiesProc DoProperties;
    AVTransHandlerGetInstructionsProc GetInstructions;
    AVTransHandlerGetButtonTextProc GetButtonText;
    AVTransHandlerGetStringOneTextProc GetStringOneText;
    AVTransHandlerGetStringTwoTextProc GetStringTwoText;
} AVTransHandlerRec, *AVTransHandler;
```

Description

Data structure containing callbacks that implement a transition handler. The callbacks implement the transition handler functions.

Header File

AVExpT.h

Related Methods

[AVAppRegisterTransHandler](#)

Members

size	Size of the data structure. Must be set to sizeof (AVTransHandlerRec) .

AVTransitionPort

AVTransitionPortRec

```
typedef struct _t_AVTransitionPort {
    void *reserved;
    AVRect32 rect32; /* Reserved. Do not alter. */
    /* The following two fields vary by platform */
    #if WIN_PLATFORM
        HDC hDC;
        RECT rect;
    #elif MAC_PLATFORM
        GWorldPtr port;
        Rect rect;
    #elif UNIX_PLATFORM
        void* port;
        void* rect;
    #endif
} AVTransitionPortRec *AVTransitionPort;
```

Description

Platform-dependent data structure for a transition.

In general, a transition port specifies a bitmap and a rectangle describing the portion of the bitmap affected by the transition. The transition handler's

[AVTransHandlerExecuteProc](#) callback must copy all the bits from the source port's bitmap within the source port's rectangle to the area in the destination port's bitmap described by the destination port's rectangle. The source and destination ports' rectangles are guaranteed to be equal in size.

Header File

AVExpT.h

Related Callbacks

[AVTransHandlerExecuteProc](#)



AVTSmallArraySize

```
typedef ASInt16 AVTSmallArraySize;
```

Description

An array size value for use in page view methods.

Header File

`ASExpT.h`

Related Methods

[AVPageViewDragOutNewRectSnapped](#)
[AVPageViewDragRectSnapped](#)

Related Structures

[AVDocPrintParams](#)
[AVLine](#)



AVTVersionNumPart

```
typedef ASInt16 AVTVersionNumPart;
```

Description

An version-number part.

Header File

`ASExpT.h`

Related Methods

[AVAppGetVersion](#)

AVUndoHandler

```
typedef struct _t_AVUndoHandler {
    ASize\_t size;
    const char *type;
    AVUndoVerifyProc VerifyUndo;
    AVUndoExecuteProc Undo;
    AVUndoVerifyProc VerifyRedo;
    AVUndoExecuteProc Redo;
    AVUndoGetTitleProc GetUndoTitle;
    AVUndoGetTitleProc GetRedoTitle;
    AVUndoReleaseProc Release;
    AVUndoBeginEndProc BeginUndoRedo;
    AVUndoBeginEndProc EndUndoRedo;
} AVUndoHandlerRec, *AVUndoHandler;
```

Description

Contains callback procedure for an [AVUndo](#) record that perform the undo and redo operations.

Header File

AVExpT.h

Related Methods

[AVDocBeginUndoOperation](#)
[AVDocClearUndos](#)
[AVDocEndUndoOperation](#)
[AVDocGetTopUndo](#)
[AVUndoGetType](#)
[AVUndoNew](#)

Members

size	Size of the data structure. Must be set to <code>sizeof (AVUndoHandlerRec)</code> .
type	The type of the undo record. Any client-defined string, which can be matched for retrieval by AVDocGetTopUndo .



AVUndoHandlerData

```
typedef void* AVUndoHandlerData;
```

Description

Private data for use by callbacks in the [AVUndoHandler](#).

Header File

AVExpT.h

Related Methods

[AVUndoGetData](#)
[AVUndoNew](#)
[AVUndoSetData](#)



AVUseValue

```
typedef ASEnum16 AVUseValue;
```

Description

Constants that determine whether to let Acrobat decide whether to use a feature, or whether to force Acrobat to use or not to use the feature. Used in [AVDocPrintOverrideData](#), which is used in [AVDocPrintParams](#).

Header File

AVExpT.h

Related Methods

[AVDocPrintPagesWithParams](#)

Members

kAVUseAuto	Let Acrobat decide whether to use a feature.
kAVuse	Force Acrobat to use a feature, even if Acrobat thinks it will not work.
kAVnoUse	Do not use a feature, even if Acrobat thinks it will work.



AVWindowCoord

```
typedef AVSDKDependentInteger AVWindowCoord;
```

Description

An x or y coordinate in the window's space. (0,0) is at the top left, and units are in pixels.

Header File

`AVExpT.h`

Related Callbacks

[AVWindowAdjustCursorProc](#)

[AVWindowMouseDownProc](#)



AVWindowRect

```
typedef AVRect AVWindowRect;
```

Description

Data structure representing a rectangle (a quadrilateral having only horizontal and vertical sides) in a window's coordinate space.

Header File

`AVExpT.h`

Related Methods

[AVWindowGetInterior](#)

[AVWindowInvalidateRect](#)

Related Callbacks

[AVWindowDrawProc](#)

Related Structures

[AVDevRect](#)

[AVScreenRect](#)



AVWindow Flags

Description

Constants that specify the attributes of an [AVWindow](#) when it is created.

Header File

AVExpT.h

Related Methods

[AVWindowNew](#)

[AVWindowNewFromPlatformThing](#)

Members

AVWIN_RESIZEABLE	Window can be resized. UNIX — Under OpenWindows, all windows have resize corners regardless of the setting of this flag.
AVWIN_HASCLOSEBOX	Window has a close box.
AVWIN_HASZOOMBOX	Window has a zoom box. UNIX — This flag only adds a Maximize decoration for non-floating windows under mwm, not for floating or modal windows. No instruction regarding zoom boxes is set for OpenWindows; the “Full Size” instruction only makes windows full-height, not full-width.
AVWIN_HASMINIMIZEBOX	Window has a minimize box.
AVWIN_HASDRAGBAR	Window has a drag bar.
AVWIN_AUXILIARY	Window is an <i>auxiliary</i> window. Such windows pass menu commands that have not been handled to the front-most document window. For example, the Find dialog box is auxiliary; menu items like Save , Close , and so on should still be enabled. A window that does not purport to operate on the topmost document window (for example, an “Establish Connection to Library of Congress” dialog) would probably not be auxiliary.
AVWIN_WANTSKEY	The Acrobat viewer may freely assign key status to this window, if appropriate. At any time after the window has been created, this flag may be set or cleared using AVWindowSetWantsKey .
AVWIN_SMALLTITLEBAR	(New for Acrobat 5.0) Window has a small title bar.

AVWindowHandler

```
typedef struct _t_AVWindowHandler {
    ASSize_t size;
    AVWindowMouseDownProc MouseDown;
    AVWindowWillCloseProc WillClose;
    AVWindowDidCloseProc DidClose;
    AVWindowDidActivateProc DidActivate;
    AVWindowDidBecomeKeyProc DidBecomeKey;
    AVWindowKeyDownProc KeyDown;
    AVWindowWillResignKeyProc WillResignKey;
    AVWindowWillDeactivateProc WillDeactivate;
    AVWindowDrawProc Draw;
    AVWindowWillBeResizedProc WillBeResized;
    AVWindowPerformEditOpProc PerformEditOp;
    AVWindowCanPerformEditOpProc CanPerformEditOp;
    AVWindowAdjustCursorProc AdjustCursor;
    AVWindowDidResizeProc DidResize;
    /* New in Acrobat 4.0 */
    AVWindowDestroyPlatformThingProc DestroyPlatformThing;
} AVWindowHandlerRec, *AVWindowHandler;
```

Description

Data structure containing callbacks that implement a window handler. The callbacks implement the window handler functions. For example, resize the window, draw its contents, handle mouse clicks, handle key presses. **NULL** values are acceptable; default behavior is then used.

Header File

AVExpT.h

Related Methods

[AVWindowNew](#)

[AVWindowNewFromPlatformThing](#)

Members

size	Size of the data structure. Must be set to <code>sizeof (AVWindowHandlerRec)</code> .



AVWindowLayer

Description

Constants that specify the layer in which a newly-created [AVWindow](#) is to appear.

Header File

AVExpT.h

Related Methods

- [AVWindowNew](#)
- [AVWindowNewFromPlatformThing](#)

Members

AVWLnonfloating	Regular window, such as a document window.
AVWLfloating	Floating window, such as a palette.
AVWLmodal	Modal dialog.
AVWLpopup	(New in Acrobat 5.0) Tool-tip window with a one-pixel border.



AVZoomType

Description

Constants that specify the zoom strategy that Acrobat is to follow.

Header File

AVExpT.h

Related Methods

[AVPageViewZoomTo](#)

[AVPageViewGetZoomType](#)

Members

AVZoomNoVary	No variable zoom (that is, zoom is a fixed value such as 1.0 for 100%).
AVZoomFitPage	Fits the page to the window.
AVZoomFitWidth	Fits the page width to the window.
AVZoomFitHeight	Fits the page height to the window.
AVZoomFitVisibleWidth	Fits the width of the portion of the page upon which marks are made to the window.
AVZoomPreferred	Uses the page's preferred zoom.
AVZoomReflowWidth	<i>(New in Acrobat 5.0)</i> Reflow page to window width.



CFURLRefRec_Ptr

```
typedef struct _t_CFURLRefRec {  
    const struct __CFURL *url;  
} CFURLRefRec;  
typedef CFURLRefRec *CFURLRefRec_Ptr;
```

Description

A structure containing the equivalent of a Mac OS platform-specific **CFURLRef**.

Header File

ASExpT.h

Related Methods

[ASPlatformPathGetCFURLRefRecPtr](#)

Character Type Codes

Description

Constants that specify a character's type (uppercase, lowercase, number, punctuation, and so forth).

Header File

`PDExpT.h`

Related Methods

[PDWordGetCharacterTypes](#)

[PDWordSplitString](#)

[PDWordFilterString](#)

Values

Character Type	Description
<code>W_CNTL</code>	A control code.
<code>W_LETTER</code>	A lowercase letter.
<code>W_UPPERCASE</code>	An uppercase letter.
<code>W_DIGIT</code>	A digit.
<code>W_PUNCTUATION</code>	A punctuation mark.
<code>W_HYPHEN</code>	A hyphen.
<code>W_SOFT_HYPHEN</code>	A hyphen that is only present because a word is broken across two lines of text.
<code>W_LIGATURE</code>	A ligature.
<code>W_WHITE</code>	A white space glyph.
<code>W_COMMA</code>	A comma (commas and periods are treated separately from other punctuation marks because they are used both as word punctuation marks and as de-limiters in numbers, and need to be treated differently in the two cases).
<code>W_PERIOD</code>	A period.
<code>W_ACCENT</code>	An accent mark.
<code>W_UNMAPPED</code>	A glyph that cannot be represented in the destination font encoding.
<code>W_END_PHRASE</code>	An end-of-phrase glyph (for example, ":", "?", "!", ":", and ";").



Character Type	Description
W_WILD_CARD	A wildcard glyph (for example, "*" and "?") that should not be treated as a normal punctuation mark.
W_WORD_BREAK	A glyph that acts as a delimiter between words (see Glyph Names of Word Separators).

Code Page Values

```
enum{
    /* Windows code pages */
    kPDCodePageWinEastEuropeanRoman = 1250,
    kPDCodePageWinCyrillic = 1251,
    kPDCodePageWinGreek = 1253,
    kPDCodePageWinTurkish = 1254,
    kPDCodePageWinHebrew = 1255,
    kPDCodePageWinArabic = 1256,
    kPDCodePageWinBaltic = 1257,

    /* Macintosh pseudo code pages */
    kPDCodePageMacCentralEuropean = -9994,
    kPDCodePageMacCroatian = -9993,
    kPDCodePageMacRomanian = -9992,
    kPDCodePageMacCyrillic = -9991,
    kPDCodePageMacUkrainian = -9990,
    kPDCodePageMacGreek = -9989,
    kPDCodePageMacTurkish = -9988,
    kPDCodePageMacHebrew = -9987,
    kPDCodePageMacArabic = -9986
};
```

Description

Code page character-mapping constants defined for [PDSysEncodingCreateFromCodePage](#).

Header File

PEExpT.h

Related Methods

[PDSysEncodingCreateFromCodePage](#)

Related Structures

[PDEFontCreateFromSysFontParams](#)

Command Keys

Description

Constants indicating the strings to be used as keys when accessing [AVCommand ASCabs](#).

Header File

AVExpT.h

Related Types

None

Strings

kAVCommandKeyPDDoc	Use the string AVDoc as a key.
kAVCommandKeyAVDoc	Use the string PDDoc as a key.
kAVCommandKeyBaseFileName	Use the string BaseFileName as a key.
kAVCommandKeyOrigExtension	Use the string OrigExtension as a key.
kAVCommandKeyGenericTitle	Use the string GenericTitle as a key.
kAVCommandKeyTitle	Use the string Title as a key.
kAVCommandKeyParams	Use the string Params as a key.
kAVCommandKeyParamsDesc	Use the string ParamsDesc as a key.
kAVCommandKeyCanDescribeParams	Use the string CanDescribeParams as a key.
kAVCommandKeyCanBatch	Use the string CanBatch as a key.
kAVCommandKeyCanShowDialog	Use the string CanShowDialog as a key.
kAVCommandKeyGroupTitle	Use the string GroupTitle as a key.



CosByte

```
typedef ASUns8 CosByte;
```

Description

Used for an array of bytes in [CosDocGetID](#).

Header File

`CosExpT.h`

Related Methods

[CosDocGetID](#)



CosByteMax

typedef [ASInt32](#) CosByteMax

Description

Return value for [CosDecryptGetMaxKeyBytes](#) and [CosEncryptGetMaxKeyBytes](#). Value is -1 on error.

Header File

CosExpT.h

Related Methods

[CosDecryptGetMaxKeyBytes](#)

[CosEncryptGetMaxKeyBytes](#)

CosDocOpenParams

```
typedef struct _t_CosDocOpenParams {
    ASSize_t size;
    ASFlagBits openFlags;
    ASFileSys fileSys;
    ASPathName pathName;
    const char* headerString;
} CosDocOpenParamsRec, *CosDocOpenParams;
```

Description

Parameters used when saving a file using [CosDocOpenWithParams](#).

Header File

CosExpT.h

Related Methods

[CosDocOpenWithParams](#)

Members

size	Size of the data structure. Must be set to <code>sizeof (CosDocOpenParamsRec)</code> .
openFlags	Bit field of flags indicating how to open the file. <code>kCosDocOpenDoRepair</code> — Repair the file if necessary.
fileSys	The <code>ASFileSys</code> with which the file is opened. May be <code>NULL</code> if using the default file system.
pathName	<i>(Required)</i> The <code>ASPathName</code> of the file to open.
headerString	Expected header string in file, for example, “%FDF-”; if <code>NULL</code> , assumes “%PDF-”

CosDocSaveFlags

```
typedef ASFlagBits CosDocSaveFlags;
```

Description

Bit flags that specify the attributes of a **CosDoc** when it is saved.

Header File

CosExpT.h

Related Methods

[CosDocSaveToFile](#)

[CosDocSaveWithParams](#)

Flag Constants

cosSaveGarbageCollect	When set, delete un-referenced objects before the save.
cosSaveFullSave	When set, write all objects, not just changes.
cosSaveBinaryOK	When set, binary data can be stored in the file.
cosSaveConcealObjStreams	When set, write any object streams in a way that is hidden from PDF 1.4 and earlier viewers. Used for hybrid files, for example.

CosDocSaveParams

CosDocSaveParamsRec

```
typedef struct _t_CosDocSaveParams {
    ASize\_t size;
    char* header;
    char* cryptData;
    ASTArraySize cryptDataLen;
    ProgressMonitor mon;
    void* monClientData;
    CosDocOpenParams cryptVersion;
} CosDocSaveParamsRec, *CosDocSaveParams;
```

Description

Parameters used when saving a file using [CosDocSaveToFile](#) and [CosDocSaveWithParams](#).

Header File

CosExpT.h

Related Methods

[CosDocSaveToFile](#)
[CosDocSaveWithParams](#)

Members

size	Size of the data structure. Must be set to sizeof (CosDocSaveParamsRec) .
header	Complete header string, such as, "%FDF-1.0".
cryptData	The encryption key to pass into the PDCryptHandler if security has been set on the document.
cryptDataLen	Length of the encryption key, in bytes. Cannot be greater than 5.
mon	Progress monitor. Use AVAppGetDocProgressMonitor to obtain the default progress monitor.
monClientData	Pointer to user-supplied data to pass to mon each time it is called.
cryptVersion	The Cos cryptographic version—the version of the algorithm that is used to encrypt and decrypt document data. cryptVersion equal to 0 is treated as cryptVersion equal to 1 to maintain backward compatibility.

CosType

```
enum {  
    /* Scalar types */  
    CosNull = 0,  
    CosInteger = 1,  
    CosFixed = 2,  
    CosBoolean = 3,  
    CosName = 4,  
    /* Non-scalar types */  
    CosString = 5,  
    CosDict = 6,  
    CosArray = 7,  
    CosStream = 8  
};  
typedef ASInt32 CosType;
```

Description

Constants that specify a Cos object's type (string, number, dictionary, and so on).

Types can be ORed to form composite types, such as "CosDictOrStream," which is 14 (`CosDict | CosStream`).

Header File

CosExpT.h

Related Methods

[CosObjGetType](#)

Members

CosNull	A NULL object, or an invalid object.
CosInteger	An integer object.
CosFixed	A fixed number object.
CosBoolean	An ASBool object.
CosName	A name object.
CosString	A string object.
CosDict	A dictionary object.
CosArray	An array object.
CosStream	A stream object.

CosStreamOpenMode

typedef [AEnum8](#) CosStreamOpenMode;

Description

Constants that specify whether filters and decryption should be applied to the stream's data.

Header File

CosExpT.h

Related Methods

[CosStreamOpenStm](#)

Values

<code>cosOpenRaw</code>	The data will be neither filtered nor decrypted.
<code>cosOpenUnfiltered</code>	The data will be decrypted but not filtered.
<code>cosOpenFiltered</code>	The data will be both decrypted and filtered. (This is the usual case.)



Cstring_Ptr

```
typedef char* Cstring_Ptr;
```

Description

A UNIX or Windows platform-specific path value.

Header File

ASExpT.h

Related Methods

[ASPlatformPathGetCstringPtr](#)

Emit Flags

Description

Enumerated [ASEnum8](#) constants that specify additional print emit options.

Header File

AVExpT.h

Related Types

[AVDocPrintParams](#)

Related Methods

[AVDocPrintPagesWithParams](#)

Values

kAVEmitHalftones	Emit the halftones specified in the document.
kAVEmitSeparableImagesOnly	Raise an error on images that cannot be represented in EPS files, following the separation conventions in Technical Note #5044, <i>Color Separation Conventions for PostScript Language Programs</i> .
kAVSuppressCropClip	Do not emit the cropbox page clip.
kAVSuppressTransfer	Do not emit the transfer functions in the document.
kAVSuppressBG	Do not emit the BlackGeneration in the document.
kAVSuppressUCR	Do not emit the UnderColorRemovals in the document.
kAVSuppressCJKFontSubst	If the field is set, calls to AVDocPrintPagesWithParams generate PostScript that suppresses printer-based font substitution for CJK fonts.
kAVSuppressRotate	(New for 5.0) Do not rotate the page.
kAVSuppressCenter	(New for 5.0) Do not center the page.
kAVSuppressAnnots	(New for 5.0) Do not emit text annotations.
kAVSetPageSize	(New for 5.0) Enable setPageSize, choose paper tray by PDF page size.

kAVSaveVM	(New for 5.0) Attempt to reduce amount of VM used on PostScript printers.
kAVOptimizeForSpeed	(New for 5.0) PostScript only - If set, PostScript is optimized for speed otherwise pages must be independent.
kAVSilent	(New for 5.0) When printing via Windows' Dynamic Data Exchange (DDE), alerts are not generated; false is returned.
kAVEmitBBoxClip	(New for 5.0) Normally, we emit a clip to BoundingBox for EPS, for one client, this breaks their workflow.
kAVEmitTransferFuncs	(New for 5.0) When set, transfer functions in the PDF file will be emitted to the PostScript output stream.
kAVPrintICCColorsAsDevice	(New for 5.0) When set, this flag causes special handling of 1 and 4 component ICC profiles, the same as that done when the "Print ICC colors as Device colors" checkbox is set in the Advanced print dialog. NOTE: do not convert the ICC profile to a color space array when printing PostScript with PostScript color management on.
kAVApplyOverPrint	(New for 5.0) When set, the print path behaves as if the user had clicked the Apply Overprint Preview checkbox in the Advanced print dialog.
kAVApplyWorkingColorSpaces	(New for 5.0) When set, the print path behaves as if the user had clicked the Apply Working Color Spaces checkbox in the Advanced print dialog.

Emit Font Options

Description

Enumerated [ASEnum8](#) constants used to specify options for printing a file to PostScript.

Header File

`AVExpT.h`

Related Types

[AVDocPrintParams](#)

Related Methods

[AVDocPrintPagesWithParams](#)

Values

<code>kAVEmitFontNoFonts</code>	<i>(Obsolete after Acrobat 3.0)</i> Embed no fonts.
<code>kAVEmitFontAllEmbType1</code>	<i>(Obsolete after Acrobat 3.0)</i> Embed all Type 1 embedded fonts.
<code>kAVEmitFontAllType1</code>	<i>(Obsolete after Acrobat 3.0)</i> Embed all Type 1 fonts.
<code>kAVEmitFontEmbeddedFonts</code>	Emit all embedded fonts.
<code>kAVEmitFontAllFonts</code>	Emit all fonts.

ExternalDocServerCreationData

ExternalDocServerCreationDataRec

```
typedef struct _t_ExternalDocServerCreationData {
    ASSize\_t size;
    ExternalDocWindowData platformWindow;
    AVExecuteProc acrobatProc;
    void* acrobatProcData;
    CrossDocLinkProc crossDocLinkProc;
    void* crossDocLinkProcData;
    AVSetMessageProc setMessage;
    void* setMessageProcData;
    /* New for Acrobat 3.01 */
    CrossDocLinkWithDestProc crossDocLinkWithDestProc;
    void* crossDocLinkWithDestData;
    /* New for Acrobat 5.0 */
    AVSetFocusProc setFocus;
    void* setFocusProcData;
} ExternalDocServerCreationDataRec, *ExternalDocServerCreationData;
```

Description

Data for an [AVDoc](#) in an external window. Platform-dependent structure used in [AVDocOpenParams](#) when opening an [AVDoc](#) with [AVDocOpenFromASFileWithParamString](#), [AVDocOpenFromASFileWithParams](#), or [AVDocOpenFromPDDocWithParams](#).

Header File

AVExpT.h

Related Methods

[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromASFileWithParams](#)
[AVDocOpenFromPDDocWithParams](#)

Members

size	Size of the data structure. Must be set to <code>sizeof (ExternalDocServerCreationDataRec)</code>

platformWindow	Platform-dependent structure of type ExternalDocWindowData representing a window. ● Macintosh — ExternalDocWindowData structure ● Windows — HWND cast as ExternalDocWindowData ● UNIX — Widget cast as ExternalDocWindowData
acrobatProc	Optional callback. Called when the “Acrobat button” (if present) is clicked in the external application.
acrobatProcData	Client-specified data for acrobatProc .
crossDocLinkProc	Callback to call when a cross-document link occurs.
crossDocLinkProcData	Client-specified data for crossDocLinkProc .
setMessage	Currently unused.
setMessageProcData	Currently unused. Client-specified data for setMessage .
crossDocLinkWithDestProc	Callback to call when a cross-document link occurs.
crossDocLinkWithDestData	Client-specified data for crossDocLinkWithDestProc .
setFocus	(New in Acrobat 5.0) Callback to call when Acrobat wishes to return focus to the browser displaying the document.
setFocusProcData	(New in Acrobat 5.0) Client-specified data for setFocus .

ExternalDocWindowData

ExternalDocWindowDataRec

```
typedef struct _t_ExternalDocWindowData {
    ExternalDocWindowRefData ref;
    AVSetCursorProc setCursor;
    void* setCursorProcData;
} ExternalDocWindowDataRec, *ExternalDocWindowData;
```

Description

Data for an [AVDoc](#) in an external window. Platform-dependent structure used in [ExternalDocServerCreationData](#) when opening an [AVDoc](#) with [AVDocOpenFromASFileWithParamString](#), [AVDocOpenFromASFileWithParams](#), or [AVDocOpenFromPDDocWithParams](#).

In Mac OS, a plug-in must handle events that effect the window, such as resize and mouse events.

Header File

AVExpT.h

Related Methods

- [AVDocOpenFromASFileWithParamString](#)
- [AVDocOpenFromASFileWithParams](#)
- [AVDocOpenFromPDDocWithParams](#)

Members

ref	Pointer to external window data. Must be of type ExternalDocWindowRefData .
setCursor	Callback for handling mouse-related events, such as mouse down or mouse movement.
setCursorProcData	Optional client-specified data for setCursor .

ExternalDocWindowRefData

ExternalDocWindowRefDataRec

```
typedef struct _t_ExternalDocWindowRefData {
    /* All of these values are handled through dereferences */
    WindowPtr* window;
    ASUns32* x;
    ASUns32* y;
    ASUns32* width;
    ASUns32* height;
    Rect* clip;
    ASInt32* portx;
    ASInt32* porty;
} ExternalDocWindowRefDataRec, *ExternalDocWindowRefData;
```

Description

(*Macintosh only*) Data for an external window. Platform-dependent structure used in [ExternalDocWindowData](#) when opening an [AVDoc](#) with [AVDocOpenFromASFileWithParamString](#), [AVDocOpenFromASFileWithParams](#), or [AVDocOpenFromPDDocWithParams](#).

Coordinates specified in this structure are in application space. Use [AVRectToRect](#) to translate from user space to device space coordinates, then use the Macintosh [GlobalToLocal](#) function to translate from device space coordinates to application space coordinates.

Header File

AVExpT.h

Related Methods

[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromASFileWithParams](#)
[AVDocOpenFromPDDocWithParams](#)

Members

window	WindowPtr for the external window.
x	x-displacement in application space coordinates from the application's window to the AVPageView in which Acrobat renders the PDF file.
y	y-displacement in application space coordinates from the application's window to the AVPageView in which Acrobat renders the PDF file.

width	Width of external window, specified in device space units.
height	Height of external window, specified in device space units.
clip	Clipping rectangle (Macintosh Rect) for external window in application space units. Usually the entire window.
portx	x-displacement in application space coordinates from the AVPageView in which Acrobat renders the PDF file to the actual PDF file page. Should usually be 0.
porty	y-displacement in application space coordinates from the AVPageView in which Acrobat renders the PDF file to the actual PDF file page. Should usually be 0.



Font Flags

Description

Constants that indicate a font's attributes (fixed width, roman or symbolic, sans serif, and so forth).

Header File

`PDExpT.h`

Related Methods

Part of the [PDFontMetricsP](#) structure used by [PDFontGetMetrics](#) and [PDFontSetMetrics](#).

Members

<code>PD_FIXED_WIDTH</code>	All glyphs in the font are the same width.
<code>PD_SERIF</code>	The font is a serif font.
<code>PD_PI</code>	The font is a symbolic (pi) font.
<code>PD_SCRIPT</code>	The font is a script font.
<code>PD_STD_ENCODING</code>	The font uses standard encoding.
<code>PD_ITALIC</code>	The font is an italic font.
<code>PD_ALL_CAP</code>	The font is an all-caps font.
<code>PD_SMALL_CAP</code>	The font is a small caps font.
<code>PD_FORCE_BOLD</code>	Force bold characters to draw bold even at small point sizes.



FSRef_Ptr

```
typedef struct FSRef *FSRef_Ptr;
```

Description

A pointer to the Mac OS platform-specific FSRef.

Header File

ASExpT.h

Related Methods

[ASPlatformPathGetFSRefPtr](#)

FSRefWithCFStringRefRec_Ptr

```
typedef struct _t_FSRefWithCFStringRefRec {  
    struct FSRef *ref;  
    const struct __CFString *str;  
} FSRefWithCFStringRefRec;  
typedef FSRefWithCFStringRefRec *FSRefWithCFStringRefRec_Ptr;
```

Description

A structure containing the equivalent of two Mac OS platform-specific types: a pointer to an **FSRef** and a **CFStringRef**.

Header File

ASExpT.h

Related Methods

[ASPlatformPathGetFSRefWithCFStringRefRecPtr](#)



FSSpec_Ptr

```
typedef struct FSSpec *FSSpec_Ptr;
```

Description

A pointer to a Mac OS platform-specific **FSSpec**.

Header File

`ASExpT.h`

Related Methods

[ASPlatformPathGetFSSpecPtr](#)



gExtensionID

Description

A global variable containing an unique identifier for the client.

Header File

`PICommon.h`

Related Methods

Any method needing to identify the client.

gResFile

Description

(*Macintosh only*) A global variable containing the file reference number for the client's file. This is needed with version 2.1 or later of the Acrobat viewers because the client's resource file is not placed at the top of the resource chain automatically. See the Macintosh chapter of Technical Note #5167, [Acrobat Development Overview](#), for further information.

Header File

PICommon.h

Related Methods

Any method needing to identify the client.

Example

```
short oldResFile;  
DialogPtr myDialog;  
  
oldResFile = CurResFile();  
UseResFile(gResFile);  
myDialog = GetNewDialog(23, NULL, (Ptr) -1);  
useResFile(oldResFile);
```

HFTData

```
typedef struct _t_HFTData
{
    ASUns32 size;
    ASCount numSelectors;
    ASVersion version;
} HFTDataRec;
typedef const HFTDataRec *HFTData;
```

Description

A data structure to pass to an HFT server to create a new **HFT**. New in Acrobat 6.0.

Header File

ASExpT.h

Related Methods

[HFTNewEx](#)

Members

size	The size of this structure.
numSelectors	The number of entries in the new HFT . This determines the number of methods that the HFT can contain; each method occupies one entry.
version	The version number.



HFTEntry

```
typedef void* HFTEntry;
```

Description

A single entry in an **HFT**.

An **HFTEntry** may be cast to a pointer to a function whose prototype must be provided by the **HFT**'s description file.

Header File

CoreExpT.h

Related Methods

[HFTReplaceEntry](#)

[HFTGetReplacedEntry](#)



HFTEntryReplaceable

Description

A flag that specifies whether an **HFT** entry is replaceable.

- If set, the new entry can be replaced. Clients should generally use this value, allowing other clients to subsequently replace the method again.
- If not set, the new entry cannot be replaced.

Header File

`CoreExpT.h`

Related Methods

[HFTReplaceEntry](#)

HFT Values

Global variables for each of the Acrobat viewer's built-in **HFTs**. These are needed when replacing a method or calling a replaced method.

Header File

PIMain.h

Related Methods

[HFTReplaceEntry](#)

[HFTGetReplacedEntry](#)

Values

When replacing a method from the API section ...	HFT must be...
AVModel	gAcroViewHFT
PDModel	gPDModelHFT
AcroSupport	gAcroSupportHFT
Cos	gCosHFT
Macintosh-specific	gMacintoshHFT
UNIX-specific	gUnixHFT
Windows-specific	gWinHFT



HiliteEntry

```
typedef struct _t_HiliteEntry {  
    ASUns16 offset;  
    ASUns16 length;  
} HiliteEntry;
```

Description

Data structure representing a single entry (starting location and length) in a highlight list.

Header File

PDExpT.h

Related Methods

[PDTextSelectCreatePageHilite](#)

[PDTextSelectCreateWordHilite](#)

Modifier Keys

Constants corresponding to various keyboard modifier keys.

Header File

AVExpT.h

Values

Constant Type	Macintosh	Windows
AV_OPTION	Option key	Ctrl key
AV_COMMAND	Command key	—
AV_CONTROL	Control key	Ctrl key
AV_SHIFT	Shift key	Shift key
AV_EXTENDED (New in Acrobat 5.0)	Some keys from the right side of an extended keyboard (such as Enter and numeric keypad)	Some keys from the right side of an extended keyboard (such as Enter)

Related Structure

[AVClickParams](#)

Related Callbacks

[AVAnnotHandlerDoClickProc](#)
[AVAnnotHandlerDoKeyDownExProc](#)
[AVAnnotHandlerDoKeyDownProc](#)
[AVDocSelectionKeyDownProc](#)
[AVPageViewClickProc](#)
[AVPageViewKeyDownProc](#)
[DoClickProcType](#)
[DoKeyDownProcType](#)

Related Methods

[AVMenuItemGetShortcut](#)
[AVMenuItemNew](#)
[AVMenuItemNewWithASText](#)
[AVPageViewDragOutNewRectSnapped](#)
[AVPageViewDragRectSnapped](#)
[AVPageViewDragRectSnappedEx](#)
[AVSysGetModifiers](#)



Page Specification

Constants used wherever a page number or range or count is required.

Header File

PDExpT.h

Related Methods

Various [PDDoc](#) methods.

Values

Flag	Value	Description
PDBeforeFirstPage	-1	Page before the first page.
PDLastPage	-2	Last page.
PDAllPages	-3	All pages of a document.
PDOddPagesOnly	-4	Odd pages of a document.
PDEvenPagesOnly	-5	Even pages of a document.



PageUnitsType

```
typedef enum _t_PageUnitsType {
    Points,
    Inches,
    Millimeters,
    Centimeters,
    Picas, /* largest avpPageUnits value */
/* following used only for kKeyPrefsDisplayMeasure pref */
    Feet,
    Yards,
    Meters,
    Kilometers,
    Miles,
    Custom
} PageUnitsType;
```

Description

Constants used to specify measurement units in [AVPrefsType](#).

Do not set the **avpPageUnits** preference to any value larger than **Picas**.

Related Methods

[AVAppGetPreference](#)

[AVAppSetPreference](#)

Header File

AVExpT.h



PDActionClipboardData

```
typedef struct _t_PDActionClipboardData  
*PDActionClipboardData;
```

Description

Opaque structure used to store [PDAction](#) data for copy and paste operations.

Header File

PDExpT.h

Related Methods

[PDActionCopy](#)

[PDActionDestroyClipboardData](#)

PDActionHandler

PDActionHandlerRec

```
typedef struct _t_PDActionHandler {
    ASSize_t size;
    void* userData;
    PDActionHandlerGetTypeProc GetType;
    PDActionHandlerCanCopyProc CanCopy;
    PDActionHandlerCopyProc Copy;
    PDActionHandlerCanPasteProc CanPaste;
    PDActionHandlerPasteProc Paste;
    PDActionHandlerDestroyDataProc DestroyData;
    PDActionHandlerDestroyProc Destroy;
} PDAnnotHandlerRec, *PDAnnotHandler;
```

Description

Data structure containing callbacks that implement an action manager. The callbacks provide copy-and-paste functionality for a particular type of action.

Header File

PDExpT.h

Related Methods

[PDRegisterActionHandler](#)

Members

size	Size of the data structure. Must be set to <code>sizeof (PDAnnotHandlerRec)</code> .
userData	Pointer to a block of user-supplied data.

PDActionHandlerData

```
typedef struct _t_PDActionHandlerData *PDActionHandlerData;
```

Description

Used to store [PDAction](#) data for copy and paste operations.

Header File

PDExpT.h

Related Callbacks

[PDActionHandlerCanPasteProc](#)
[PDActionHandlerCopyProc](#)
[PDActionHandlerDestroyDataProc](#)
[PDActionHandlerPasteProc](#)

PDAannotArray

PDAannotArrayRec

```
typedef struct _s_PDAannotArray
{
    ASInt32 annotCount;
    PDAannot* annots;
} PDAannotArrayRec, *PDAannotArray;
```

Description

Used by [PDDocExportSomeNotes](#); represents an array of **PDAannots**.

Header File

PDExpT.h

Related Methods

[PDDocExportSomeNotes](#)

Members

annotCount	Annotation count.
annots	Pointer to an array of PDAannots .

PDAnnotClipboardData

```
typedef struct _t_PDAnnotClipboardData *PDAnnotClipboardData;
```

Description

Used to store [PDAnnot](#) data for copy and paste operations.

Header File

PDExpT.h

Related Methods

[PDAnnotCanPaste](#)

[PDAnnotCopy](#)

[PDAnnotDestroyClipboardData](#)

[PDAnnotPaste](#)

PDAnnot Flags

Description

Constants that indicate annotation properties.

NOTE: These flags are not the ones used in the **flags** field of the [AVAnnotHandler](#) structure.

Header File

PDExpT.h

Related Methods

[PDAnnotGetFlags](#)

[PDAnnotSetFlags](#)

Members

pdAnnotInvisible	Controls whether the annotation is invisible—or is drawn with the missing handler icon—if its handler is not available.
pdAnnotHidden	Controls whether the annotation should be drawn—whether its handler is present or not.
pdAnnotPrint	Controls whether the annotation is printed or not. To be printed, an annotation must have an Appearance (AP) key.
pdAnnotNoZoom	If set, the annotation does not zoom with the page view.
pdAnnotNoRotate	If set, the annotation does not rotate with the page.
pdAnnotNoView	If set, the annotation is not viewed but can be printed.
pdAnnotReadOnly	If set, the annotation does not interact with the user.

PDAnnotHandler

PDAnnotHandlerRec

```
typedef struct _t_PDAnnotHandler {
    ASSize_t size;
    void* userData;
    PDAnnotHandlerGetTypeProc GetType;
    PDAnnotHandlerGetAnnotInfoProc GetAnnotInfo;
    PDAnnotHandlerDeleteAnnotInfoProc DeleteAnnotInfo;
    PDDocWillExportAnnotProc WillExportAnnot;
    PDDocWillImportAnnotProc WillImportAnnot;
    PDAnnotWillPrintProc WillPrintAnnot;
    PDAnnotHandlerGetAnnotInfoFlagsProc GetAnnotInfoFlags;
    PDAnnotHandlerCanCopyProc CanCopy;
    PDAnnotHandlerCopyProc Copy;
    PDAnnotHandlerCanPasteProc CanPaste;
    PDAnnotHandlerPasteProc Paste;
    PDAnnotHandlerDestroyDataProc DestroyData;
    PDAnnotHandlerDestroyProc Destroy;
    PDAnnotHandlerGetHeelPointProc GetHeelPoint;
    PDAnnotHandlerGetPrintAppearanceProc GetPrintAppearance;
} PDAnnotHandlerRec, *PDAnnotHandler;
```

Description

Data structure containing callbacks that implement an annotation manager. The callbacks implement the annotation manager functions. For example, view, delete, or export the annotations of a document as a list, sorted by type, author, or date.

To fully use a **PDAnnotHandler**, the **AVAnnotHandler** associated with this annotation must have its **AVAnnotHandlerGetInfoProc** and **AVAnnotHandlerDeleteInfoProc** callbacks defined.

Header File

PDExpT.h

Related Methods

[PDRegisterAnnotHandler](#)

Members

size	Size of the data structure. Must be set to sizeof (PDAnnotHandlerRec) .
userData	Pointer to a block of user-supplied data.

PDAnnotInfo

PDAnnotInfoRec

```
typedef struct _t_PDAnnotInfoRec {
    ASSize\_t size;
    ASFlagBits nOperationFlags;
    unsigned char* cAuthor;
    unsigned char* cContents;
    unsigned char* cDate;
    ASFixed fxLayer; /* New in Acrobat 5.0 */
} PDAnnotInfoRec, *PDAnnotInfo;
```

Description

Information associated with an annotation.

Header File

PDExpT.h

Related Callbacks

[PDAnnotHandlerDeleteAnnotInfoProc](#)

[PDAnnotHandlerGetAnnotInfoProc](#)

Related Methods

[PDRegisterAnnotHandler](#)

Members

size	Size of the data structure. Must be set to sizeof (PDAnnotInfoRec) .
nOperationFlags	Operations allowed on this annotation. Operations permitted are: ● PDAnnotOperationSummarize—OK to summarize annotations ● PDAnnotOperationFilter—OK to filter annotations ● PDAnnotOperationManager—OK to manage annotations ● PDAnnotIgnorePerms—Allow modifying this annotation type in a write-protected document ● PDAnnotOperationAll—All operations are allowed.
cAuthor	Author of this annotation. Must be in either PDFDocEncoding or Unicode (with the 0xFEFF prefix).

cContents	Associated text for this annotation. Must be in either PDFDocEncoding or Unicode (with the 0xFEFF prefix).
cDate	Modification date of this annotation. The date should be the standard date format as specified in Section 3.8.3 in the PDF Reference .
fxLayer	<i>(New in Acrobat 5.0)</i> The layer of this annotation.



PDAnnotPrintOp

```
typedef AEnum16 PDAnnotPrintOp;
```

Description

Constants that identify the type of a print operation being performed.

Header File

PDExpT.h

Related Callbacks

[PDAnnotHandlerGetPrintAppearanceProc](#)

Values

<code>kPDAnnotPrintStandard</code>	A standard print operation.
<code>kPDAnnotPrintVariableData</code>	User selected "Form Fields Only"



PDCharSet

Constants that identify the character set of a Type 1, multiple master Type 1, or TrueType font.

Header File

PDExpT.h

Related Methods

[PDFontGetCharSet](#)

Values

PDStandardRomanCharSet	The font uses Adobe Standard encoding. This is determined by the "Uses Adobe Standard Encoding" bit in the font descriptor.
PDUnknownCharSet	The font does not use Adobe Standard Encoding.
PDAdobeExpertCharSet	Currently unused.

PDChromaticity

```
typedef struct _t_PDChromaticity {  
    ASFixed x;  
    ASFixed y;  
} PDChromaticity;
```

Description

Data structure containing a monitor's chromaticity, for use in displaying device-independent color.

x and **y** are the two values needed to specify the chromaticity.

Header File

PDExpT.h

Related Types

[PDColorCalP](#)

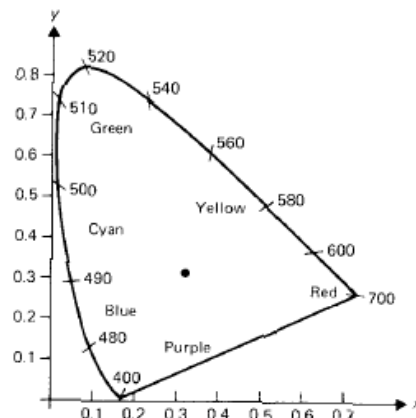
Related Methods

[PDPrefGetColorCal](#)

[PDPrefSetColorCal](#)

Members

x	The x-axis component of the monitor's chromaticity. See figure. Must be ≥ 0 and < 1 . x + y must be ≤ 1 .
y	The y-axis component of the monitor's chromaticity. See figure. Must be > 0 and ≤ 1 . x + y must be ≤ 1 .





PDColorCal

PDColorCalP

```
typedef struct _t_PDColorCal {  
    PDChromaticity whiteChrom;  
    PDChromaticity redChrom;  
    PDChromaticity greenChrom;  
    PDChromaticity blueChrom;  
    ASFixed redGamma;  
    ASFixed greenGamma;  
    ASFixed blueGamma;  
} PDColorCal, *PDColorCalP;
```

Description

Data structure used to represent the characteristics of an output device; needed for device-independent color.

Header File

PDExpT.h

Related Methods

[PDPrefGetColorCal](#)
[PDPrefSetColorCal](#)



PDColorSpace

Constants that specify the color space in which a color value is specified (for example, RGB or grayscale).

Header File

`PDExpT.h`

Related Types

[PDColorValue](#)

Related Methods

[PDImageSelGetDeviceAttr](#)

Members

PDDeviceGray	Grayscale. Requires 1 value entry to specify the color.
PDDeviceRGB	Red–Green–Blue color specification. Requires 3 value entries to specify the color.
PDDeviceCMYK	Cyan–Magenta–Yellow–Black color specification. Requires 4 value entries to specify the color.



PDColorValue

PDColorValueRec

```
typedef struct _t_PDColorValueRec {  
    PDColorSpace space;  
    ASFixed value[4];  
} PDColorValueRec, *PDColorValue;
```

Description

Data structure representing a color. The number of elements needed in the **value** field depends on the color space type (specified in the **space** field). See [PDColorSpace](#) for more information.

See also [AVPrefsType](#).

Header File

PDExpT.h

Related Methods

- [AVAppBeginFullScreen](#)
- [AVPageViewGetColor](#)
- [AVPageViewSetColor](#)
- [PDAnnotGetColor](#)
- [PDAnnotSetColor](#)
- [PDStyleGetColor](#)



PDCount

```
typedef ASInt32 PDCount;
```

Description

A numeric count value for use in [PDImageAttrs](#).

Header File

PDExpT.h

PDCryptBatchHandler

PDCryptBatchHandlerRec

```
typedef struct _t_PDCryptBatchHandler *PDCryptBatchHandler;
typedef struct _t_PDCryptBatchHandler {
    ASSize_t size;
    PDCryptBatchShowDialogProc BatchShowDialog;
    PDCryptBatchParamDescProc BatchParamDesc;
    PDCryptBatchNewAuthDataProc BatchNewAuthData;
    PDCryptBatchAuthorizeProc BatchAuthorize;
    PDCryptBatchFreeAuthDataProc BatchFreeAuthData;
    PDCryptBatchUpdateSecurityDataProc
        BatchUpdateSecurityData;
    PDCryptBatchPreSequenceProc BatchPreSequence;
    PDCryptBatchPostSequenceProc BatchPostSequence;
} PDCryptBatchHandlerRec;
```

Description

Callbacks used to open secured files and to modify security settings while batch is processing a list of files. These callbacks are not called while opening files through the UI. In addition, the regular **PDCryptHandler** functions are not called during batch operations.

Header File

PDExpT.h

Related Methods

[PDRegisterCryptHandler](#)
[PDRegisterCryptHandlerEx](#)

Members

size	Size of the data structure. Must be set to <code>sizeof (PDCryptBatchHandlerRec)</code> .

PDCryptFilterHandler

PDCryptFilterHandlerRec

```
typedef struct _t_PDCryptFilterHandler {
    ASSize_t size;
    PDCryptFilterAuthorizeProc Authorize;
    PDCryptFilterGetDataProc GetData;
    PDCryptFilterStreamProc DecryptStream;
    PDCryptFilterStreamProc EncryptStream;
    PDCryptFilterStringProc DecryptString;
    PDCryptFilterStringProc EncryptString;
} PDCryptFilterHandlerRec;
```

Description

Data structure in [PDCryptHandler](#) containing callbacks that implement a particular security filter. The crypt filter handler is optional. When it is provided, it must perform authorization and supply a key, but need not support either encryption or decryption. It can support only decryption, or both encryption and decryption. If the filter handler does not support encryption, however, the security handler is responsible for not allowing an application to cause Full Save.

The **DecryptStream** and **DecryptString** callbacks are required only if the handler performs decryption; the **EncryptStream** and **EncryptString** callbacks are required only if the handler performs encryption. Leave unused callbacks uninitialized.

The filter handlers are expected to return a 0-length buffer or EOF when unauthorized access is made.

Upon the first call to a procedure of the [PDCryptFilterStreamProc](#) type, the handler is expected to fill out an [ASCryptStmRec](#) structure with pointers to callback routines for various types of stream access.

Header File

```
PDExpT.h
```

Related Methods

```
PDRegisterCryptHandler
PDRegisterCryptHandlerEx
```

Members

size	Size of the data structure. Must be set to <code>sizeof(PDCryptFilterHandlerRec)</code> .
Authorize	Required if the filter handler is supplied. Called to determine whether the user should have access to this filter.



GetData	Required if the filter handler is supplied. Called to retrieve and encryption/decryption key for the filter.
DecryptStream	Optional. Must be set if the handler performs decryption, even if the filter's method is V2 . Called to decrypt a data stream.
EncryptStream	Optional. Called to encrypt a data stream.
DecryptString	Optional. Can be used if the filter's method is V2 and the handler performs decryption. Called to decrypt a string.
EncryptString	Optional. Called to encrypt a string.

PDCryptHandler

PDCryptHandlerRec

```
typedef struct _t_PDCryptHandler {
    ASSize_t size;
    PDCryptAuthorizeProc Authorize;
    PDCryptNewAuthDataProc NewAuthData;
    PDCryptGetAuthDataProc GetAuthData;
    PDCryptNewSecurityDataProc NewSecurityData;
    PDCryptValidateSecurityDataProc ValidateSecurityData;
    PDCryptUpdateSecurityDataProc UpdateSecurityData;
    PDCryptNewCryptDataProc NewCryptData;
    PDCryptFillEncryptDictProc FillEncryptDict;
    PDCryptGetSecurityInfoProc GetSecurityInfo;
/* New in Acrobat 3.0 */
    PDCryptFreeSecurityDataProc FreeSecurityData;
    PDCryptFreeAuthDataProc FreeAuthData;
    PDCryptFreeCryptDataProc FreeCryptData;
/* New callbacks for Acrobat 4.05 and later. */
    PDCryptNewCryptDataExProc NewCryptDataEx;
/* New callbacks for Acrobat 5.0 and later. */
    PDCryptAuthorizeExProc AuthorizeEx;
    PDCryptGetAuthDataExProc GetAuthDataEx;
    PDCryptDisplaySecurityDataProc DisplaySecurityData;
    PDCryptReservedProc GetReservedData;
    PDCryptBatchHandler CryptBatchHandler;
/* New callbacks for Acrobat 6.0 and later. */
    PDCryptFilterHandler CryptFilterHandler;
    PDCryptGetDocPermsProc GetDocPerms;
} PDCryptHandlerRec, *PDCryptHandler;
```

Description

Data structure containing callbacks that implement a security handler. The callbacks implement the security handler functions; for example, get authorization data (such as a password) from the user, set permissions, read and write security-related data in a PDF file, and so on.

Header File

PDExpT.h

Related Methods

[PDRegisterCryptHandler](#)
[PDRegisterCryptHandlerEx](#)



Members

size	Size of the data structure. Must be set to sizeof (PDCryptHandlerRec) .

PDDocCopyParams

PDDocCopyParamsRec

```
typedef struct _t_PDDocCopyParams{
    ASSize\_t size;
    ASPathName newPath;
    ASFileSys fileSys;
    ProgressMonitor progMon;
    void* progMonData;
    CancelProc cancelProc;
    void* cancelProcData;
    ASBool saveChanges;
} PDDocCopyParamsRec, *PDDocCopyParams;
```

Description

Structure used by [PDDocCopyToFile](#) to specify file copy information.

Header File

PDExpT.h

Related Methods

[PDDocCopyToFile](#)

Members

size	Size of the data structure. Must be set to sizeof (PDDocCopyParamsRec) .
newPath	Pathname to copy to, specified in whatever format is correct for fileSys .
fileSys	Pointer to an ASFileSysRec containing the file system that does the copy.
progMon	A progress monitor. May be NULL .
progMonData	Pointer to user-supplied data to pass to progMon each time it is called. Must be NULL if progMon is NULL .
cancelProc	A cancel procedure. May be NULL .
cancelProcData	Pointer to user-specified data to pass to cancelProc each time it is called. Must be NULL if cancelProc is NULL .



saveChanges	Whether changes should be saved if the document is dirty. When true , if the document is dirty and the product is Acrobat, save all in-memory changes to the new copy. Otherwise, just copy the on-disk bytes. Ignored in Adobe Reader.



PDDocFlags

Constants that specify various file status attributes.

Header File

PDExpT.h

Related Methods

[PDDocClearFlags](#)

[PDDocGetFlags](#)

[PDDocSetFlags](#)

Values

Flag	Access	Description
PDDocNeedsSave	get/set	Document has been modified and needs to be saved.
PDDocRequiresFullSave	set only	Document cannot be saved incrementally; when it is saved using PDDocSave , the PDSaveFull flag must be specified (see PDSaveFlags).
PDDocIsModified	get only	Document has been modified slightly (such as bookmarks or text annotations have been opened or closed), but not in a way that warrants saving.
PDDocDeleteOnClose	get/set	Document is based on a temporary file that must be deleted when the document is closed or saved.
PDDocWasRepaired	get only	Document was repaired when it was opened.
PDDocNewMajorVersion	get only	Document's major version is newer than current.
PDDocNewMinorVersion	get only	Document's minor version is newer than current.
PDDocOldVersion	get only	Document's version is older than current.
PDDocSuppressErrors	get/set	do not display errors.
PDDocIsEmbedded	get/set	Document is embedded in a compound document (OLE, OpenDoc).

Flag	Access	Description
PDDocIsLinearized	get only	Document is linearized for page-served remote (network) access.
PDDocIsOptimized	set only	Document is optimized. If this flag is cleared, the Batch Optimizer client and the Adobe PDF Library do not save the file optimized. You can, therefore, linearize a PDF file without optimizing it. Optimizing without linearizing is <i>not</i> allowed, however.

PDDocInsertPagesParams

PDDocInsertPagesParamsRec

```
typedef struct _t_PDDocInsertPagesParams {
    ASSize_t size;
    PDDoc targetDoc;
    PDPageNumber insertAfterThisPage;
    PDDoc srcDoc;
    PDPageNumber srcFromPage;
    PDPageNumber srcToPage;
    PDSmallFlagBits insertFlags;
    ASErrCode error;
} PDDocInsertPagesParamsRec, *PDDocInsertPagesParams;
```

Description

Structure used to pass information to [PDDocWillInsertPagesEx](#) and [PDDocDidInsertPagesEx](#) notifications.

Header File

PDExpT.h

Related Methods

[PDDocInsertPages](#)

Members

size	Size of the data structure. Must be set to sizeof (PDDocInsertPagesParamsRec) .
targetDoc	The document into which pages are inserted. This document must have at least one page.
insertAfterThisPage	The page number in targetDoc after which pages from srcDoc are inserted. The first page is 0. If PDBeforeFirstPage (see PDExpT.h) is used, the pages are inserted before the first page in targetDoc . Use PDLastPage to insert pages after the last page in targetDoc .
srcDoc	The document containing the pages that are inserted into targetDoc .
srcFromPage	The page number of the first page in srcDoc to insert into targetDoc . The first page is 0.
srcToPage	The page number of the last page in srcDoc to insert into targetDoc .

insertFlags	Flags that determine what additional information is copied from srcDoc into targetDoc. Must be an OR of the following (see PDExpT.h): PDInsertAll — Inserts the entire document srcDoc into the document targetDoc. This operation not only inserts the pages themselves, but also merges other document data from srcDoc into targetDoc. In particular, the following happens: 1. The bookmark tree of srcDoc is merged into the bookmark tree of targetDoc by copying it as a new first-level sub-tree of targetDoc's bookmark tree root, of which it becomes the last child. If targetDoc has no bookmark tree, it acquires one identical to the bookmark tree from srcDoc. 2. Named destinations from srcDoc (of PDF 1.1 and later) are copied into targetDoc. If there are named destinations in srcDoc with the same name as some named destination in targetDoc, the ones in targetDoc retain their names and the copied named destinations are given new names based on the old ones with distinguishing digits added. Actions and bookmarks referring to the old names are made to refer to the new names after being copied into targetDoc. 3. Document logical structure from srcDoc is copied into targetDoc. The top-level children of the structure tree root of srcDoc are copied as new top-level children of the structure tree root of targetDoc; a structure tree root is created in targetDoc if there was none before. The role maps of the two structure trees are merged, with name conflicts resolved in favor of the role mappings present in targetDoc. Attribute objects are not copied, nor is class map information from srcDoc merged into that for targetDoc. If PDInsertAll flag is <i>not</i> set, pages copied from srcDoc into targetDoc will have their structure back-pointer information stripped off. PDInsertBookmarks — Inserts bookmarks as well as pages. PDInsertThreads — Inserts threads as well as pages.
error	Error code, which is only valid for the PDDocDidInsertPagesEx notification.

PDDocOCChangeType

```
typedef ASUns8 PDDocOCChangeType;
```

Description

Constants that specify types of changes to the optional-content structures of a **PDDoc**. These types of changes may affect visibility in all **PDOCContexts**. This enumeration is used in the **PDDoc** optional-content notifications.

Header File

PDExpT.h

Related Notifications

[PDDocOCDidChange](#)
[PDDocOCWillChange](#)

Members

kPDOCGCreate	Optional-content group (OCG) or groups created.
kPDOCGProperties	OCG properties changed.
kPDOCGReplace	OCG replaced by another.
kPDOCGDestroy	OCG destroyed.
kPDOCMDAttach	Content made optional.
kPDOCMDRemove	Content made non-optional.
kPDOCConfigCreate	Optional-content configuration created.
kPDOCConfigChange	Configuration changed.
kPDOCConfigDestroy	Configuration destroyed.

PDDocOpenParams

PDDocOpenParamsRec

```
typedef struct _t_PDDocOpenParams{
    ASSize_t size;
    ASFile file;
    ASPathName fileName;
    ASFileSys fileSys;
    PDAuthProcEx authProcEx;
    void* authProcClientData;
    ASBool doRepair;
    PDPerms restrictPerms;
} PDDocOpenParamsRec, *PDDocOpenParams;
```

Description

Structure used by [PDDocOpenWithParams](#) to specify file open information. The parameters are very similar to those in [PDDocOpenEx](#) and [PDDocOpenFromASFileEx](#).

Header File

PDExpT.h

Related Methods

[PDDocOpenWithParams](#)

Members

size	Size of the data structure. Must be set to sizeof (<i>PDDocOpenParamsRec</i>).
file	Pathname to the file, specified in whatever format is correct for fileSys .
fileName	Pathname to the file, specified in whatever format is correct for fileSys .
fileSys	Pointer to an ASFileSysRec containing the file system in which the file resides.

authProcEx	Authorization callback, called only if the file is encrypted. This callback should obtain whatever information is needed to determine whether the user is authorized to open the file, then call PDDocAuthorize (which returns the permissions that the authentication data enables). If the file is encrypted and authProcEx is NULL or returns false , pdErrPassword is raised. The Acrobat viewer's built-in authorization procedure requires the user to enter a password, and allows the user to try three times before giving up.
authProcClientData	Pointer to user-specified data to pass to authProcEx each time it is called.
doRepair	If true , attempt to repair the file if it is damaged; if false , do not attempt to repair the file if it is damaged.
restrictPerms	The permissions to remove from the document.

PDDocPreSaveInfo

```
typedef struct _t_PDDocPreSaveInfo {
    ASSize_t size;
    CosObjSetCallbackFlagProc callbackProc;
} PDDocPreSaveInfoRec, *PDDocPreSaveInfo;
```

Description

Structure used to flag Cos objects you wish to access while a **PDDoc** is saved.

Header File

PDExpT.h

Related Callbacks

[CosObjSetCallbackFlagProc](#)

Related Methods

[PDDocOpenWithParams](#)

Members

size	Size of the data structure. Must be set to sizeof(PDDocPreSaveInfo) .
callbackProc	CosObjSetCallbackFlagProc callback to set flags in Cos objects to provide access to them during the save.

PDDocReadAhead Flags

Flags describing what type of data to read “ahead.”

Header File

PDExpT.h

Related Methods

[PDDocReadAhead](#)

Values

kPDDocReadAheadAcroForms	Allows the AcroForm client to request that all the AcroForm data be read “ahead,” before the viewer needs it. This flag is ignored if the PDF file does not contain a Forms hint table. See Section F.3.5 in the PDF Reference for information about the Forms hint table.
kPDDocReadAheadTemplates	Requests that the PDF file’s Forms Template data be read “ahead,” before the viewer needs it. There is currently no Template hint table defined, so this flag simply causes the rest of the file to be read.
kPDDocReadAheadPageLabels	Requests that the PDF file’s page label data be read “ahead,” before the viewer needs it. There is currently no page label hint table defined, so this flag simply causes the rest of the file to be read.
kPDDocReadAheadStructure	Requests that the PDF file’s logical structure data be read “ahead,” before the viewer needs it. There is currently no logical structure hint table defined, so this flag simply causes the rest of the file to be read.

PDDocRequestReason

```
enum {
    kPDDocRequestUnderway = 0,
    kPDDocRequestComplete,
    kPDDocRequestCancelled,
    kPDDocRequestError
};
typedef ASEnum8 PDDocRequestReason;
```

Description

A constant the represents the request status to a callback that processes a set of pages or an entire document file.

Header File

```
PDExpT.h
```

Related Callbacks

```
PDDocRequestEntireFileProc
PDDocRequestPagesProc
```

Related Methods

```
PDDocRequestEntireFile
PDDocRequestPages
```

Values

<code>kPDDocRequestUnderway</code>	The request is still being processed.
<code>kPDDocRequestComplete</code>	The requested data has arrived.
<code>kPDDocRequestCancelled</code>	The request is cancelled because the file is being closed.
<code>kPDDocRequestError</code>	An error occurred.

PDDocSaveParams

PDDocSaveParamsRec

```
typedef struct _t_PDDocSaveParams {
    ASSize_t size;
    PDSaveFlags saveFlags;
    ASPathName newPath;
    ASFileSys fileSys;
    ProgressMonitor mon;
    void* monClientData;
    CancelProc cancelProc;
    void* cancelProcClientData;
    /* Added in Acrobat 4.0 */
    PDDocPreSaveProc preSaveProc;
    void* preSaveProcClientData;
    CosObjOffsetProc offsetProc;
    void* offsetProcClientData;
    ASInt16 major;
    ASInt16 minor;
    /* Added in Acrobat 6.0 */
    ASBool XMPPacketReadOnly;
    ASInt32 XMPPacketPaddingBytes;
    PDDocPreWriteProc preWriteProc;
    void* preWriteProcClientData;
    PDSaveFlags2 saveFlags2;
    ASUns32 numSubFilesToCompact;
} PDDocSaveParamsRec, *PDDocSaveParams;
```

Description

Parameters used when saving a file with [PDDocSaveWithParams](#) and returned by the [PDDocWillSaveEx](#) notification. Most of these parameters are the same as the parameters for [PDDocSave](#).

Header File

PDExpT.h

Related Methods

[PDDocSaveWithParams](#)

Members

size	Size of the data structure. Must be set to sizeof (PDDocSaveParamsRec)
saveFlags	Bit flags controlling the save operation.

newPath	The path to which the file is saved. A path must be specified when either PDSaveFull or PDSaveCopy are used for saveFlags. If PDSaveIncremental is specified in saveFlags, then newPath should be NULL. If PDSaveFull is specified and newPath is the same as the file's original path, the new file is saved to a file system determined temporary path, then the old file is deleted and the new file is renamed to newPath.
fileSys	File system. If NULL, uses the fileSys of the document's current backing file. Implementation Restriction: Files can only be saved to the same file system, thus fileSys must be NULL or an error is raised.
mon	Progress monitor. Use AVAppGetDocProgressMonitor to obtain the default. May be NULL.
monClientData	Pointer to user-supplied data to pass to mon each time it is called. Must be NULL if mon is NULL.
cancelProc	A cancel procedure. Use AVAppGetCancelProc to obtain the current cancel procedure. May be NULL.
cancelProcClientData	Pointer to user-specified data to pass to cancelProc each time it is called. Must be NULL if cancelProc is NULL.
preSaveProc	Callback to flag Cos objects you wish to access while the PDDoc is saved.
preSaveProcClientData	Pointer to user-specified data to pass to preSaveProc each time it is called.
offsetProc	Callback to get information about interesting Cos objects while the PDDoc is saved.
offsetProcClientData	Pointer to user-specified data to pass to offsetProc each time it is called.
major	Major PDF version number of the document. If major equals 0, both major and minor are ignored. A version number greater than the version associated with the application (e.g. PDF 1.5 for Acrobat 6.0) or less than 1.2 is also ignored

minor	Minor PDF version number of the document.
XMPPacketReadOnly	When true , makes the XML Packet containing the XMP metadata for the document read-only and suppresses the generation of padding. When false , makes the XML Packet writable and includes the number of padding bytes specified by XMPPacketPaddingBytes .
XMPPacketPaddingBytes	The number of padding bytes to include in the XML Packet containing XMP metadata for the document, if XMPPacketReadOnly is false . Non-positive values specify the default padding of 2048 bytes. Ignored when XMPPacketReadOnly is true .
preWriteProc	A callback procedure to call immediately before the document is written to disk.
preWriteProcClientData	Data to pass to the preWriteProc procedure.
saveFlags2	Additional bit flags for the save operation.
numSubFilesToCompact	Currently unused. Set to 0.

PDDrawParams

```
typedef struct _t_PDDrawParams
{
    ASUns32 size;
    void *window;
    void *displayContext;
    ASFixedMatrix matrix;
    ASUns32 flags
    ASFixedRect updateRect;
    CancelProc cancelProc;
    void *cancelProcClientData;
    PDOCContext clientOCContext;
} PDDrawParamsRec, *PDDrawParams;
```

Description

Parameters used for optional-content drawing control. The parameters are the same as those passed to the original version of the method, with the addition of an optional-content context that determines what contents are visible.

Header File

PDExpT.h

Related Methods

[PDDrawCosObjWithParams](#)

[PDPageEnumContents](#)

Members

size	Size of the data structure.
window	Pointer to a platform-dependent window object (WindowPtr or CWindowPtr in Mac OS, or HWND in Windows). In Mac OS, to draw into an offscreen GWorld , pass NULL in window and pass the GWorldPtr in displayContext . In Windows, to draw into an offscreen DC, pass NULL for window .
displayContext	A platform-dependent display context structure (GWorldPtr in Mac OS, HDC in Windows). In Mac OS, displayContext is ignored if window is non- NULL . NOTE: Note: displayContext cannot be reliably used as the hDC for a printer device.
matrix	Pointer to the matrix to concatenate onto the default page matrix. It is useful for converting from page to window coordinates and for scaling.

flags	See above. The only value you should use is 0.
updateRect	A rectangle represented by the coordinates of its four sides.
cancelProc	Procedure called periodically to check for user cancel of the drawing operation. The default cancel procedure can be obtained using AVAppGetCancelProc . May be NULL in which case no cancel procedure is used.
cancelProcClientData	Pointer to user-supplied data to pass to cancelProc each time it is called. Should be NULL if cancelProc is NULL .
clientOCContext	An optional-content context that determines what contents are visible. NULL uses the document's optional-content context, as returned by PDDocGetOCContext (pddoc), which is equivalent to calling the version of the method without optional-content parameters. This context is copied and the copy is used in drawing. This allows a client to change its copy of the context without raising an exception.



PDEBlackPointFlt

```
typedef PDEXYZColorFlt PDEBlackPointFlt;
```

Description

Structure describing a black point in a calibrated color space.

Header File

`PEExpT.h`

Related Types

[PDEGrayCalFlt](#)

[PDELabCalFlt](#)

[PDERGBCalFlt](#)

[PDEWhitePointFlt](#)

Related Methods

[PDEColorSpaceCreate](#)



PDEColorRangeFlt

```
typedef struct {  
    float min;  
    float max;  
} PDEColorRangeFlt;
```

Description

Structure describing a color range in a calibrated Lab color space.

Header File

PEExpT.h

Related Types

[PDELabCalFlt](#)

Related Methods

[PDEColorSpaceCreate](#)

PDEColorSpaceStruct

```
typedef union {  
    PDEGrayCalFlt* calGray;  
    PDERGBCalFlt* calRGB;  
    PDELabCalFlt* lab;  
    PDEICCBasedColorData* icc;  
    PDEIndexedColorData* indexed;  
    PDEPatternColorSpace patternbase;  
    PDESeparationColorData* sep;  
    PDEDeviceNColorData* devn;  
} PDEColorSpaceStruct;
```

Description

A color space structure for [PDEColorSpaceCreate](#). See Section 4.5 in the [PDF Reference](#) for information on color spaces.

Header File

PEExpT.h

Related Methods

[PDEColorSpaceCreate](#)



PDEColorSpec

PDEColorSpecP

```
typedef struct _t_PDEColorSpec {
    PDEColorSpace space;
    PDEColorValue value;
} PDEColorSpec, *PDEColorSpecP;
```

Description

Structure describing a color specification.

Header File

PEExpT.h

Related Methods

Numerous

Members

space	The specified PDEColorSpace.
value	The color value.



PDEColorValue

PDEColorValueP

```
typedef struct _t_PDEColorValue {
    ASFixed color[7];
    PDEObject colorObj2;
    PDEObject colorObj;
} PDEColorValue, *PDEColorValueP;
```

Description

Structure describing a color value.

Header File

PEExpT.h

Related Methods

Numerous
Members

color	Color value components. For instance, a Gray color space has 1 component, an RGB color space has 3 components, a CMYK has 4 components, and so on.
colorObj2	For a DeviceN color space.
colorObj	For color spaces whose color values do not have numeric values, such as the Pattern and Separation color spaces.

PDEContentAttrs

PDEContentAttrsP

```
typedef struct _t_PDEContentAttrs {
    ASUns32 flags;
    ASFixed cacheDevice[8];
    ASInt32 formType;
    ASFixedRect bbox;
    ASFixedMatrix matrix;
    CosObj XUID;
} PDEContentAttrs, *PDEContentAttrsP;
```

Description

Structure describing attributes of a [PDEContent](#) object.

Header File

PEExpT.h

Related Methods

[PDEContentGetAttrs](#)
[PDEContentToCosObj](#)

Members

flags	PDEContentFlags flags.
cacheDevice	If flags has kPDESetCacheDevice set, the first 6 cache device values contain the operands for the d1 (setcachdevice) page operator. If flags has kPDESetCacheDevice set, cacheDevice contains 2 charwidth values.
formType	Only used if PDEContent contains a Form XObject . Corresponds to FormType key in the XObject Form attributes dictionary.
bbox	Only used if PDEContent contains a Form. Bounding box of the PDEContent object. Corresponds to BBox key in the XObject Form attributes dictionary.
matrix	Only used if PDEContent contains a Form. Transformation matrix for the PDEContent object. Corresponds to Matrix key in the XObject Form attributes dictionary.
XUID	Only used if PDEContent contains a Form. The form's XUID, an ID that uniquely identifies the form. Corresponds to XUID key in the XObject Form attributes dictionary.

PDEContentFlags

```
typedef enum {
    kPDESetCacheDevice = 0x0001,
    kPDESetCharWidth = 0x0002,
    kPDEFormMatrix = 0x0004
} PDEContentFlags;
```

Description

Bit field for [PDEContentAttrs](#).

Header File

PEExpT.h

Related Methods

[PDEContentGetAttrs](#)
[PDEContentToCosObj](#)

Members

kPDESetCacheDevice	Indicates cacheDevice contains 6 cache device values.
kPDESetCharWidth	Indicates cacheDevice contains 2 charwidth values.
kPDEFormMatrix	Indicates formMatrix contains a valid matrix.

PDEContentGetResourceFlags

Bit field for [PDEContentAttrs](#).

Header File

PEExpT.h

Related Methods

[PDEContentGetResources](#)

Members

kPDEGetFonts	Obtain font resources.
kPDEGetXObjects	Obtain Xobject resources.
kPDEGetColorSpaces	Obtain color space resources.

PDEContentToCosObjFlags

```
typedef enum {
    kPDEContentToPage = 0x0001,
    kPDEContentToForm = 0x0002,
    kPDEContentToCharProc = 0x0004,
    kPDEContentRev1Compat = 0x0008,
    kPDEContentDoNotResolveForms = 0x0010,
    kPDEContentDoNotResolveType3 = 0x0020,
    kPDEContentEmitDefaultRGBAndGray = 0x0040
} PDEContentToCosObjFlags;
```

Description

Bit field for the [PDEContentToCosObj](#) method, indicating the type of object to create and how it is created.

Header File

PEExpT.h

Related Methods

[PDEContentToCosObj](#)

Members

kPDEContentToPage	Create page contents.
kPDEContentToForm	Create a form.
kPDEContentToCharProc	Create charprocs.
kPDEContentRev1Compat	Currently unused.
kPDEContentDoNotResolveForms	Currently unused.
kPDEContentDoNotResolveType3	Currently unused.
kPDEContentEmitDefaultRGBAndGray	Emit calibrated RGB and gray information using the PDF 1.0 compatible mechanism. In this case, generate rg and k page operators and place DefaultGray and DefaultRGB color space arrays in the Resources dictionary, as described in Section 4.5.7 in the PDF Reference .

PDEDash

PDEDashP

```
typedef struct _t_PDEDash {
    ASFixed dashPhase;
    ASInt32 dashLen;
    ASFixed dashes[11];
} PDEDash, *PDEDashP;
```

Description

Structure describing a dash specification, as described in Table 4.8 in the [PDF Reference](#). See Section 4.3.2 for more information on line dash patterns.

Header File

PEExpT.h

Related Methods

Numerous

Members

dashPhase	Dash phase. Phase is a number that specifies a distance in user space into the dash pattern at which to begin marking the path.
dashLen	Number of entries in the dash array, an element of the Border array.
dashes	Dash array, which specifies distances in user space for the length of dashes and gaps.

PDEDeviceNColorData

```
typedef struct _t_PDEDeviceNColorData {
    ASSize_t size;
    ASAtom* names;
    ASUns32 nNames;
    PDEColorSpace alt;
    CosObj tintTransform;
} PDEDeviceNColorData;
```

Description

Structure describing a **DeviceRGB** or **DeviceCMYK** color space.

Header File

PEExpT.h

Related Types

[PDEColorSpaceStruct](#)

Related Methods

[PDEColorSpaceCreate](#)

Members

size	Size of the data structure. Must be set to <code>sizeof(PDEDeviceNColorData)</code> .
names	Names of colorants.
nNames	Number of colorants.
alt	Alternative color space.
tintTransform	The tintTransform dictionary or function. See Section 4.5.5 in the PDF Reference for more information.

PDEElementCopyFlags

```
typedef enum {  
    kPDEElementCopyForClip = 0x0001,  
    kPDEElementCopyClipping = 0x0002  
} PDEElementCopyFlags;
```

Description

Bit field for [PDEElementCopy](#).

Header File

PEExpT.h

Related Methods

[PDEElementCopy](#)

Members

kPDEElementCopyForClip	Copied element does not need gstate or clip.
kPDEElementCopyClipping	Acquire the clip path and put it in the copied object.

PDEEnumElementsFlags

```
typedef enum {  
    kPDEContentIgnoreMarkedContent = 0x0001  
} PDEEnumElementsFlags;
```

Description

Bit field for [PDEEnumElements](#) method.

Header File

PEExpT.h

Related Methods

[PDEEnumElements](#)

Members

kPDEContentIgnoreMarkedContent	Indicates whether Marked Content is ignored in the enumeration. This may be useful when generating elements purely for display purposes.

PDEFilterArray

PDEFilterArrayP

```
typedef struct _t_PDEFilterArray {
    ASInt32 numFilters;
    PDEFilterSpec spec[1];
} PDEFilterArray, *PDEFilterArrayP;
```

Description

Filter information for streams.

Although the **PDEFilterSpec** is declared as a one-member array in the header file, more members can be added by dynamically allocating the **PDEFilterArray** with space for as many filters as you would like to add.

In practice, there is seldom need for more than two filters.

Header File

PEExpT.h

Related Methods

- [PDEContentToCosObj](#)
- [PDEImageCreate](#)

Example

```
pdeFilterArray = (PDEFilterArrayP) malloc(offsetof(PDEFilterArray, spec)
+ (sizeof(PDEFilterSpec) *2));
```

Members

numFilters	Number of filters in the array.
spec	Variable length array of filter spec.

PDEFilterSpec

PDEFilterSpecP

```
typedef struct _t_PDEFilterSpec {
    CosObj decodeParms;
    CosObj encodeParms;
    ASAtom name;
    ASInt16 padding;
} PDEFilterSpec, *PDEFilterSpecP;
```

Description

Filter element in a filter array.

Members

decodeParms	Parameters used by the decoding filters specified with the Filter key. Corresponds to the DecodeParms key in the stream dictionary. Must be set to NULL if PDEFilterSpec is specified but no encode or decode parameters are specified. This can be done by passing CosNewNull for the unused encode and/or decode params. Required decode params for DCTDecode are Columns, Rows, and Colors. The parameters should be passed in a CosDict.
encodeParms	Parameters used when encoding the stream. Required for DCTDecode and JBIG2Decode filters; optional for other filters. • The DCTDecode encoder requires that the Colors, Rows, and Columns parameters are passed in the encodeParms dictionary; see the PDF Reference. • The JBIG2Decode encoder requires that the Width and Height parameters are passed. These must be integers that specify the image width and height in pixels. The following parameters are optional for this filter: –JB2Quality: An integer between 4 and 10, with 4 the most lossy and 10 the least lossy. Default is 10. –InvertImage: true if 1=black and 0=white in the image data. Default is false. –CompleteJB2File: When true, use the output encoded stream as a standalone JBIG2 file. When false (the default) output is embedded in a PDF document. Must be set to NULL if PDEFilterSpec is specified but no encode or decode parameters are specified. This can be done by passing CosNewNull for the unused encode and/or decode params.

name	Filter name. Supported filters are: ASCIIHexDecode ASCII85Decode LZWDecode DCTDecode CCITTFaxDecode RunLengthDecode FlateDecode JBIG2Decode
padding	Reserved — used to align to 32 bits.

Header File

PEExpT.h

Related Methods

[CosNewStream](#)

[PDEContentToCosObj](#)

[PDEImageCreate](#)

Example

```
CosObj dctDict = CosNewDict(theCosDoc,false, 3);
CosDictPut(dctDict, ASAtomFromString("Columns"),
CosNewInteger(NULL,false,width_of_image));

CosDictPut(dctDict, ASAtomFromString("Rows"), CosNewInteger(NULL, false,
hight_of_image));

CosDictPut(dctDict, ASAtomFromString("Colors"), CosNewInteger(NULL,
false,num_colors));
/* 3 for RGB 4 for CMYK */

pdeFilters->spec[i].encodeParms = dctDict;
```




PDEFontAttrs

PDEFontAttrsP

```
typedef struct _t_PDEFontAttrs {
    ASAtom name;
    ASAtom type;
    ASAtom charSet;
    ASAtom encoding;
    ASUns32 flags;
    ASFixedRect fontBBox;
    ASInt16 missingWidth;
    ASInt16 stemV;
    ASInt16 stemH;
    ASInt16 capHeight;
    ASInt16 xHeight;
    ASInt16 ascent;
    ASInt16 descent;
    ASInt16 leading;
    ASInt16 maxWidth;
    ASInt16 avgWidth;
    ASInt16 italicAngle;
    /* New in Acrobat 4.0 */
    ASAtom cidFontType;
    ASInt16 wMode;
    ASAtom psName;
    ASAtom lang;
    ASAtom registry;
    ASAtom ordering;
    ASInt32 supplement;
    /* New in Acrobat 5.0 */
    ASInt32 cantEmbed;
    ASAtom deltaEncoding;
    /* New in Acrobat 5.0.5 */
    ASUns32 protection;
} PDEFontAttrs, *PDEFontAttrsP;
```

Description

Attributes of a [PDEFont](#) and of a [PDSysFont](#). This structure is also referenced in [PDEFontCreateParams](#).



Header File

PEExpT.h

Related Methods

[PDEFontCreate](#)
[PDEFontCreateWithParams](#)
[PDEFontGetAttrs](#)
[PDFindSysFont](#)
[PDFindSysFontEx](#)
[PDSysFontGetAttrs](#)
[PDSysFontGetEncoding](#)
[PDSysFontAcquirePlatformData](#)

Members

name	An ASAtom for font name, as in "CourierStd". Corresponds to the BaseFont key in the font dictionary of a PDF file (see Section 5.6.3 in the PDF Reference).
type	An ASAtom for font type, corresponding to the Subtype key in a font dictionary. May be "Type1," "TrueType," "MMType1," or "Type0."
charSet	An ASAtom for "Roman" or ASAtomNull . If "Roman," the characters must be a subset of the Adobe Standard Roman Character Set.
encoding	An ASAtom for font encoding. May be MacRomanEncoding , WinAnsiEncoding , or ASAtomNull . In the case of ASAtomNull , call PDSysFontGetEncoding to get more information about the encoding.
flags	Desired font flags, one or more of Font Flags .
fontBBox	Font bounding box in 1000 EM units.
missingWidth	Width of missing character (.notdef).
stemV	Vertical stem width.
stemH	Horizontal stem width.
capHeight	Capital height.
xHeight	X height.
ascent	Maximum ascender height.
descent	Maximum descender depth.
leading	Additional leading between lines.

maxWidth	Maximum character width.
avgWidth	Average character width.
italicAngle	Italic angle in degrees, if any.
cidFontType	ASAtom representing the CID font type: CIDFontType0 or CIDFontType2.
wMode	Writing mode. Must be one of: ● 0 for horizontal writing ● 1 for vertical writing
psName	ASAtom representing the PostScript name of a TrueType font.
lang	ASAtom representing the ISO 639 language code. These are available from http://www.iso.ch .
registry	ASAtom representing the CIDFont's Registry information, as in "Adobe-Japan".
ordering	ASAtom representing the CIDFont's Ordering information, for example, "1".
supplement	The SystemSupplement field from the CIDFont.
cantEmbed	A non-zero value means the font cannot be embedded.
deltaEncoding	The name of the base encoding; that is, the BaseEncoding entry in an encoding dictionary (see section 5.5.5 of the PDF Reference). The Differences entry of the encoding dictionary describes differences (deltas) from the base encoding.
protection	Allows setting one of the following bits to disable font embedding: ● kPDFontNoEmbedding = 1: font should not be embedded. ● kPDFontNoEditableEmbedding = 2: font should not be embedded for editing purposes.

PDEFontCreateFlags

```
typedef enum {
    kPDEFontCreateEmbedded = 0x0001,
    kPDEFontWillSubset = 0x0002,
    kPDEFontDoNotEmbed = 0x0004,
    kPDEFontEncodeByGID = 0x0008
/* New for 5.0*/
    kPDEFontDeferWidths = 0x0010,
    kPDEFontCreateSubset = kPDEFontWillSubset,
    kPDEFontCreateGIDOverride = 0x0020,
    kPDEFontCreateToUnicode = 0x0040
} PDEFontCreateFlags;
```

Description

Flags for [PDEFontCreateFromSysFont](#).

Header File

PEExpT.h

Related Methods

[PDEFontCreateFromSysFont](#)
[PDEFontCreateFromSysFontAndEncoding](#)
[PDEFontCreateFromSysFontWithParams](#)

kPDEFontWillSubset

Subset the font. If you want to subset a font, set both the **kPDEFontCreateEmbedded** and **kPDEFontWillSubset** flags.

Members

kPDEFontCreateEmbedded	Create an embedded font. By itself, this will <i>not</i> subset the font.
kPDEFontWillSubset	Subset the font. If you want to subset a font, set both the kPDEFontCreateEmbedded and kPDEFontWillSubset flags. You must call PDEFontSubsetNow to actually subset the font. Both embedding and sub-setting a font creates a CFF font.
kPDEFontDoNotEmbed	Do not embed the font. You cannot set both this and the kPDEFontWillSubset flags. Nor can you set kPDEFontCreateEmbedded .
kPDEFontEncodeByGID	Create a CIDFont with identity (GID) encoding.

kPDFontDeferWidths	(New for Acrobat 5.0) Wait to get widths until later (affects Type0 fonts only).
kPDFontCreateSubset	(New for Acrobat 5.0) Create a subset.
kPDFontCreateGIDOverride	(New for Acrobat 5.0) PDFLib will convert cp to gid with identity embedded.
kPDFontCreateToUnicode	(New for Acrobat 5.0) Create ToUnicode CMap.

PDFontCreateNeedFlags

```
typedef enum {
    kPDFontCreateNeedWidths = 0x00010000,
    kPDFontCreateNeedToUnicode = 0x00020000,
    kPDFontCreateNeedEmbed = 0x00040000
} PDFontCreateNeedFlags;
```

Description

Bit flags for [PDFontGetCreateNeedFlags](#).

Header File

PEExpT.h

Members

kPDFontCreateNeedWidths	Need to create widths.
kPDFontCreateNeedToUnicode	Need to create ToUnicode stream.
kPDFontCreateNeedEmbed	Need to embed it.

PDEFontCreateFromSysFontParams

PDEFontCreateFromSysFontParamsRec

```
typedef struct _t_PDEFontCreateFromSysFontParams {
    ASUns32 structSize;
    ASUns32 flags;
    ASAtom snapshotName;
    ASFixed *mmDesignVec;
    long ctCodePage;
    ASAtom encoding;
} PDEFontCreateFromSysFontParamsRec,
*PDEFontCreateFromSysFontParams;
```

Description

Data structure used with PDEFont creation.

Header File

PEExpT.h

Related Methods

[PDEFontCreateFromSysFontWithParams](#)

Members

structSize	Size of the data structure. Must be set to sizeof(PDEFontCreateFromSysFontParamsRec) .
flags	A bit mask of the PDEFontCreateFlags .
snapshotName	The name of a multiple master snapshot. See the PDF Reference for more information on snapshots.
mmDesignVec	Pointer to multiple master font design vector.
ctCodePage	Used to select a specific code page supported by the font. When a non-zero code page is supplied, embedding must be turned on and the kPDEFontEncodeByGID flag set. One of the Code Page Values .
encoding	Used to specify which encoding to use with a CID font. Pass ASAtomNull to use the platform default.

PDEFontCreateParams

PDEFontCreateParamsP

```
typedef struct _t_PDEFontCreateParams {
    PDEFontAttrsP attrsP;
    ASUns32 attrsSize;
    ASInt32 firstChar;
    ASInt32 lastChar;
    ASInt16* widthsP;
    char** encoding;
    ASAtom encodingBaseName;
    ASStm fontStm;
    ASInt32 len1;
    ASInt32 len2;
    ASInt32 len3;
    ASBool hasDW;
    ASInt32 dw;
    CosObj w;
    ASBool hasDW2;
    ASInt32 dw2[2];
    CosObj w2;
    ASInt32 toUnicodeLen;
    ASStm toUnicodeStm;
    ASStm cidToGidMapStm;
    char* panoseNo;
    CosObj fd;
    ASStm cidSetStm;
    ASUns32 flags;
    /* New in Acrobat 4.0 */
    ASFixed* mmDesignVec;
} PDEFontCreateParamsRec, *PDEFontCreateParamsP;
```

Description

Parameters used for [PDEFontCreateWithParams](#) to describe a font.

Header File

PEExpT.h

Related Types

[PDEColorSpaceStruct](#)

Related Methods

[PDEFontCreateWithParams](#)

Members

attrsP	Pointer to a PDEFontAttrs for the font attributes.
attrsSize	Size of the data structure. Must be set to sizeof (PDEFontAttrs) .
firstChar	First character index for the widths array, widthsP .
lastChar	Last character index for the widths array, widthsP .
widthsP	Pointer to widths array.
encoding	An array of 256 pointers to glyph names specifying the custom encoding. If any pointer is NULL , no encoding information is written for that entry.
encodingBaseName	An ASAtom representing the encoding base name if the encoding is a custom encoding. If encoding is NULL , encodingBaseName is used as the value of the encoding and must be one of WinAnsiEncoding , MacRomanEncoding , or MacExpertEncoding . If no encoding value is desired, use ASAtomNull . However, for Type 0 fonts, this field must be a valid CMap name, or PDEFontCreateWithParams will fail.
fontStm	Stream with font information.
len1	Length in bytes of the ASCII portion of the Type 1 font file after it has been decoded. For other font formats, such as TrueType or CFF, only len1 is used, and it is the size of the font.
len2	Length in bytes of the encrypted portion of the Type 1 font file after it has been decoded.
len3	Length in bytes of the portion of the Type 1 font file that contains the 512 zeros, plus the cleartomark operator, plus any following data.
hasDW	If true , the dw and w fields are used; if false , they are not used.
dw	(Optional) Default width for glyphs in a CIDFont. See Section 5.6.3 in the PDF Reference for more information.

w	A Cos array of a set of lists that define the widths for the glyphs in the CIDFont. Each list can specify individual widths for consecutive CIDs, or one width for a range of CIDs. See Section 5.6.3 on character widths in CIDFonts in the PDF Reference for information on the format of this array.
hasDW2	If true , the dw2 and w2 fields are used; if false , they are not used.
dw2	<i>(Optional; applies only to CIDFonts that are used for vertical writing)</i> The default metric for writing mode 1. This entry is an array of two numbers: the y component of the position vector and the y component of the displacement vector for writing mode 1. The x component of the position vector is always half the width of the character. The x component of the displacement vector is always 0. The default value is [880 -1000]. For information on writing mode 1, see Section 5.6.3 on vertical writing in the PDF Reference .
w2	<i>(Optional; applies only to CIDFonts that are used for vertical writing)</i> A Cos array defining the metrics for vertical writing. Its format is similar to the format of the array in w . It defines the x and y components of the position vector, and the y component of the displacement vector. The x component of the displacement vector is always 0. See Section 5.6.3 on character widths in CIDFonts in the PDF Reference for information on the format of this array.
toUnicodeLen	<i>(Optional)</i> Length of toUnicodeStm .
toUnicodeStm	<i>(Optional)</i> A stream containing a CMap that defines the mapping from character codes to Unicode values. This entry is recommended for fonts that do not use one of the predefined CMaps. If present, this allows strings in the encoding to convert to Unicode values for export to other applications or clients. For more information, see Section 5.6 on Type 0 fonts in the PDF Reference .
cidToGidMapStm	A stream contain the mapping from CID to glyphindex ("GID"). The glyphindex for a particular CID value <i>c</i> is a 2-byte value stored in bytes $2*c$ and $2*c+1$; the first byte is the high-order byte.
panoseNo	A 12-byte string containing the Family Class ID, Family SubClass ID, and 10 bytes for the Panose classification number for the font. For additional details on the Panose number, see <i>Japanese TrueType Font Property Selection Guidelines</i> by the TrueType Conference Technical Committee.

fd	A Cos dictionary identifying a subset of characters in a CIDFont. The values are dictionaries with entries that override the values in the FontDescriptor dictionary for the subset of characters. See Section 5.6 in the PDF Reference for more information.
cidSetStm	A stream identifying which CIDs are present in the CIDFont file. It is required if the CIDFont file is embedded and only contains a subset of the glyphs in the character collection defined by the CIDSystemInfo. If this entry is missing, then it is assumed that the CIDFont file contains all the glyphs for the character collection. The stream's length should be rounded up to the nearest multiple of 8. The bits should be stored in bytes with the high-order bit first. Each bit corresponds to a CID. The first bit of the first byte corresponds to CID 0, the next bit corresponds to CID 1, and so on. If the subset contains a CID, the bit for that CID should be set. For compactness, the stream may use one of the compression filters to encode the data. For more information, see Section 5.6 in the PDF Reference .
flags	One of the PDEFontCreateFlags describing how to embed and subset the font.
mmDesignVec	Pointer to multiple master font design vector.

PDEFontInfoP

PDEFontInfoRec

```
typedef struct _t_PDEFontInfo {
    ASAtom name;
    ASAtom type;
    ASAtom charSet;
    ASAtom encoding;
    ASInt16 wMode;
} PDEFontInfoRec, *PDEFontInfoP;
```

Description

PDEFont information.

Header File

PDSysFontExpT.h

Related Methods

PDSysFontGetInfo

Members

name	An ASAtom for font name, as in "CourierStd."
type	An ASAtom for font type, "Type 1," "TrueType," and so on.
charSet	An ASAtom for "Roman" or ASAtomNull. If "Roman," the characters must be a subset of the Adobe Standard Roman Character Set.
encoding	An ASAtom for font encoding, as in WinAnsiEncoding.
wMode	Writing mode: 0 = horizontal; 1 = vertical.

PDEGraphicState

PDEGraphicStateP

```
typedef struct _t_PDEGraphicState {  
    ASUns32 wasSetFlags;  
    PDEColorSpec fillColorSpec;  
    PDEColorSpec strokeColorSpec;  
    PDEDash dash;  
    ASFixed lineWidth;  
    ASFixed miterLimit;  
    ASFixed flatness;  
    ASInt32 lineCap;  
    ASInt32 lineJoin;  
    ASAtom renderIntent;  
    PDEExtGState extGState;  
} PDEGraphicState, *PDEGraphicStateP;
```

Description

Attributes of a [PDEElement](#) or a [PDEText](#) sub-element.

Header File

PEExpT.h

Related Methods

[PDEDefaultGState](#)
[PDETextAdd](#)

Members

wasSetFlags	PDEGraphicStateWasSetFlags indicating if an attribute has been set. NOTE: Support for these flags is not complete. For compatibility, you should set them, but do not depend on reading their values back. The intended use is with XObject Forms to indicate whether the value is inherited or explicitly set.
fillColorSpec	Fill color specification. The default value is DeviceGray , fixedZero .
strokeColorSpec	Stroke color specification. The default value is DeviceGray , fixedZero .
dash	Dash specification. The default value is [0, 0].

lineWidth	Line width, corresponding to the w (setlinewidth) operator. The default value is fixedOne .
miterLimit	Miter limit, corresponding to the M (setmiterlimit) operator. The default value is fixedTen .
flatness	Line flatness, corresponding to the i (setflat) operator. The default value is fixedZero .
lineCap	Line cap style, corresponding to the J (setlinecap) operator. The default value is 0 .
lineJoin	Line join style, corresponding to the j (setlinejoin) operator. The default value is 0 .
renderIntent	A color rendering intent, corresponding to the Intent key in the image dictionary. The default value is 0 .
extGState	An extended graphics, corresponding to the gs operator. The default value is CosObj , NULL .

PDEGraphicStateWasSetFlags

```
typedef enum {
    kPDEFillCSpaceWasSet = 0x0001,
    kPDEFillCValueWasSet = 0x0002,
    kPDEStrokeCSpaceWasSet = 0x0004,
    kPDEStrokeCValueWasSet = 0x0008,
    kPDEDashWasSet = 0x0010,
    kPDELineWidthWasSet = 0x0020,
    kPDEmiterLimitWasSet = 0x0040,
    kPDEFlatnessWasSet = 0x0080,
    kPDELineCapWasSet = 0x0100,
    kPDELineJoinWasSet = 0x0200,
    kPDERenderIntentWasSet = 0x0400,
    kPDEExtGStateWasSet = 0x0800
} PDEGraphicStateWasSetFlags;
```

Description

Bit flags indicating the graphics state that was set.

Header File

PEExpT.h

Related Methods

[PDEDefaultGState](#)

[PDETextAdd](#)

Members

kPDEFillCSpaceWasSet	A fill color space was set, corresponding to the cs (setcolorspace) operator.
kPDEFillCValueWasSet	A color fill value was set, corresponding to the sc (setcolor) operator.
kPDEStrokeCSpaceWasSet	A color space stroke value was set, corresponding to the CS (setcolorspace) operator.
kPDEStrokeCValueWasSet	A color stroke value was set, corresponding to the SC (setcolor) operator.
kPDEDashWasSet	A dash specification was set, corresponding to the d (setdash) operator.
kPDELineWidthWasSet	The line width was set, corresponding to the w (setlinewidth) operator.

kPDEMiterLimitWasSet	The miter limit was set, corresponding to the M (setmiterlimit) operator.
kPDEFlatnessWasSet	Line flatness was set, corresponding to the i (setflat) operator.
kPDELineCapWasSet	Line cap style was set, corresponding to the J (setlinecap) operator.
kPDELineJoinWasSet	Line join style was set, corresponding to the j (setlinejoin) operator.
kPDERenderIntentWasSet	A color rendering intent was set, corresponding to the Intent key in the image dictionary.
kPDEExtGStateWasSet	An extended graphics state was set, corresponding to the gs operator.



PDEGrayCalFlt

```
typedef struct _t_PDEGrayCalFlt {  
    PDEWhitePointFlt whitePoint;  
    PDEBlackPointFlt blackPoint;  
    float gamma;  
} PDEGrayCalFlt;
```

Description

Structure describing a **CalGray** color space.

Default PDEGrayCalFlt: PDEGrayCalFlt calGray = {{0, 0, 0}, {0, 0, 0}, 1};

Header File

PEExpT.h

Related Types

[PDEColorSpaceStruct](#)

Related Methods

[PDEColorSpaceCreate](#)

PDEICCBasedColorData

```
typedef struct _t_PDEICCBasedColorData {
    ASSize_t size;
    ASStm iccstream;
    ASUns32 nComps;
    PDEColorSpace altCs;
} PDEICCBasedColorData;
```

Description

Structure describing an **ICCBased** color space.

Header File

PEExpT.h

Related Types

[PDEColorSpaceStruct](#)

Related Methods

[PDEColorSpaceCreate](#)

Members

size	Size of the data structure. Must be set to sizeof (PDEICCBasedColorData) .
iccstream	Stream containing ICC Profile.
nComps	Number of color components (1, 3, or 4).
altCs	<i>(Optional)</i> Alternate color space.

PDEImageAttrFlags

```
typedef enum {
    kPDEImageExternal = 0x0001,
    kPDEImageIsMask = 0x0002,
    kPDEImageInterpolate = 0x0004,
    kPDEImageHaveDecode = 0x0008,
    kPDEImageIsIndexed = 0x0010,
/* Added in Acrobat 4.0 */
    kPDEImageMaskedByPosition = 0x0020,
    kPDEImageMaskedByColor = 0x0040
} PDEImageAttrFlags;
```

Description

Bit flags for [PDEImageAttrs](#). See Section 4.8.4 in the [PDF Reference](#) for more information on image attributes.

Header File

PEExpT.h

Related Methods

[PDEImageCreate](#)
[PDEImageGetAttrs](#)
[PDEImageGetData](#)

Members

kPDEImageExternal	Indicates image is an XObject .
kPDEImageIsMask	Indicates image is an image mask.
kPDEImageInterpolate	Indicates interpolate is true .
kPDEImageHaveDecode	Indicates there is a decode array.
kPDEImageIsIndexed	Indicates image uses an indexed color space.
kPDEImageMaskedByPosition	Indicates image has a Mask key containing an image mask stream.
kPDEImageMaskedByColor	Indicates image has a Mask key containing an array of color values.

PDEImageAttrs

PDEImageAttrsP

```
typedef struct _t_PDEImageAttrs {
    ASUns32 flags;
    ASInt32 width;
    ASInt32 height;
    ASInt32 bitsPerComponent;
    ASFixed decode[8];
    /* New in Acrobat 4.0 */
    ASAtom intent;
} PDEImageAttrs, *PDEImageAttrsP;
```

Description

Attributes of a [PDEImage](#).

Header File

PEExpT.h

Related Methods

[PDEImageCreate](#)
[PDEImageGetAttrs](#)
[PDEImageGetData](#)

Members

flags	PDEImageAttrFlags indicating image attributes.
width	Width of the image, corresponding to the Width key in the image dictionary.
height	Height of the image, corresponding to the Height key in the image dictionary.
bitsPerComponent	Number of bits used to represent each color component in the image, corresponding to the BitsPerComponent key in the image dictionary.
decode	An array of numbers specifying the mapping from sample values in the image to values appropriate for the current color space. These values correspond to the Decode key in the image dictionary.
intent	Color rendering intent, corresponding to the Intent key in the image dictionary.



PDEImageDataFlags

```
typedef enum {
    kPDEImageEncodedData = 0x0001
} PDEImageDataFlags;
```

Description

Flags for [PDEImageGetData](#), [PDEImageGetDataStm](#), [PDEImageSetData](#), and [PDEImageSetDataStm](#).

Header File

PEExpT.h

Related Methods

[PDEImageGetData](#)
[PDEImageGetDataStm](#)
[PDEImageSetData](#)
[PDEImageSetDataStm](#)

Members

kPDEImageEncodedData	Indicates filter is active; data is encoded.

PDEIndexedColorData

```
typedef struct _t_PDEIndexedColorData {
    ASSize_t size;
    PDEColorSpace baseCs;
    ASUns16 hival;
    char* lookup;
    ASUns32 lookupLen;
} PDEIndexedColorData;
```

Description

Structure describing an indexed color space.

Header File

PEExpT.h

Related Types

[PDEColorSpaceStruct](#)

Related Methods

[PDEColorSpaceCreate](#)

Members

size	Size of the data structure. Must be set to <code>sizeof(PDEIndexedColorData)</code> .
baseCs	Base color space.
hival	Highest color value.
lookup	Indexed color lookup data.
lookupLen	Number of bytes in lookup data.



PDELabCalFlt

```
typedef struct _t_PDELabCalFlt {  
    PDEWhitePointFlt whitePoint;  
    PDEBlackPointFlt blackPoint;  
    PDEColorRangeFlt rangeA, rangeB;  
} PDELabCalFlt;
```

Description

Structure describing a **L*a*b*** color space.

Default is {{0, 0, 0}, {0, 0, 0}, {-100, 100}, {-100, 100}};

Header File

PEExpT.h

Related Types

[PDEColorSpaceStruct](#)

Related Methods

[PDEColorSpaceCreate](#)

PDEPathElementType

Constant values that describe path segment operators in [PDEPath](#) elements.

Header File

PEExpT.h

Related Methods

[PDEPathAddSegment](#)

[PDEPathCreate](#)

[PDEPathSetData](#)

Members

kPDEMoveTo	Designates m (moveto) operator, which moves the current point.
kPDELineTo	Designates l (lineto) operator, which appends a straight line segment from the current point.
kPDECurveTo	Designates c (curveto) operator, which appends a Bézier curve to the path.
kPDECurveToV	Designates v (curveto) operator, which appends a Bézier curve to the current path when the first control point coincides with initial point on the curve.
kPDECurveToY	Designates y (curveto) operator, which appends a Bézier curve to the current path when the second control point coincides with final point on the curve.
kPDERect	Designates re operator, which adds a rectangle to the current path.
kPDEClosePath	Designates h (closepath) operator, which closes the current sub-path.

PDEPathOpFlags

```
typedef enum {
    kPDEInvisible = 0x00,
    kPDEStroke = 0x01,
    kPDEFill = 0x02,
    kPDEEOFill = 0x04
} PDEPathOpFlags;
```

Description

Bit flags for paint operators in a [PDEPath](#).

Header File

PEExpT.h

Related Methods

[PDEPathGetPaintOp](#)
[PDEPathSetPaintOp](#)

Members

kPDEInvisible	Path is neither stroked nor filled, so it is invisible.
kPDEStroke	Stroke the path, as with the S (stroke) operator.
kPDEFill	Fills the path, using the nonzero winding number rule to determine the region to fill, as with the f (fill) operator.
kPDEEOFill	Fills the path, using the even–odd rule to determine the region to fill, as with the f* (eofill) operator.



PDEPatternColorSpace

```
typedef PDEColorSpace PDEPatternColorSpace;
```

Description

A [PDEColorSpace](#) that describes a **Pattern** color space.

Header File

`PEExpT.h`

Related Types

[PDEColorSpaceStruct](#)

Related Methods

[PDEColorSpaceCreate](#)



PDEPSAttrs

PDEPSAttrsP

```
typedef struct _t_PDEPSAttrs {
    ASUns32 flags;
} PDEPSAttrs, *PDEPSAttrsP;
```

Description

Attributes of a [PDEPS](#) object.

Header File

PEExpT.h

Related Methods

- [PDEPSCreate](#)
- [PDEPSGetAttrs](#)

Members

flags	kPDEPSInline

PDERGBCalFlt

```
typedef struct _t_PDERGBCalFlt {  
    PDEWhitePointFlt whitePoint;  
    PDEBlackPointFlt blackPoint;  
    float redGamma;  
    float greenGamma;  
    float blueGamma;  
    float matrix[9];  
} PDERGBCalFlt;
```

Description

Structure describing a **CalRGB** color space. Same as **AGMRGBCalFlt** (Only available as part of the PDF Library SDK).

Default is `{{0, 0, 0}, {0, 0, 0}, 1, 1, 1, {1, 0, 0, 0, 1, 0, 0, 0, 1}}`;

Header File

PEExpT.h

Related Types

AGMRGBCalFlt (Only available as part of the PDF Library SDK)

[PDEColorSpaceStruct](#)

Related Methods

[PDEColorSpaceCreate](#)

PDESeparationColorData

```
typedef struct _t_PDESeparationColorData {
    ASSize_t size;
    ASAtom name;
    PDEColorSpace alt;
    CosObj tintTransform;
} PDESeparationColorData;
```

Description

Structure describing a separation color space.

Header File

PEExpT.h

Related Types

[PDEColorSpaceStruct](#)

Related Methods

[PDEColorSpaceCreate](#)

Members

size	Size of the data structure. Must be set to <code>sizeof (PDESeparationColorData)</code> .
name	Name of separation or colorant.
alt	Alternative colorspace.
tintTransform	The tintTransform dictionary or function. See Section 4.5.5 in the PDF Reference .

PDESftMaskCreateFlags

```
typedef enum {  
    kPDESftMaskTypeLuminosity    = 0x0001,  
    kPDESftMaskTypeAlpha        = 0x0002  
} PDESftMaskCreateFlags;
```

Description

Bitwise flags for use with [PDESftMaskCreate](#).

Header File

PDEExpT.h

Related Methods

[PDESftMaskCreate](#)

Members

kPDESftMaskTypeLuminosity	Specifies how the mask is to be computed.
kPDESftMaskTypeAlpha	Specifies how the mask is to be computed.



PDETextFlags

```
typedef enum {
    kPDETextRun = 0x0001,
    kPDETextChar = 0x0002,
    kPDETextPageSpace = 0x0004,
    kPDETextGetBounds = 0x0008
} PDETextFlags;
```

Description

Bit field used in [PDEText](#) methods.

Header File

PEExpT.h

Related Methods

- [PDETextAdd](#)
- [PDETextGetFont](#)
- [PDETextGetText](#)

Members

kPDETextRun	Text run.
kPDETextChar	Character (text run with only one character).
kPDETextPageSpace	Obtain the advance width in user space.
kPDETextGetBounds	Fill in the left and right bounds of the text run's bounding box.



PDETextRenderMode

Constant values indicating text rendering mode set by the **Tr** operator.

Header File

PEExpT.h

Related Methods

[PDETextCreate](#)

Members

kPDETextFill	Fill text.
kPDETextStroke	Stroke text.
kPDETextFillAndStroke	Fill and stroke text.
kPDETextInvisible	Text with no fill and no stroke (invisible).

PDETextState

PDETextStateP

```
typedef struct _t_PDETextState {
    ASUns32 wasSetFlags;
    ASFixed charSpacing;
    ASFixed wordSpacing;
    ASInt32 renderMode;
    ASFixed fontSize;
    ASFixed hScale;
    ASFixed textRise;
} PDETextState, *PDETextStateP;
```

Description

Attributes of a [PDEText](#) element.

Header File

PEExpT.h

Related Methods

- [PDETextAdd](#)
- [PDETextGetTextState](#)
- [PDETextRunSetTextState](#)

Members

wasSetFlags	PDETextStateWasSetFlags indicating if an attribute has been set. NOTE: Support for these flags is not complete. For compatibility, you should set them, but do not depend on reading their values back. The intended use is with XObject Forms to indicate whether the value is inherited or explicitly set. NOTE: PDFEdit ignores the wasSetFlags flag, so you must initialize the PDETextState fields.
charSpacing	Character spacing was set, corresponding to the Tc operator.
wordSpacing	Word spacing, corresponding to the Tw operator.
renderMode	Text rendering mode, corresponding to the Tr operator.
fontSize	(New in 5.0) Default is 1.

hscale	(New in 5.0) Default=100 (==100%)
textRise	(New in 5.0) Specifies the distance, in text space units that are not scaled, to move the baseline up or down from its default location. See Section 5.2.6 in the PDF Reference .

PDETextStateWasSetFlags

```
typedef enum {  
    kPDECharSpacingWasSet = 0x0001,  
    kPDEWordSpacingWasSet = 0x0002,  
    kPDERenderModeWasSet = 0x0004  
} PDETextStateWasSetFlags;
```

Description

Constant values describing the text state that was set.

Header File

PEExpT.h

Related Methods

[PDETextAdd](#)
[PDETextGetTextState](#)
[PDETextRunSetTextState](#)

Members

kPDECharSpacingWasSet	Character spacing was set, corresponding to the Tc operator.
kPDEWordSpacingWasSet	Word spacing was set, corresponding to the Tw operator.
kPDERenderModeWasSet	Text rendering mode was set, corresponding to the Tr operator.

PDEType

Constant values that specify types of [PDEObject](#), which is the superclass for [PDEContent](#), [PDEElement](#), [PDEClip](#), and so on.

Header File

PEExpT.h

Related Methods

[PDEObjectGetType](#)

Members

kPDEContent	PDEContent object.
kPDEText	PDEText object.
kPDEPath	PDEPath object.
kPDEImage	PDEImage object.
kPDEForm	PDEForm object.
kPDEXObject	PDEXObject object.
kPDEClip	PDEClip object.
kPDEFont	PDEFont object.
kPDEColorSpace	PDEColorSpace object.
kPDEExtGState	PDEExtGState object.
kPDEPlace	PDEPlace object.
kPDEContainer	PDEContainer object.
kPDSysFont	PDSysFont object.
kPDEPattern	PDEPattern object.
kPDEDeviceNColors	(New in 4.0) PDEDeviceNColors object.
kPDEShading	(New in 4.0) PDEShading object.
kPDEGroup	(New in 4.0) PDEGroup object.
kPDEUnknown	(New in 4.0) PDEUnknown object.
kPDEBeginContainer	(New in 5.0) PDEBeginContainer object.
kPDEEndContainer	(New in 5.0) PDEEndContainer object.

kPDEBeginGroup	(New in 5.0) PDEBeginGroup object.
kPDEEndGroup	(New in 5.0) PDEEndGroup object.
kPDEXGroup	(New in 5.0) PDEXObject object.
kPDESoftMask	(New in 5.0) PDESoftMask object.
kPDSysEncoding	(New in 5.0) PDSysEncoding object.
kPDEDoc	(New in 5.0) Reserved for future use.
kPDEPage	(New in 5.0) Reserved for future use.



PDEXGroupCreateFlags

Description

Constant value that specifies the type of transparency group to create.

Header File

`PEExpT.h`

Related Methods

[PDESoftMaskCreate](#)

Members

kPDEXGroupTypeTransparency	Creates a transparency XGroup object.



PDEWhitePointFlt

```
typedef PDEXYZColorFlt PDEWhitePointFlt;
```

Description

Structure describing a white point in a calibrated color space.

Header File

PEExpT.h

Related Types

[PDEBlackPointFlt](#)
[PDEGrayCalFlt](#)
[PDELabCalFlt](#)
[PDERGBCalFlt](#)

Related Methods

[PDEColorSpaceCreate](#)



PDEXYZColorFlt

```
typedef struct {  
    float x;  
    float y;  
    float z;  
} PDEXYZColorFlt;
```

Description

Structure describing a point in a calibrated RGB color space.

Header File

PEExpT.h

Related Types

[PDEBlackPointFlt](#)

[PDERGBCalFlt](#)

[PDEWhitePointFlt](#)

Related Methods

[PDEColorSpaceCreate](#)

PDFFileSpecHandler

```
typedef struct _t_PDFFileSpecHandler {
    ASSize\_t size;
    PDFFileSpecNewFromASPathProc NewFromASPath;
    PDFFileSpecAcquireASPathProc AcquireASPath;
    /* New for Acrobat 3.0 */
    PDLaunchActionProc LaunchAction;
    /* Added in Acrobat 4.0 */
    ASFileSys fileSys;
} PDFFileSpecHandlerRec, *PDFFileSpecHandler;
```

Description

Data structure that implements a file specification handler. It contains callbacks that implement the filespec handler's functions (converting from a file specification to an [ASPathName](#), creating a new file specification from an [ASPathName](#), and launching the specified file).

Header File

PDExpT.h

Related Methods

[PDRegisterFileSpecHandlerByName](#)

Members

size	Size of the data structure. Must be set to <code>sizeof (PDFFileSpecHandlerRec)</code> .
fileSys	The file system that is used to read data for this file specification handler.



PDFontAngle

```
typedef ASInt16 PDFontAngle;
```

Description

An italic angle value in degrees, for use in [PDFontMetrics](#).

Header File

```
PDExpT.h
```

PDFontEncoding

```
typedef AEnum8 PDFontEncoding;
```

Description

Constant values that specify a font's encoding.

Header File

PDExpT.h

Related Methods

[PDFontGetEncodingIndex](#)

Members

PDBuiltInEncoding	The encoding specified internally in the font. In the case of a Type 1 or MMType 1 font, this is specified by the Encoding value in the font's fontdict. In the case of TrueType fonts, this is the encoding specified by the default one-byte CMap for the platform.
PDMacRomanEncoding	MacRomanEncoding , as defined in Appendix D in the PDF Reference .
PDMacExpertEncoding	MacExpertEncoding , as defined in Appendix D in the PDF Reference .
PDWinAnsiEncoding	WinAnsiEncoding , as defined in Appendix D in the PDF Reference .
PDStdEncoding	StandardEncoding , as defined in Appendix D in the PDF Reference .
PDFDocEncoding	PDFDocEncoding , as defined in Appendix D in the PDF Reference . This will never be returned for a font; it is used internally.

PDFontFlags

```
typedef struct _t_PDFontFlags {
    ASInt32 notUsed;
    ASUns32 flags;
} PDFontFlags;
```

Description

Bit flags containing additional information about a font.

Header File

PXExpT.h

Values

notUsed	Not used. For backward compatibility.
flags	An OR of the PDFONTFLAGS_ values. All unused flags are off. As of Acrobat 6, just one PDFONTFLAGS_ value was defined— PDFONTFLAGS_USEDBYFORM , which indicates that the font is used by AcroForm.

PDFontMetrics

PDFontMetricsP

```
typedef struct _t_PDFontMetrics {
    ASFlagBits flags;
    ASFixedRect fontBBox;
    PDiFontMetric missingWidth;
    PDiFontMetric stemV;
    PDiFontMetric stemH;
    PDiFontMetric capHeight;
    PDiFontMetric xHeight;
    PDiFontMetric ascent;
    PDiFontMetric descent;
    PDiFontMetric leading;
    PDiFontMetric maxWidth;
    PDiFontMetric avgWidth;
    PDFontAngle italicAngle;
    /* Added in Acrobat 4.0 */
    PDFontStyles style;
    PDFontOffset baseLineAdj;
} PDFontMetrics, *PDFontMetricsP;
```

Description

Data structure containing information about a font's metrics. See Section 5.5.1 in the [PDF Reference](#) for more information about font metrics. You also can find information about Adobe Font Metrics (AFM) on the <http://partners.adobe.com/asn> web site.

Header File

PDExpT.h

Related Methods

[PDFontGetMetrics](#)

[PDFontSetMetrics](#)

Members

flags	Must be an OR of the Font Flags values. All unused flags must be off.
fontBBox	Font bounding box in 1000 EM units. (An EM is a typographic unit of measurement equal to the size of a font. In a 12-point font, an EM is 12 points.)

missingWidth	The width of missing char (.notdef).
stemV	The vertical stem width.
stemH	The horizontal stem width.
capHeight	The capital height.
xHeight	The X height.
ascent	The maximum ascender height.
descent	The maximum descender depth.
leading	The additional leading between lines.
maxWidth	The maximum character width.
avgWidth	The average character width.
italicAngle	The italic angle in degrees, if any.
style	Panose and sFamily class values.
baseLineAdj	Baseline adjustment, which is a vertical adjustment for font baseline difference and writing mode 1 (vertical). This should only be used for CIDFontType 2 fonts with font substitution. See Section 5.6.3 in the PDF Reference for more information.



PDFontOffset

```
typedef ASInt16 PDFontOffset;
```

Description

A font offset value, for use in [PDFontMetrics](#).

Header File

PDExpT.h

PDFontStyles

```
typedef struct _t_PDFontStyles {  
    ASUns8 sFamilyClassID;  
    ASUns8 sFamilySubclassID;  
    ASUns8 bFamilyType;  
    ASUns8 bSerifStyle;  
    ASUns8 bWeight;  
    ASUns8 bProportion;  
} PDFontStyles;
```

Description

Data structure containing Panose and sFamily class values for a font. Used in the [PDFontMetrics](#) structure. See Section 5.7.2 in the [PDF Reference](#) for more information. For additional details on the Panose number, see *Japanese TrueType Font Property Selection Guidelines* by the TrueType Conference Technical Committee.

Header File

PDExpT.h

Related Methods

[PDFontGetMetrics](#)

[PDFontSetMetrics](#)

Members

sFamilyClassID	Number that identifies the font family and determines the meaning of the remaining Panose digits. Possible families are Latin, Kanji, Hebrew, and so forth.
sFamilySubclassID	Number to identify the kind of family: text, decorative, handwritten, symbols, and so on.
bFamilyType	Number to identify the family type: text, decorative, handwritten, symbols, and so on.
bSerifStyle	Number that specifies the font's serif style, such as cove, obtuse cove, square, bone, and so forth.
bWeight	Number that specifies the font's weight, such as very light, heavy, black, and so on.
bProportion	Number that specifies the font's proportions, such as modern, expanded, condensed, mono-spaced, and so on.

PDGraphicEnumMonitor

```
typedef struct _t_PDGraphicEnumMonitor {
    ASSize_t size;
    PDGraphicEnumTextProc EnumText;
    PDGraphicEnumPathProc EnumPath;
    PDGraphicEnumImageProc EnumImage;
    PDGraphicEnumXObjectRefProc EnumXObjectRef;
    PDGraphicEnumSaveProc EnumSave;
    PDGraphicEnumRestoreProc EnumRestore;
    PDGraphicEnumCharWidthProc EnumCharWidth;
    PDGraphicEnumCacheDeviceProc EnumCacheDevice;
    /* New for Acrobat 3.0 */
    PDGraphicEnumXObjectRefMatrixProc EnumXObjectRefMatrix;
} PDGraphicEnumMonitorRec, *PDGraphicEnumMonitor;
```

Description

An array of callbacks to pass to [PDCharProcEnum](#), [PDFFormEnumPaintProc](#) or [PDPageEnumContents](#). One of the callbacks is called for every renderable object in the page contents. Pass **NULL** for a callback to *not* enumerate that type of object. Each array element must be either **NULL** or a valid callback. Enumeration of the page contents halts if the callback returns **false**.

NOTE: [PDPageEnumContents](#) is provided only for backwards compatibility. It has not been updated beyond PDF Version 1.1 and may not work correctly for newly-created PDF 1.2 or later files. You should use the PDFEdit API to enumerate page contents.

NOTE: In versions at least through Acrobat 2.1, enumeration does not stop even if a method returns **false**.

Header File

PDExpT.h

Related Methods

[PDCharProcEnum](#)
[PDFFormEnumPaintProc](#)
[PDPageEnumContents](#)

Members

size	Size of the data structure. Must be set to sizeof(PDGraphicEnumMonitorRec) .

PDGraphicEnumParams

```
typedef struct _t_PDGraphicEnumParams {
    ASUns32 size;
    PDOCContext clientOCContext;
    PDOCContext usedOCContext;
    PDGraphicEnumMonitor mon;
    void *monObj;
} PDGraphicEnumParamsRec, *PDGraphicEnumParams;
```

Description

Enumeration parameters used for optional-content drawing control in [PDFormEnumPaintProcWithParams](#) and [PDCharProcEnumWithParams](#). The parameters are the same as those passed to the original versions of these methods ([PDFormEnumPaintProc](#) and [PDCharProcEnum](#)) with the addition of an optional-content context that determines what contents are visible.

Header File

PDExpT.h

Related Methods

[PDFormEnumPaintProcWithParams](#)
[PDCharProcEnumWithParams](#)

Members

size	Size of the data structure.
clientOCContext	An optional-content context that determines what contents are visible. NULL uses the document's optional-content context, as returned by PDDocGetOCContext (pddoc) , which is equivalent to calling the version of the method without optional-content parameters. This context is copied and the copy is used in drawing. This allows a client to change its copy of the context without raising an exception.
useOCContext	Filled by the method with the context that will be used during enumeration. This is a copy of the context specified by clientOCContext .
mon	The graphic enumeration monitor.
monObj	Pointer to user-supplied data to pass to the monitor.



PDGraphicState

PDGraphicStateP

```
typedef struct _t_PDGraphicState {
    ASFixedMatrix ctm;
    ASFixed fillColor[4];
    ASFixed strokeColor[4];
    ASAtom fillCSpace;
    ASAtom strokeCSpace;
    ASFixed flatness;
    ASInt32 lineCap;
    ASFixed dashPhase;
    ASArraySize dashLen;
    ASFixed dashes[10];
    ASInt32 lineJoin;
    ASFixed lineWidth;
    ASFixed miterLimit;
} PDGraphicState, *PDGraphicStateP;
```

Description

Data structure containing information about the current graphics state.

Header File

PDExpT.h

Related Methods

[PDGraphicGetState](#)

PDHostSepsPlate

```
typedef struct _t_PDHostSepsPlateRec {
    ASSize_t size;
    ASAtom colorantName;
    ASUns8 whatToDo;
    ASUns32 wasColorSet;
    ASStm epsStm;
    ASFile file;
    ASPathName path;
    float frequency;
    float angle;
} PDHostSepsPlateRec, *PDHostSepsPlate;
```

Description

Specifies per-plate options when generating or printing color separations from Acrobat. Used in [PDHostSepsSpec](#). Some of the values are set internally by Acrobat.

Header File

PDExpT.h

Related Methods

[AVDocPrintSeparations](#)
[PDPageMakeSeparations](#)

Members

size	The size of the structure. Must be set to sizeof (PDHostSepsPlateRec) .
colorantName	Name of colorant from Separation or DeviceN colorspace, or a spot or process color name (Cyan, Magenta, Yellow, Black).
whatToDo	How to handle the colorant. Value can be: ● kEmitPlate ● kDontEmitPlate ● kConvertToProcess
wasColorSet	Set internally. Used to determine whether marks were made on this plate to aid in blank detection.
epsStm	Set internally. Stream into which Acrobat emits PostScript for this plate. NULL if whatToDo is not set to kEmitPlate .
file	Set internally. Used only when separating documents, to close the file afterward.



path	Set internally. Used only when separating documents, to reopen the stream on the next page.
frequency	The frequency for this ink. -1 to use the default value.
angle	The angle for this ink. -1 to use the default value.

PDHostSepsSpec

```
typedef struct _t_PDHostSepsSpecRec {
    ASSize\_t size;
    ASUns32 psLevel;
    ASBool binaryOK;
    ASBool emitAnnots;
    ASBool emitHalftones;
    ASBool emitTransferFuncs;
    ASBool emitSeparableImagesOnly;
    ASBool suppressCJKSubstitution;
    ASEnum8 emitFontOption;
    ASBool TTasT42;
    ASBool printerHasFarEastFonts;
    ASUns32 transparencyLevel;
    ASBool useCMYKWorkingColorspace;
    char destProfile[256];
    ASBool convertToProcessUsingOPP;
    ASUns32 numPlates;
    PDHostSepsPlate *plates;
    PDOContext ocContext;
    ASBool applyOCGPrintOverrides;
    ASBool negative;
    ASEnum8 mirrorprint;
    ASUns32 whichMarks;
    ASBool western;
} PDHostSepsSpecRec, *PDHostSepsSpec;
```

Description

Used to control the generation of print color separations from Acrobat.

Header File

PDExpT.h

Related Methods

[AVDocPrintSeparations](#)
[PDPageMakeSeparations](#)

Members

size	The size of the structure. Must be set to <code>sizeof (PDHostSepsSpecRec)</code> .

psLevel	The PostScript printing level. 2 means emit as level 2, 3 means level 3. Used if the emitToPrinter or emitToFile print parameter is true .
binaryOK	true if a binary channel to the printer is supported, false otherwise. Used if the emitToPrinter or emitToFile print parameter is true .
emitAnnots	When true , emit annotations.
emitHalftones	When true , emit halftones.
emitTransferFuncs	When true , emit transfer functions.
emitSeparableImagesOnly	When true , emit separable images only.
suppressCJKSubstitution	When true , suppress CJK substitution.
emitFontOption	Font output options. Can be: ● kHSEmitFontNoFonts: Embed no fonts. ● kHSEmitFontEmbeddedFonts: Emit all embedded fonts. ● kHSEmitFontAllFonts: Emit all fonts.
TTasT42	When true , send TrueType fonts as TrueType fonts (level 3 and most level 2 PS printers). When false , convert TT to T1 (typically desirable only for Level 1 PS where no TT handling is present).
printerHasFarEastFonts	true means do not include far-east fonts.
transparencyLevel	The transparency level, 0-100.
useCMYKWorkingColorspace	When true , color manage DeviceCMYK. When false , pass it directly onto the process plates.
destProfile	The profile description of a valid CMYK profile, such as the strings seen in the Color Management panel of the General Preferences dialog; for example, "U.S. Web Coated (SWOP) v2".
convertToProcessUsingOPP	When true , use overprint preview (OPP) for convert to process, better simulating what would happen if the spot ink were really used.
numPlates	Number of items in the plates array.



plates	List of the colorant names and what to do with them for separations.
ocContext	The optional-content context to use for visibility state information, or NULL to use the document's current states in the default context.
applyOCGPrintOverrides	When true , apply print-specific visibility state settings from the optional-content group.
negative	When true , invert the plate.
mirrorprint	One of the following constants: kPDPrintFlipNone = 0x01, kPDPrintFlipX = 0x02, kPDPrintFlipY = 0x04, kPDPrintFlipXY = 0x08 Mirroring is done in the PostScript output stream.
whichMarks	Page mark indication. A bit-wise OR of the PDPageMarkFlags values.
western	When true , use western style for page marks.



PDiFontMetric

```
typedef ASInt16 PDiFontMetric;
```

Description

A font metric value (never negative), for use in [PDFontMetrics](#).

Header File

```
PDExpT.h
```

PDImageAttrs

PDImageAttrsP

```
typedef struct _t_PDImageAttrs {
    PDImageScalar width;
    PDImageScalar height;
    PDCount bitsPerComponent;
    ASBool imageMask;
    ASBool interpolate;
    ASBool haveDecode;
    ASFixed decode[8];
    ASAtom colorSpaceName;
    ASBool isIndexed;
    PDCount hiVal;
    CosObj colorSpace;
    ASArraySize dataLen;
    /* Added in Acrobat 4.0 */
    PDCount comps;
} PDImageAttrs, *PDImageAttrsP;
```

Description

Data structure containing information about the attributes of an image. See Section 4.8 in the [PDF Reference](#) for more information.

Header File

PDExpT.h

Related Methods

- [PDImageGetAttrs](#)
- [PDInlineImageGetAttrs](#)

Members

width	(Required) Width of the source image in samples.
height	(Required) Height of the source image in samples.
bitsPerComponent	(Required) The number of bits used to represent each color component.
imageMask	(Optional) true if the image should be treated as a mask; false otherwise.

interpolate	<i>(Optional)</i> true if interpolation is performed; false otherwise. Interpolation attempts to smooth transitions between sample values.
haveDecode	true if decode is used; false otherwise.
decode	<i>(Optional)</i> An array of numbers specifying the mapping from sample values in the image to values appropriate for the current color space.
colorSpaceName	ASAtom representing the color space name.
isIndexed	true if the color space is indexed; false otherwise.
hiVal	<i>(Optional)</i> Used if isIndexed is true . Colors are specified by integers in the range 0 to hiVal .
colorSpace	<i>(Required for images, not allowed for image masks)</i> Cos object of the color space used for the image samples.
dataLen	Length of sample data, in bytes.
comps	Number of components in colorSpace . For instance, comps is 3 for an RGB color space.



PDImageScalar

```
typedef ASInt32 PDImageScalar;
```

Description

A signed measurement of an image offset, for use in [PDImageAttrs](#).

Header File

```
PDExpT.h
```

PDLayoutMode

```
typedef AEnum8 PDLayoutMode;
```

Description

Constant values that define the layout of a document. The layout can be set as the viewer's `avpPageViewLayoutMode` preference (set by [AVAppSetPreference](#)) or in a view of a document by the `pageViewLayoutMode` field in [AVDocViewDef](#) (set by [AVDocGetViewDef](#)).

Header File

`PDExpT.h`

Related Methods

[AVDocGetViewDef](#)

[AVPageViewGetLayoutMode](#)

[AVPageViewSetLayoutMode](#)

Values

<code>PDLayoutDontCare</code>	<i>(Default)</i> Use the user preference when opening the file, as specified in the <code>avpPageViewLayoutMode</code> preference, set by AVAppSetPreference .
<code>PDLayoutSinglePage</code>	Use single-page mode, as in pre-Acrobat 3.0 viewers.
<code>PDLayoutOneColumn</code>	Use one-column continuous mode.
<code>PDLayoutTwoColumnLeft</code>	Use two-column continuous mode with first page on left.
<code>PDLayoutTwoColumnRight</code>	Use two-column continuous mode with first page on right.

PDLLinkAnnotBorder

```
typedef struct _t_PDLLinkAnnotBorder {  
    ASInt32 hCornerRadius;  
    ASInt32 vCornerRadius;  
    ASInt32 width;  
    ASInt32 dashArrayLen;  
    ASFixed dashArray [PDAnnotMaxDashes] ;  
} PDLLinkAnnotBorder;
```

Description

The border's dash pattern is specified by **dashArray** and **dashArrayLen**. This is a normal PostScript dash pattern (an array of on/off stroke lengths used cyclically) except that zero is always used as the offset ("phase") and the number of array entries is limited to **PDAnnotMaxDashes**. The number of valid **dashArray** entries is specified by **dashArrayLen**—a **dashArrayLen** of zero specifies a solid border.

Header File

PDExpT.h

Related Methods

[PDLLinkAnnotGetBorder](#)

[PDLLinkAnnotSetBorder](#)

PDOCConfigBaseState

```
typedef ASUns8 PDOCConfigBaseState;
```

Description

Enumerated type that contains the legal values for the **BaseState** key in an optional-content configuration dictionary ([PDOCConfig](#)). When you initialize a [PDOCContext](#) with the value **kOCCInit_FromConfig**, this enumeration represents the starting state of the optional-content groups (OCGs) before the contents of the configuration's ON and OFF OCG lists are processed.

If the base state is **kPDOC_BaseStateUnchanged** and the [PDOCConfig](#) is just being constructed, the current states of the OCGs from the [PDDoc](#)'s own [PDOCConfig](#) are used.

Header File

```
PXExpT.h
```

Related Methods

```
PDOCConfigGetInitState
PDOCConfigSetInitState
```

Values

kPDOC_BaseStateOFF	The starting state of OCGs is OFF.
kPDOC_BaseStateON	The starting state of OCGs is ON.
kPDOC_BaseStateUnchanged	Use the current states of OCGs from the PDDoc 's PDOCConfig .

PDOCContextChangeType

```
typedef ASUns8 PDOCContextChangeType;
```

Description

An enumeration of types of changes that can cause visibility to change within a given context. This enumeration is used in the optional-content change notifications.

Header File

PXExpT.h

Related Notifications

[PDOCContextDidChange](#)
[PDOCContextWillChange](#)

Values

kPDOCGState	Optional-content group state changing.
kPDOCContextDrawEnumType	Optional-content context's PDOCDrawEnumType changing.
kPDOCContextNonOCDrawing	Context's non-OC drawing changing.
kPDOCContextIntent	Context's intent changing.
kPDOCContextInit	Context being reset using PDOCContextInit .

PDOContextInitPolicy

```
typedef ASUns8 PDOContextInitPolicy;
enum {
    kOCCInit_OFF,
    kOCCInit_ON,
    kOCCInit_FromOtherContext,
    kOCCInit_FromConfig
};
```

Description

An enumeration of initialization policies for optional-content contexts. The policy determines how to initialize the state of optional-content groups (OCGs) when creating or initializing a new context.

Header File

PXExpT.h

Related Methods

[PDOContextNew](#)
[PDOContextInit](#)

Members

kPDOContextInit_OFF	The starting state of OCGs is OFF.
kPDOContextInit_ON	The starting state of OCGs is ON.
kPDOContextInit_FromOtherContext	Use the current states of OCGs from a specified context.
kPDOContextInit_FromConfig	Use the current states of OCGs from a specified configuration.

PDOCDrawEnumType

```
typedef ASUns8 PDOCDrawEnumType;
enum {
    kPDOC_VisibleOC = 0,
    kPDOC_Alloc,
    kPDOC_NoOC,
    kPDOC_LastDrawEnumType = kPDOC_NoOC
};
```

Description

Enumerated type that, together with the **NonOCDrawing** value, controls drawing or enumerating content on a page with optional content.

- Content that *is* marked as optional content is drawn or not drawn according to the **PDOCDrawEnumType** and the visibility state as determined by the OCGs and OCMDs.
- Content that is *not* marked as optional content is drawn when **NonOCDrawing** is **true**, and not drawn when **NonOCDrawing** is **false**.

Header File

PXExpT.h

Related Methods

[PDOCContextGetOCDrawEnumType](#)
[PDOCContextSetOCDrawEnumType](#)
[PDOCContextGetNonOCDrawing](#)
[PDOCContextSetNonOCDrawing](#)

Members

kPDOC_VisibleOC	Draw or enumerate optional content that is visible, according to the current state of OCGs and OCMDs. This is the normal default mode.
kPDOC_Alloc	Draw or enumerate all optional content, regardless of its visibility state. If the context's NonOCDrawing is true , all contents of document are shown.
kPDOC_NoOC	Draw or enumerate no optional content, regardless of its visibility state. If the context's NonOCDrawing is false , nothing is drawn, resulting in a blank page.



PDOCMDVisPolicy

```
typedef ASUns8 PDOCMDVisPolicy;
enum {
    kOCMDVisibility_AllOn,
    kOCMDVisibility_AnyOn,
    kOCMDVisibility_AnyOff,
    kOCMDVisibility_Alloff
};
```

Description

An enumeration of visibility policies for optional-content membership dictionaries (OCMDs), that are the legal values for **/P** key in the dictionary. The policy determines the visibility of content with respect to the ON-OFF state of optional-content groups (OCGs) listed in the dictionary.

Header File

PXExpT.h

Related Methods

[PDOCMDCreate](#)
[PDOCMDFindOrCreate](#)
[PDOCMDGetVisPolicy](#)



PDocOperation

Description

Enumerated data type. Specifies the type of changes that occurred for the [PDocPrintingTiledPage](#) and [PDocDidChangePages](#) notifications. Not all **Did** notifications have corresponding **Will** notifications. The exceptions are listed in the table.

Header File

PDocExpT.h

Methods

[PDocDeletePages](#)
[PDocInsertPages](#)
[PDocMovePage](#)
[PDocPageAddCosContents](#)
[PDocPageAddCosResource](#)
[PDocPageRemoveCosContents](#)
[PDocPageRemoveCosResource](#)
[PDocPageSetRotate](#)
[PDocPageSetCropBox](#)
[PDocPageSetMediaBox](#)

Members

<code>pdOpInsertPages</code>	Page insertion.
<code>pdOpDeletePages</code>	Page deletion.
<code>pdOpMovePages</code>	Page rearrangement.
<code>pdOpRotatePages</code>	Page rotation.
<code>pdOpCropPages</code>	Page cropping.
<code>pdOpAddResource</code>	Only PDocDidChangePages exists, not PDocWillChangePages .
<code>pdOpRemoveResource</code>	Only PDocDidChangePages exists, not PDocWillChangePages .
<code>pdOpAddContents</code>	Only PDocDidChangePages exists, not PDocWillChangePages .
<code>pdOpRemoveContents</code>	Only PDocDidChangePages exists, not PDocWillChangePages .
<code>pdOpSetMediaBox</code>	Page media box modification.

PDPageDrawFlags

```
typedef ASUns32 PDPageDrawFlags;
enum {
    kPDPageDoLazyErase,
    kPDPageUseAnnotFaces,
    kPDPageIsPrinting
};
```

Description

Bit flags indicating how a page is rendered.

Header File

PXExpT.h

Related Methods

PDPageDrawContentsPlaced (Only available with PDF Library SDK)

Values

kPDPageDoLazyErase	Erase the page while rendering only as needed.
kPDPageUseAnnotFaces	Draw annotation appearances.
kPDPageIsPrinting	The page is being printed.

PDPageInk

PDPageInkRec

```
typedef struct _t_PDPageInkRec {
    ASSize_t size;
    ASAtom colorantName;
    ASBool isProcessColor;
    ASUns8 whatToDo;
    ASUns8 r, g, b;
    float frequency;
    float angle;
} PDPageInkRec, *PDPageInk;
```

Description

Represents an ink used on a page.

Header File

PDExpT.h

Related Methods

[AVPageViewGetNumVisibleInks](#)
[AVPageViewGetVisibleInks](#)
[AVPageViewSetInkPreview](#)
[AVPageViewSetVisibleInks](#)
[PDPageEnumInks](#)

Members

size	The size of the structure. Must be set to sizeof (PDPageInkRec) .
colorantName	Name of colorant from Separation or DeviceN colorspace, or process color name.
isProcessColor	true if this is a process color, false if it is a spot color.
whatToDo	How to handle the colorant for a separation preview. Value can be: ● kEmitColorant ● kDontEmitColorant ● kConvertToProcess
r, g, b	RGB values for on-screen display of a colorswatch.



frequency	The frequency for this ink.
angle	The angle for this ink.



PDPageMarkFlags

```
typedef ASUns32 PDPageMarkFlags;  
enum {  
    kPDPageEmitColorBars = 0x0001,  
    kPDPageEmitRegMarks = 0x0002,  
    kPDPageEmitCropMarks = 0x0004,  
    kPDPageEmitBleedMarks = 0x0008,  
    kPDPageEmitPageInfo = 0x0010,  
    kPDPageEmitTrimMarks = 0x0020,  
    kPDPageEmitSlurMarks = 0x0040  
};
```

Description

Bit flags indicating which page marks are emitted for color separations.

Header File

PDExpT.h

Related Types

[PDHostSepsSpec](#)

Related Methods

[AVDocPrintSeparations](#)

[PDPageMakeSeparations](#)



PDPageMode

Constant value that specifies whether thumbnail images or bookmarks are shown.

Header File

PDExpT.h

Related Methods

[AVDocGetViewMode](#)

[AVDocSetViewMode](#)

[PDDocGetPageMode](#)

[PDDocSetPageMode](#)

Values

PDDontCare	Leaves view mode as is.
PDUseNone	Displays document, but neither thumbnails nor bookmarks.
PDUseThumbs	Displays document plus thumbnails.
PDUseBookmarks	Displays document plus bookmarks.
PDFullScreen	Displays document in full-screen viewing mode. This is equivalent to AVAppBeginFullScreen .



PDPageNumber

```
typedef ASInt32 PDPageNumber;
```

Description

A 0-based page number for use in [AVPageView](#) and [AVDoc](#) methods. Negative for special values.

Header File

```
PDExpT.h
```

Related Methods

- [AVDocGetPageText](#)
- [AVDocGetViewDef](#)
- [AVPageViewGetFirstVisiblePageNum](#)
- [AVPageViewGetLastVisiblePageNum](#)
- [AVPageViewGetNextView](#)
- [AVPageViewGetPageNum](#)
- [AVPageViewGetSelectedAnnotPageNum](#)
- [AVPageViewGetVisibleAnnotPage](#)
- [AVPageViewGoTo](#)
- [AVPageViewPageNumIsVisible](#)
- [AVPageViewSetPageNum](#)

Related Callbacks

- [AVDocSelectionGetAVRectProc](#)
- [AVSelectionPageRangeEnumProc](#)

PDPageRange

```
typedef struct _t_PDTextSelectRange {
    ASInt32 startPage;
    ASInt32 endPage;
    ASInt32 pageSpec;
} PDPageRange;
```

Description

Specifies a range of pages in a document. Page numbers begin with 0.

Header File

PDExpT.h

Related Types

PDPrintParams (Only available with PDF Library SDK)

Members

startPage	Starting page number.
endPage	Ending page number.
pageSpec	Pages in the range to print. Must be one of: PDAllPages , PDEvenPagesOnly , or PDOddPagesOnly . See Page Specification .

PDPageStmToken

PDPageStmTokenRec

```
typedef struct _t_PageStmToken {
    ASSize_t size;
    CosType type;
    ASUns32 flags;
    ASInt32 iVal;
    char sVal[kPDPageStmStringMax];
    ASSize_t sValLen;
} PDPageStmTokenRec, *PDPageStmToken;
```

Description

Data structure used by [PDPageStmGetToken](#). It contains information about the current page contents token.

Header File

PDExpT.h

Related Methods

[PDPageStmGetToken](#)

Members

size	Size of the data structure. Must be set to <code>sizeof(PDPageStmTokenRec)</code> .
type	The token's Cos type (<code>CosInteger</code> , <code>CosString</code> , <code>CosBoolean</code> , and so on).
flags	Additional information about the token. The only flag currently defined is <code>kPDPageStmTokenHexString</code> , meaning that the token is a hexadecimal encoded string.
iVal	Token's value if the token is a Cos integer, fixed number, or name.
sVal	Token's value if the token is a Cos string.
sValLen	Length of <code>sVal</code> , in bytes.

PDPathEnumMonitor

PDPathEnumMonitorRec

```
typedef struct _t_PDPathEnumMonitor {
    ASSize_t size;
    PDPathMoveToProc MoveTo;
    PDPathLineToProc LineTo;
    PDPathCurveToProc CurveTo;
    PDPathVCurveToProc VCurveTo;
    PDPathYCurveToProc YCurveTo;
    PDPathRectProc Rect;
    PDPathClosePathProc ClosePath;
} PDPathEnumMonitorRec, *PDPathEnumMonitor;
```

Description

Data structure containing callbacks used by [PDPathEnum](#). One callback is called for each path operator encountered; the callback to call depends on the operator.

Header File

PDExpT.h

Related Methods

[PDPathEnum](#)

Members

size	Size of the data structure. Must be set to <code>sizeof(PDPathEnumMonitorRec)</code> .



PDPathPaintOp

Constant value that specifies the path painting operator used on a path.

Header File

PDExpT.h

Related Methods

[PDPathGetPaintOp](#)

Values

pdPathNoPaint	Path is not painted.
pdPathOpClose	Path contains a closepath operator.
pdPathStroke	Path contains a stroke operator.
pdPathFill	Path contains a fill operator.
pdPathEOFill	Path contains an eofill operator.
pdPathClip	Path contains a clip operator.
pdPathEOClip	Path contains an eoclip operator.

PDPermReqObj

```
typedef ASInt16 PDPermReqObj;
```

Description

Constant value that describes the target object of a permissions request.

Header File

PDExpT.h

Related Callbacks

PDCryptAuthorizeExProc
PDCryptGetAuthDataExProc
PDCryptGetDocPermsProc

Related Methods

PDDocPermRequest

Values

PDPermReqObjDoc	Document object. Applicable operations: ● PDPermReqOprAll ● PDPermReqOprModify (Doc Info, open action, page label, modifying document) ● PDPermReqOprCopy (Copy to clipboard) ● PDPermReqOprAccessible ● PDPermReqOprSelect (Selection only) ● PDPermReqOprOpen ● PDPermReqOprSecure ● PDPermReqOprPrintHigh ● PDPermReqOprPrintLow ● PDPermReqOprFullSave ● PDPermReqOprImport (Non-PDF) ● PDPermReqOprExport (Non-PDF & text extract API, Search & Catalog) ● PDPermReqOprAny

PDPermReqObjPage	Page object. Applicable operations: ● PDPermReqOprAll ● PDPermReqOprCreate ● PDPermReqOprDelete ● PDPermReqOprCopy (duplicate page) ● PDPermReqOprRotate ● PDPermReqOprCrop ● PDPermReqOprInsert ● PDPermReqOprReplace ● PDPermReqOprReorder ● PDPermReqOprAny ● PDPermReqOprModify
PDPermReqObjLink	Link object (external file/URL link). Applicable operations: ● PDPermReqOprAll ● PDPermReqOprCreate ● PDPermReqOprDelete ● PDPermReqOprModify ● PDPermReqOprAny
PDPermReqObjBookmark	Bookmark object. Applicable operations: ● PDPermReqOprAll ● PDPermReqOprCreate ● PDPermReqOprDelete ● PDPermReqOprModify ● PDPermReqOprAny
PDPermReqObjThumbnail	Thumbnail object. Applicable operations: ● PDPermReqOprAll ● PDPermReqOprCreate ● PDPermReqOprDelete ● PDPermReqOprAny
PDPermReqObjAnnot	Annotation object. Applicable operations: ● PDPermReqOprAll ● PDPermReqOprCreate ● PDPermReqOprDelete ● PDPermReqOprModify ● PDPermReqOprSummarize ● PDPermReqOprImport ● PDPermReqOprExport ● PDPermReqOprAny

PDPermReqObjForm	Form object. Applicable operations: ● PDPermReqOprAll ● PDPermReqOprCreate ● PDPermReqOprDelete ● PDPermReqOprModify ● PDPermReqOprFillIn ● PDPermReqOprImport (Form data) ● PDPermReqOprExport (Form data) ● PDPermReqOprAny
PDPermReqObjSignature	Digital signature object. Applicable operations: ● PDPermReqOprAll ● PDPermReqOprCreate ● PDPermReqOprDelete ● PDPermReqOprModify ● PDPermReqOprFillIn (sign) ● PDPermReqOprAny

PDPermReqOpr

```
typedef ASInt16 PDPermReqOpr;
```

Description

Constant value that describes the target operation of a permissions request.

Header File

PDExpT.h

Related Callbacks

[PDCryptAuthorizeExProc](#)
[PDCryptGetAuthDataExProc](#)
[PDCryptGetDocPermsProc](#)

Related Methods

[PDDocPermRequest](#)

Values

PDPermReqOprAll = 1	Check all operations.
PDPermReqOprCreate	Generic.
PDPermReqOprDelete	Delete.
PDPermReqOprModify	Modify.
PDPermReqOprCopy	Copy
PDPermReqOprAccessible	For Accessibility use.
PDPermReqOprSelect	For document, selecting (not copying) text or graphics.
PDPermReqOprOpen	For document open.
PDPermReqOprSecure	For document—changing security settings.
PDPermReqOprPrintHigh	For document—regular printing.
PDPermReqOprPrintLow	For document—low quality printing.
PDPermReqOprFillIn	Form fill-in or to sign existing field.
PDPermReqOprRotate	Rotate.
PDPermReqOprCrop	Crop.
PDPermReqOprSummarize	For summarize notes.

PDPermReqOprInsert	Insert.
PDPermReqOprReplace	For page.
PDPermReqOprReorder	For page.
PDPermReqOprFullSave	For document.
PDPermReqOprImport	For notes and image.
PDPermReqOprExport	For notes. "ExportPS" should check print.
PDPermReqOprAny	Used for checking to see if any operation is allowed.
PDPermReqOprUnknownOpr	Used for error checking
PDPermReqOprSubmitStandalone	Used to submit forms from outside of the browser.
PDPermReqOprSpawnTemplate	Allows a form to spawn a template page.

PDPermReqStatus

```
typedef ASInt16 PDPermReqStatus;  
enum {  
    PDPermReqDenied = -1,  
    PDPermReqGranted = 0,  
    PDPermReqUnknownObject = 1,  
    PDPermReqUnknownOperation = 2,  
    PDPermReqOperationNA = 3  
};
```

Description

An constant value that describes whether **PDDoc** permissions allow a requested operation to be performed for a particular object in the document.

Header File

`PDExpT.h`

Related Callbacks

[PDCryptAuthorizeExProc](#)
[PDCryptGetAuthDataExProc](#)

Related Methods

[PDDocAuthorize](#)
[PDDocPermRequest](#)

Members

PDPermReqDenied	Request was denied.
PDPermReqGranted	Request was granted.
PDPermReqUnknownObject	The object is unknown.
PDPermReqUnknownOperation	The operation is unknown.
PDPermReqOperationNA	The operation is not applicable for the specified object.



PDPermReqVersion

```
#define PDPermReqVersion 0x0003
```

Description

An constant version value for [PDPermReqObj](#) and [PDPermReqOpr](#).

Header File

`PDExpT.h`

Related Callbacks

[PDCryptGetDocPermsProc](#)

PDPerms

Constant values that specify permissions which allow operations on a document file.

Header File

PDExpT.h

Related Methods

AVCryptGetPassword
 PDDocAuthorize
 PDDocGetPermissions (obsolete)
 PDDocPermRequest

Related Callbacks

PDCryptAuthorizeProc
 PDCryptGetAuthDataProc

Values

<code>pdPermOpen</code>	The user can open and decrypt the document.
<code>pdPermSecure</code>	The user can change the document's security settings.
<code>pdPermPrint</code>	The user can print the document. Page Setup access is unaffected by this permission, since that affects Acrobat's preferences—not the document's. In the Document Security dialog, this corresponds to the Printing entry.
<code>pdPermEdit</code>	The user can edit the document more than adding or modifying text notes (see also <code>pdPermEditNotes</code>). In the Document Security dialog, this corresponds to the Changing the Document entry.
<code>pdPermCopy</code>	The user can copy information from the document to the clipboard. In the Document Security dialog, this corresponds to the Content Copying or Extraction entry.
<code>pdPermEditNotes</code>	The user can add, modify, and delete text notes (see also <code>pdPermEdit</code>). In the Document Security dialog, this corresponds to the Authoring Comments and Form Fields entry.
<code>pdPermSaveAs</code>	The user can perform a "Save As..." If both <code>pdPermEdit</code> and <code>pdPermEditNotes</code> are disallowed, "Save" will be disabled but "Save As..." is enabled. The "Save As..." menu item is not necessarily disabled even if the user is not permitted to perform a "Save As...". NOTE: Not settable by clients.

<code>pdPermOwner</code>	The user can perform all operations, regardless of the permissions specified by the document. Unless this permission is set, the document's permissions will be reset to those in the document after a full save.
<code>pdPermSettable</code>	The OR of all operations that can be set by the user in the Standard Security dialog (<code>pdPermPrint</code> + <code>pdPermEdit</code> + <code>pdPermCopy</code> + <code>pdPermEditNotes</code>).
<code>pdPermAll</code>	All permissions.
<code>pdPermUser</code>	A convenience value, that is, (<code>pdPermAll</code> – <code>pdPermOpen</code> – <code>pdPermSecure</code>).

PDPrintWhat

```
typedef AEnum8 PDPrintWhat;
enum {
    PDPrintWhat_DOCUMENT,
    PDPrintWhat_DOCUMENT_AND_COMMENTS,
    PDPrintWhat_FORM_FIELDS_ONLY,
    PDPrintWhat_COUNT, /*internal*/
    PDPrintWhat_MIN=PDPrintWhat_DOCUMENT
};
```

Description

An constant value passed to the [PDDocWillPrintDocInMode](#) notification to specify the type of print operation being performed.

Members

<code>PDPrintWhat_DOCUMENT</code>	Print only the document.
<code>PDPrintWhat_DOCUMENT_AND_COMMENTS</code>	Print the document and associated annotations.
<code>PDPrintWhat_FORM_FIELDS_ONLY</code>	Print only the data within form fields.

Header File

PDExpT.h

Related Methods

[AVDocPrintPagesWithParams](#)

PDResourceEnumMonitor

PDResourceEnumMonitorRec

```
typedef struct _t_PDResourceEnumMonitor {
    ASSize_t size;
    PDResourceEnumFontProc EnumFont;
    PDResourceEnumXObjectProc EnumXObject;
    PDResourceEnumProcSetProc EnumProcSet;
    PDResourceEnumColorSpaceProc EnumColorSpace;
} PDResourceEnumMonitorRec, *PDResourceEnumMonitor;
```

Description

Data structure containing callbacks used when enumerating the resources of a form with [PDFormEnumResources](#) or [PDPageEnumResources](#).

NOTE: [PDPageEnumResources](#) is provided only for backwards compatibility. It has not been updated beyond PDF Version 1.1 and may not work correctly for newly-created PDF 1.2 or later files. You should use the PDFEdit API to enumerate page resources.

Header File

```
PDExpT.h
```

Related Methods

```
PDFormEnumResources
```

Members

size	Size of the data structure. Must be set to <code>sizeof(PDResourceEnumMonitorRec)</code> .

PDRotate

```
typedef AEnum16 PDRotate;  
enum {  
    pdRotate0 = 0,  
    pdRotate90 = 90,  
    pdRotate180 = 180,  
    pdRotate270 = 270  
};
```

Description

Constant values that specify page rotation, in degrees. Used for routines that set and get the value of a page's **Rotate** key.

Header File

PDExpT.h

Related Methods

[PDPageGetRotate](#)

[PDPageSetRotate](#)



PDSaveFlags

Constant values that specify options for saving a file.

Members

PDSaveFull	Write the entire file to the filename specified by newPath . In addition to this flag, the PDSaveCollectGarbage , PDSaveCopy , and PDSaveLinearized may be specified. Clients that use PDSaveFull are also encouraged to use PDSaveCollectGarbage .
PDSaveCollectGarbage	Remove unreferenced objects, often reducing file size. Clients are encouraged to use this flag. This flag can only be specified if PDSaveFull is also used.
PDSaveCopy	Write a copy of the file into the file specified by newPath , but keep using the old file. This flag can only be specified if PDSaveFull is also used.
PDSaveLinearized	Write the file linearized for page-served remote (network) access. This flag can only be specified if PDSaveFull is also used. During linearization, all Cos objects in the PDF file can be (and usually are) renumbered and recached in memory. Any PD- or Cos-level objects stored prior to linearization are <i>invalidated</i> if PDSaveLinearized is set. If invalid objects are used as a parameter in any method, a genErrBadParm (bad parameter) exception is raised. Thus, if any objects have been acquired, they need to be released prior to saving a PDDoc linearized. Register for the PDDocWillSave and PDDocDidSave notifications if the client has acquired any objects; release them after the PDDocWillSave notification and acquire them again after the PDDocDidSave notification.
PDSaveBinaryOK	It is OK to store binary data in the PDF file.
PDSaveIncremental	Only write those portions of the file that have changed.
PDSaveWithPSHeader	(<i>Obsolete</i>) Write a PostScript header as part of the saved file.



PDSaveForceIncremental	Perform an incremental save even if the save is to a different file or the document's version number has changed.
PDSaveKeepModDate	do not update ModDate in InfoDict.

Header File

`PDExpT.h`

Related Methods

[PDDocSaveWithParams](#)

Related Structure

[PDDocSaveParams](#)

PDSaveFlags2

```
typedef ASFlagBits PDSaveFlags2;
```

Description

Bitwise flags that specify save options.

The first three flags, **PDSaveUncompressed**, **PDSaveCompressed**, and **PDSaveCompressStructureOnly**, are mutually exclusive; they can all be off, but at most one can be on. They are all ignored unless the **PDSaveFull** flag is on.

PDSaveUncompressed = 1 << 0	Remove (decompress) all objects in the document. The result is compatible with all versions of PDF and Acrobat.
PDSaveCompressed = 1 << 1	Compress objects, without restrictions about which objects to compress. The result is compatible only with PDF 1.5 (Acrobat 6) or greater.
PDSaveCompressStructureOnly = 1 << 2	Compress the objects, and ONLY the objects, that are related to logical structure (for example, tagged PDF). The result is compatible with any version of PDF or Acrobat, but the compressed objects are usable only in PDF 1.5 (Acrobat 6).
PDSaveRemoveASCIIIFilters = 1 << 3	Remove ASCII85 filters from all streams. This flag is ignored unless PDSaveFull is on.
PDSaveAddFlate = 1 << 4	Encode any unencoded stream with Flate, except for Metadata streams, which are never encoded, and for streams that would be larger if encoded. This flag is ignored unless PDSaveFull is on.
PDSaveReplaceLZW = 1 << 5	Replace all LZW filters with FlateEncode filters. This flag is ignored unless PDSaveFull is on.
PDSaveOptimizeXObjects = 1 << 6	Merge identical forms and images, as determined by an MD5 hash of their contents.
PDSaveOptimizeContentStreams = 1 << 7	Look for common initial subsequences among content streams (the sequences of marking operators), and generate "substreams" that can be shared.

PDSaveOptimizeFonts = 1 << 8	Merge identical font descriptors and encodings. Does not merge the top-level font dict.
PDSaveOptimizeMarkedJBIG2Dictionaries = 1 << 9	Delete symbols specific to deleted images from JBIG2 dictionaries that could not be processed at the time of image deletion. Currently only effective after deleting pages or extracting pages.

Header File

`PDExpT.h`

Related Methods

[PDDocSaveWithParams](#)

Related Structure

[PDDocSaveParams](#)



PDSmallFlagBits

```
typedef ASUns16 PDSmallFlagBits;
```

Description

A flag value for use in [PDDocInsertPagesParams](#).

Header File

`ASExpT.h`

Related Methods

[PDDocInsertPages](#)

PDSMCInfo

PDSMCInfoP

```
typedef struct _t_PDSMCInfo
{
    ASSize_t size;
    ASInt32 mcid;
    ASBool directContent;
    CosObj containingStm;
    CosObj stmOwner;
    CosObj page;
} PDSMCInfo, *PDSMCInfoP;
```

Description

Information about how a marked content PDS object (a marked content kid) is included in its parent.

Header File

PDSExpT.h

Related Methods

[PDSMCGetInfo](#)

Members

size	The size of this data structure.
mcid	The marked-content identifier of the content kid in question.
directContent	If true , content is contained within the page's content stream; in this case, the fields containingStm and stmOwner are ignored. If false , content is contained within some other content stream, such as a Form XObject or an annotation.
containingStm	The stream object containing the marked content in question.
stmOwner	The object owning the stream, as defined by the StmOwn key in an OBJR .
page	Page on which the marked content is drawn, whether directly as part of page content or indirectly by being in a Form XObject or annotation drawn on that page.

PDFSysFontMatchFlags

Font matching flags for [PDFindSysFontForPDEFont](#) and [PDFindSysFont](#).

Header File

`PSFExpT.h`

Related Methods

[PDFindSysFont](#)

[PDFindSysFontForPDEFont](#)

Values

<code>kPDFSysFontMatchNameAndCharSet</code>	Match the font name and character set.
<code>kPDFSysFontMatchFontType</code>	Match the font type.
<code>kPDFSysFontMatchWritingMode</code>	Match the writing mode, that is, horizontal or vertical.

PDSysFontPlatData

PDSysFontPlatDataP

```

/* In Windows */
typedef struct _t_PDSysFontPlatData {
    ASSize_t size;
    LOGFONT* lf;
    ASPathName fontPath;
    ASPathName afmPath;
} PDSysFontPlatData, *PDSysFontPlatDataP;

/* In Mac OS */
typedef struct _t_PDSysFontPlatData {
    ASSize_t size;
    ASInt16 fontID;
    ASInt16 fontStyle;
} PDSysFontPlatData, *PDSysFontPlatDataP;

/* In UNIX */
typedef struct _t_PDSysFontPlatData {
    ASSize_t size;
    ASPathName fontPath;
    ASPathName afmPath;
} PDSysFontPlatData, *PDSysFontPlatDataP;

```

Description

System font platform-dependent data.

Header File

PDSysFontExpT.h

Related Methods

[PDSysFontAcquirePlatformData](#)
[PDSysFontReleasePlatformData](#)



Members

size	Size of the data structure. Must be set to sizeof (PDSysFontPlatData) .
lf	<i>(Windows only)</i> Windows LOGFONT structure defining font attributes.
fontPath	<i>(Optional for Windows)</i> A path to the font file. Set only if lf is not present.
afmPath	<i>(Optional for Windows)</i> A path to the font AFM file. Set only if lf is not present.
fontID	Macintosh FOND id.
fontStyle	Macintosh style value within that Fond. The default value is 0.

PDTextSelectRange

PDTextSelectRangeRec

```
typedef struct _t_PDTextSelectRange {  
    ASInt32 start;  
    ASInt32 end;  
    ASInt32 ofsStart;  
    ASInt32 ofsEnd;  
} PDTextSelectRangeRec, *PDTextSelectRange;
```

Description

Data structure used to specify a range of text in a text selection.

Use 0 for **ofsStart** and **ofsEnd** for whole-word selections. Nonzero values for **ofsStart** and **ofsEnd** are supported by **PDText** but are currently ignored by the Acrobat viewer's user interface code (which highlights only whole-word selections). If **ofsEnd** is 0, **end** is the first word *not* selected.

Header File

PDExpT.h

Related Methods

[PDTextSelectCreateRanges](#)
[PDTextSelectGetRange](#)

Members

start	Word offset of the word containing the start of the selection.
end	Word offset of the word containing the end of the selection.
ofsStart	Character offset into start of the start of the selection.
ofsEnd	Character offset into end of the end of the selection.



PDTextState

PDTextStateP

```
typedef struct _t_PDTextState {  
    PDFont font;  
    ASFixed charSpacing;  
    ASFixed wordSpacing;  
    ASFixed horizontalScale;  
    ASFixed leading;  
    ASFixed textRise;  
    ASFixed textSize;  
    ASInt32 renderMode;  
    ASFixedMatrix textMatrix;  
} PDTextState, *PDTextStateP;
```

Description

Data structure containing information about the current text state.

Header File

PDExpT.h

Related Methods

[PDTextGetState](#)

PDThumbCreationServer

```
typedef struct _t_PDThumbCreationServer {
    ASSize_t size;
    PDThumbCreationNotifyPageProc NotifyPage;
    PDThumbCreationGetThumbDataProc GetThumbData;
    PDThumbCreationDrawThumbProc DrawThumb
} PDThumbCreationServerRec, *PDThumbCreationServer;
```

Description

Data structure containing callbacks that implement a creation server. The callbacks implement the creation server functions.

Header File

PDExpT.h

Related Methods

[PDDocCreateThumbs](#)

Members

size	Size of the data structure. Must be set to <code>sizeof (PDThumbCreationServerRec)</code> .



PDTile

PDTileRec

```
typedef struct {
    ASUns32 overlap;
    ASBool center;
    ASInt32 marksflags;
    ASInt32 paperWidth;
    ASInt32 paperHeight;
    char *docTitle;
    char *docDate;
    char *docTime;
    ASInt32 col;
    ASInt32 row;
    ASInt32 numCols;
    ASInt32 numRows;
    ASInt32 xOffset;
    ASInt32 yOffset;
}PDTileRec, *PDTile;
```

Description

Printing flags.

Header File

PDExpT.h

Related Notifications

- [PDDocDidPrintTiledPage](#)
- [PDDocPrintingTiledPage](#)
- [PDDocWillPrintTiledPage](#)

Members

overlap	# of points to overlap (UI units may anything, clients convert to point).
center	Center the page's contents on the physical paper.

marksflags	What printer marks to emit. One of the following: kPDemitNoMarks 0 —nothing kPDemitWesternTileMarks 0x0001 —tile marks kPDemitEasternTileMarks 0x0002 —tile marks kPDemitSlug 0x0004 —emit info about document; name, page number, and so on.
paperWidth	Width of paper, client-provided, since client has PPD access.
paperHeight	The height of the paper.
docTitle	Title string for slug (optional).
docDate	Date string for slug (optional).
docTime	Time string for slug (optional).
The remaining fields are for communicating during print time: the current page's state, which page is being printed, and so on.	
col	Column.
row	Row.
numCols	Number of columns.
numRows	Number of rows.
xOffset	x offset.
yOffset	y offset.

PDWordFinderConfig

```
typedef struct _t_PDWordFinderConfig {
    ASSize_t recSize;
    ASBool disableTaggedPDF;
    ASBool noXYSort;
    ASBool preserveSpaces;
    ASBool noLigatureExp;
    ASBool noEncodingGuess;
    ASBool unknownToStdEnc;
    ASBool ignoreCharGaps;
    ASBool ignoreLineGaps;
    ASBool noAnnots;
    ASBool noHyphenDetection;
    ASBool trustNBSpace;
    ASBool noExtCharOffset;
    ASBool noStyleInfo;
    const ASUns16 *decomposeTbl;
    ASSize_t decomposeTblSize;
    const ASUns16 *charTypeTbl;
    ASSize_t charTypeTblSize;
} PDWordFinderConfigRec, *PDWordFinderConfig;
```

Description

A word finder configuration that customizes the way the extraction is performed. In the default configuration, all options are **false**.

Header File

```
PDExpT.h
```

Related Methods

```
PDDocCreateWordFinderEx
```

Members

recSize	The size of the current configuration record.
disableTaggedPDF	When true , disables Tagged PDF support and treats the document as non-Tagged PDF. Use this to keep the word finder in legacy mode when it is created with the latest algorithm version (WF_LATEST_VERSION).

noXYSort	When true , disables generating an XY-ordered word list. This option replaces the sort order flags in the older version of the word finder creation command (PDDocCreateWordFinder). Setting this option is equivalent to omitting the WXE_XY_SORT flag.
preserveSpaces	When true , the word finder preserves space characters during word breaking. Otherwise, spaces are removed from output text. When false (the default) you can add spaces later by considering the word attribute flag WXE_ADJACENT_TO_SPACE , but there is no way to restore the exact number of consecutive space characters.
noLigatureExp	When true , disables the expansion of ligatures using the default ligatures. Default ligatures are: fi, ff, fl, ffi, ffl, ch, cl, ct, ll, ss, fs, st, oe, OE.
noEncodingGuess	When true , disables guessing encoding of fonts that have unknown or custom encoding, when there is no ToUnicode table. Inappropriate encoding conversions can cause the word finder to mistakenly recognize non-Roman single-byte fonts as Standard Roman encoding fonts and extract the text in an unusable format. When this option is selected, the word finder avoids such unreliable encoding conversions and tries to provide the original characters without any encoding conversion for a client with its own encoding handling. Use the PDWordGetCharEncFlags method to detect such characters.
unknownToStdEnc	When true , assumes any font with unknown or custom encoding to be Standard Roman. This option overrides the noEncodingGuess option.
ignoreCharGaps	When true , disables converting large character gaps to space characters, so that the word finder reports a character space only when a space character appears in the original PDF content. This option has no effect on Tagged PDF.
ignoreLineGaps	When true , disables treating vertical movements as line breaks, so that the word finder determines a line break only when a line break character or special tag information appears in the original PDF content. This option has no effect on Tagged PDF.
noAnnots	When true , disables extracting text from text annotations. Normally, the word finder extracts text from the normal appearances of text annotations that are inside the page crop box.

noHyphenDetection	When true, disables finding and removing soft hyphens in non-Tagged PDF, so that the word finder trusts hard hyphens as non-soft hyphens. This option has no effect on Tagged PDF files. Normally, the word finder does not differentiate soft and hard hyphen characters in non-Tagged PDF files, because these are often misused.
trustNBSpace	When true, disables treating non-breaking space characters as regular space characters in non-Tagged PDF, so that the word finder preserves the space without breaking the word. This option has no effect on Tagged PDF files. Normally, the word finder does not differentiate breaking and non-breaking space characters in non-Tagged PDF files, because these are often misused.
noExtCharOffset	When true, disables generating extended character offset information to improve text extraction performance. The extended character offset information is necessary to determine exact character offset for character-by-character text selection. The beginning character offset of each word is always available regardless of this option, and can be used for word-by-word text selection with reasonable accuracy. When a client has no need for the detailed character offset information, it can use this option to improve the text extraction efficiency. There is a minor difference in the text extraction performance, and less memory is needed for the extracted word list.
noStyleInfo	When true, disables generating character style information to improve text extraction performance and memory efficiency. When you select this option, you cannot use PDWordGetNthCharStyle and PDWordGetStyleTransition with the output of the word finder.

decomposeTbl	A custom UTF-16 decomposition table. This table can be used to expand Unicode ligatures not included in the default ligature list. Each decomposition record contains a UTF-16 character code (either 16-bit or 32-bit surrogate), a replacement UTF16 string, and the delimiter 0x0000. For example: <pre>const ASUns16 myDecompTbl [] = {0x00b2, 0x0032, 0x0000, 0x00c6, 0x0061, 0x0065, 0x0000, 0xFB01, 0xFB01, 0x0000};</pre> This replaces superscript '2' with '2', the 'AE' ligature with 'a' and 'e', and disables the default handling of the 'fi' ligature. The word finder applies this substitution after identifying the character types, so the word-breaking process uses the character attributes of the original, rather than the replacement characters. See charTypeTbl below.
decomposeTblSize	Size of the decomposeTbl in bytes.
charTypeTbl	A custom character type table to enhance word breaking quality. Each character type record contains a region start value, a region end value, and a character type flag as defined in PDExpT.h. A character code is in UTF-16, either 16-bit or 32-bit surrogate. For example: <pre>const ASUns16 myCharTypeTbl [] = {0x0082, 0x0082, W_CNTL+W_WORD_BREAK, 0x00b2, 0x00b3, W_DIGIT}</pre> This identifies 0x0082 as a control-plus-word break character, and 0x00b2 and b3 as digits. If you need to change a character's type along with its character code, you must define the original character code in the custom character type table. For example, if 'a' is transformed to '1' in the decomposition table, 'a' should be transformed to W_DIGIT here, so that the word finder can recognize the replaced character, '1', as a number.
charTypeTblSize	Size of the charTypeTbl in bytes

PIHandshakeData_V0200

```
typedef struct {
    ASUns32 handshakeVersion;
    ASAtom appName;
    ASAtom extensionName;
    PIExportHFTsProcType exportHFTsCallback;
    PIImportReplaceAndRegisterProcType
        importReplaceAndRegisterCallback;
    PIInitProcType initCallback;
    PIUnloadProcType unloadCallback;
} PIHandshakeData_V0200;
```

Description

Structure describing client data for handshaking with the Acrobat viewer.

Header File

PIVersn.h

Related Callbacks

[PIHandshake](#)

Members

handshakeVersion	(Required) Handshake version. Always use HANDSHAKE_V0200 .
appName	(Optional) Name of host application.
extensionName	(Required) Name of the client.
exportHFTsCallback	(Optional) Callback to register the HFTs this client is exporting. May be NULL .
importReplaceAndRegisterCallback	(Optional) Callback to import other clients' HFTs, replace HFT functions, and register for notifications. May be NULL .
initCallback	(Required) Callback for client to initialize itself.
unloadCallback	(Optional) Callback to clean up when the client terminates. May be NULL .



POSIXPath_Ptr

```
typedef char* POSIXPath_Ptr;
```

Description

A C string containing a POSIX path (UTF-8 encoding).

Header File

ASExpT.h

Related Methods

[ASPlatformPathGetPOSIXPathPtr](#)



Predefined Cursors

Description

A group of predefined cursor constants.

Header File

AVExpT.h

Related Methods

[AVSysGetStandardCursor](#)

Values

Cursor	Design
ARROW_CURSOR	
IBEAM_CURSOR	
CROSSHAIR_CURSOR	
BOX_IBEAM_CURSOR	
HAND_CURSOR	
FIST_CURSOR	
ZOOM_IN_CURSOR	
ZOOM_OUT_CURSOR	
ZOOM_MAX_CURSOR	
LINK_CURSOR	
GROW_CURSOR	
BAR_IBEAM_CURSOR	
WAIT_CURSOR	
MOVEPAGE_CURSOR	
COPYPAGE_CURSOR	
MOVEPAGES_CURSOR	
COPYPAGES_CURSOR	
REPLACEPAGE_CURSOR	

Cursor	Design
REPLACEPAGES_CURSOR	
NOP_CURSOR	
THREAD_CURSOR	
WORDFINDER_CURSOR	
HIDDEN_CURSOR	
GROWTOPLEFT_CURSOR	
GROWBOTTOMLEFT_CURSOR	
MOVE_CURSOR	
HAND_THREAD_UP_CURSOR	
HAND_THREAD_END_CURSOR	
HAND_THREAD_UP_END_CURSOR	
HAND_THREAD_BEGIN_CURSOR	
THREAD_CONNECT_CURSOR	
THREAD_END_CURSOR	
VERT_IBEAM_CURSOR	
GROWLEFTRIGHT_CURSOR	
HIGHLIGHT_CURSOR	
GROWTOPBOTTOM_CURSOR	
CROPTOOL_CURSOR	 <i>New in 5.0</i>
CROPTOOL_SCISSORS_CURSOR	 <i>New in 5.0</i>
DRAGLEFTRIGHT_CURSOR	 <i>New in 5.0</i>
DRAGUPDOWN_CURSOR	 <i>New in 5.0</i>
VERTBEAMNOBAR_CURSOR	 <i>New in 5.0</i>

ProgressMonitor

ProgressMonitorRec

```
typedef struct _t_ProgressMonitor {
    ASSize_t size;
    PMBeginOperationProc beginOperation;
    PMEndOperationProc endOperation;
    PMSetDurationProc setDuration;
    PMSetCurrValueProc setCurrValue;
    PMGetDurationProc getDuration;
    PMGetCurrValueProc getCurrValue;
    PMSetTextProc setText;
} ProgressMonitorRec, *ProgressMonitor;
```

Description

Replaced by [ASProgressMonitor](#) in Acrobat 5.0.

Data structure containing callbacks that implement a progress monitor. The callbacks implement the progress monitor functions. A progress monitor is used to display progress during potentially time-consuming operations. Progress monitors are included as parameters in many API calls. Acrobat's built-in progress monitor can be obtained by calling [AVAppGetDocProgressMonitor](#).

Header File

ASExpT.h

Related Methods

- [AVAppGetDocProgressMonitor](#)
- [PDDocCreateThumbs](#)
- [PDDocSave](#)

Members

size	Size of the data structure. Must be set to <code>sizeof(ProgressMonitorRec)</code> .

Punctuation Characters

Description

Constants that specify various punctuation characters.

Header File

`WinAnsiResources.c`

Related Methods

Numerous

Character	Description
<code>acute</code>	´
<code>ampersand</code>	&
<code>asciicircumflex</code>	^
<code>asciitilde</code>	~
<code>asterisk</code>	*
<code>at</code>	@
<code>backslash</code>	\
<code>bar</code>	
<code>braceleft</code>	{
<code>braceright</code>	}
<code>bracketleft</code>	[
<code>bracketright</code>	]
<code>cedilla</code>	,
<code>cent</code>	¢
<code>colon</code>	:
<code>comma</code>	,
<code>copyright</code>	©
<code>currency</code>	¤

Character	Description
degree	°
dieresis	¨
divide	÷
dollar	\$
equal	=
exclamation point	!
exclamdown	¡
grave	`
greaterthan	>
guillemotleft	«
guillemotright	»
hyphen	-
lessthan	<
logicalnot	¬
macron	¯
multiply	*
numbersign	#
onehalf	½
onequarter	¼
ordfeminine	ª
ordmasculine	º
paragraph	¶
parenleft	(
parenright	)
percent	%
period	.



Character	Description
<code>periodcentered</code>	·
<code>plus</code>	+
<code>plusminus</code>	±
<code>question mark</code>	?
<code>questiondown</code>	¿
<code>quotedbl</code>	"
<code>quotesingle</code>	'
<code>registered</code>	®
<code>section</code>	§
<code>semicolon</code>	;
<code>slash</code>	/
<code>space</code>	" "
<code>sterling</code>	£
<code>threequarters</code>	$\frac{3}{4}$
<code>underscore</code>	—
<code>yen</code>	¥

Quad

```
typedef struct _t_Quad {  
    ASCoord tlh, tlv;  
    ASCoord trh, trv;  
    ASCoord blh, blv;  
    ASCoord brh, brv;  
} Quad, *QuadP;
```

Description

A structure that represents a quadrilateral defined by four corner points.

NOTE: The coordinate numeric types changed in Acrobat 6.

Header File

`ASExpT.h`

Related Methods

[AVPageViewInvertQuad](#)

Security Info Flags

Description

Constants that specify various information about the Acrobat viewer's security and permissions.

Header File

PDExpT.h

Related Methods

[PDDocGetNewSecurityInfo](#)

Members

pdInfoHasOwnerPW	The document has an owner password.
pdInfoHasUserPW	The document has a user password.
pdInfoCanPrint	The document can be printed.
pdInfoCanEdit	The document can be modified, for example by adding notes, links, or bookmarks (see also pdInfoCanEditNotes).
pdInfoCanCopy	The document text and graphics can be copied to the clipboard.
pdInfoCanEditNotes	The document's notes, but nothing else, can be modified (see also pdInfoCanEdit).



StdPassword

```
typedef char StdPassword[MAX_PWCHARS+1];
```

Description

Character string containing a password for the standard Acrobat security handler.

Header File

PDExpT.h

Related Types

[StdSecurityData](#)

StdSecurityData

StdSecurityDataRec

```
typedef struct _t_StdSecurityData {  
    os_size_t size;  
    ASBool newUserPW;  
    ASBool hasUserPW;  
    StdPassword userPW;  
    ASBool newOwnerPW;  
    ASBool hasOwnerPW;  
    StdPassword ownerPW;  
    PDPerms perms;  
    ASInt32 keyLength; /*new in Acrobat 5.0*/  
} StdSecurityDataRec, *StdSecurityData;
```

Description

Structure describing the data for the standard security handler provided in the Acrobat viewer.

Header File

PDExpT.h

Members

size	Size of the data structure. Must be set to sizeof (StdSecurityDataRec) .
newUserPW	true if the user password should be changed, false otherwise.
hasUserPW	true if a user password is provided, false otherwise.
userPW	The user password.
newOwnerPW	true if the owner password should be changed, false otherwise.
hasOwnerPW	true if an owner password is provided, false otherwise.
ownerPW	The owner password.
perms	The permissions to allow for a file. See <code>PDTypes.h</code> .
keyLength	(New in Acrobat 5.0) Encryption key length, in bytes.



Tool Button Flags

Constants that specify whether a toolbar button is external to the viewer or not.

Header File

AVExpT.h

Related Methods

[AVToolButtonSetExternal](#)

Members

TOOLBUTTON_INTERNAL	Indicates toolbar button is visible only in the viewer's toolbar.
TOOLBUTTON_EXTERNAL	Indicates toolbar button is visible only in the toolbar of an external application (such as a Web browser).



Transition Duration

Duration of a [PDTrans](#).

Header File

`PDExpT.h`

Related Methods

[PDTransNew](#)

Members

fxDefaultPageDuration	Constant used to indicate that there is no page timing information available.
fxDefaultTransDuration	Default duration for a transition.



WinPort

```
typedef struct _t_WinPort {  
    HWND hWnd;  
    HDC hDC;  
} WinPortRec, *WinPort;
```

Description

The **HWND** is that of the document window's [AVPageView](#) region (the portion of the window in which the PDF file's pages are drawn).

Header File

AVExpT.h

Related Methods

[AVPageViewAcquireMachinePort](#)

Word Attributes

Constants that specify various attributes of words.

Header File

`PDExpT.h`

Related Methods

[PDWordGetAttr](#)

[PDWordGetAttrEx](#)

Members

Group 0 (default group)	
WXE_ADJACENT_TO_SPACE	The character following the end of the word is a space (either an explicit space character encoded in a string, or one that appears implicitly because the drawing point was moved).
WXE_HAS_UNMAPPED_CHAR	One or more characters in the word cannot be represented in the output font encoding.
WXE_HAS_LIGATURE	The word contains a ligature.
WXE_HAS_DIGIT	One or more characters in the word are digits.
WXE_HAS_HYPHEN	There is a hyphen in the word.
WXE_HAS_SOFT_HYPHEN	There is a soft hyphen in the word.
WXE_HAS_PUNCTUATION	One or more characters in the word are punctuation marks. Other flag bits can be checked to test whether the punctuation was at the beginning of the word (WXE_HAS_LEADING_PUNC), the end of the word (WXE_HAS_TRAILING_PUNC), or elsewhere in the word.
WXE_HAS_LEADING_PUNC	The first character in the word is a punctuation mark. If this bit is set, WXE_HAS_PUNCTUATION will also be set.
WXE_HAS_TRAILING_PUNC	The last character in the word is a punctuation mark. If this bit is set, WXE_HAS_PUNCTUATION will also be set.
WXE_HAS_NONALPHANUM	The word contains a character outside the range of A-Z, a-z, 0-9.
WXE_HAS_LETTER	The word contains a character between A-Z or a-z.

WXE_HAS_UPPERCASE	The word contains a character between A-Z.
WXE_LAST_WORD_ON_LINE	The word is at the end of the current text line (that is, the word is followed by a line break).
WXE_ROTATED	The writing direction of the word is not in a multiple of 90 degrees or the bounding box of the text is skewed. This flag indicates that the quads of the word should be used to specify the highlight area correctly.
WXE_VERTICAL_FLOW	The writing direction of the word is either 90 or 180 degrees. This flag ignores the page rotation parameter of the page dictionary. Therefore, if the page is rotated 90 degrees, this flag will be set on each word that appears horizontally on the screen.
Group 1 (for word finders created with algorithm WF_VERSION_3 or higher)	
WXE_FRONT_TAB	Insert a tab character before this word when the text is passed to another application.
WXE_ENCODING_WARNING	Unreliable encoding conversion happens with this word. Use PDWordGetCharEncFlags method to get the WordFinder Character Encoding Flags .
WXE_REVERSE_DIRECTION	The text is rotated 180 or 90 degrees CCW on the page. When this flag is set, the text flow direction is either right-to-left or top-to-down.
WXE_WORD_IS_UNICODE	The text is in Unicode format.

WordFinder Character Encoding Flags

Constants that specify how reliably the word finder identified a character encoding.

Header File

PDExpT.h

Related Methods

[PDWordGetCharEncFlags](#)

Members

WXE_ENC_UNMAPPED	The font has no useful encoding information. The word finder copied the original character string from the PDF content. If the word finder is extracting the text in Unicode, it inserts additional bytes, 0x00s, to form 16-bit characters (for example, "0x41, 0x42" "0x00, 0x41, 0x00, 0x42"). A common example of this case is a Symbol font without a ToUnicode table. If WordFinder is extracting text in Unicode and the original character codes are in double-byte, those characters are mapped to UTF-16 Surrogate Supplementary Private Use Area-B (U+10000 to U+10FFFD. For example, "0x3881" --> "0xDB, 0xCE, 0xDC, 0x81". When no valid Unicode values are available during the word-finding process, these characters are treated as symbolic characters and each character becomes a word.
WXE_ENC_MISSING	The character code is not available in the output encoding space. This character is replaced by a space character.
WXE_ENC_NO_UCS	The word finder is not able to find a reliable Unicode value from the character. When insufficient encoding information is available in the PDF file, the word finder guesses the character encoding.
WXE_ENC_ACTUALT	The character comes from an ActualText, rather than the page content.

WordFinder Sort Order Flags

Constants that specify which tables the word finder is to fill.

Header File

PDExpT.h

Related Methods

[PDDocCreateWordFinder](#)

[PDDocCreateWordFinderUCS](#)

Members

WXE_PDF_ORDER	Enumerates words in the order they appear in the PDF file. This order may not be the same as that in which a person would read them on a page.
WXE_XY_SORT	Enumerates words sorted by their x- and y-coordinates on the page. For a page with a single column of text, this is usually close to the order in which a person would read the text on the page.



Lists

AVCommand Descriptions (Built-in Commands)

Built-in commands are organized into groups:

- [Comments Command Group](#)
- [Document Command Group](#)
- [JavaScript Command Group](#)
- [Page Command Group](#)
- [PDF Consultant Command Group](#)

Comments Command Group

Comment Command	Parameters
Delete All Comments <code>DeleteAll</code>	None
Summarize Comments <code>Summarize</code>	<code>kCommentsCmdKeySortBy</code> : <code>ASInt32</code> , one of: <code>kAVSortByPage</code> <code>kAVSortByAuthor</code> <code>kAVSortByModDate</code> <code>kAVSortByType</code> <code>kCommentsCmdKeyOutputPath</code> : <code>ASPathName</code> <code>kCommentsCmdKeySaveWithOriginal</code> : <code>ASBool</code> If <code>true</code> , the command writes the file to the path specified by the <code>OutputPath</code> parameter.

Document Command Group

Document Command	Parameters
Common parameter for all document commands	<code>kDocCmdKeyLeaveAsIs</code> : <code>kASValueCab</code> "LeaveAsIs" rules apply

Document Command	Parameters
Accessibility Checker AccessibilityCheck If keypages is selected page and AVDoc is NULL , the first page is processed.	kAccCheckCmdKeyCreateLog : ASBool Default is true .
	kAccCheckCmdKeyCreateComments : ASBool Default is false .
	kAccCheckCmdKeyFileSys : kASValuePointer kAccCheckCmdKeyPathText : ASText Log file destination. If FileSys is NULL , the log file is written to the destination directory. If PathText is not specified, the log is written to the same directory as the source document. If FileSys is supplied, the file is opened through that file system; otherwise the default is used.
	kAccCheckCmdKeyAlternateText : kASValueBool Default is true .
	kAccCheckCmdKeyLanguageSpecified : kASValueBool Default is true .
	kAccCheckCmdKeyCharEncodings : kASValueBool Default is true .
	kAccCheckCmdKeyCheckStructure : kASValueBool Default is true .
	kAccCheckCmdKeyFieldDescriptions : kASValueBool Default is true .
	kAccCheckCmdKeyPages : kASValueInteger Default is all pages. If keypages is selected page and avdoc is NULL , the first page is processed.
	kAccCheckCmdKeyFromPage : kASValueInteger Default is 0.
	kAccCheckCmdKeyToPage : kASValueInteger Default is 0.
	kAccCheckCmdKeyInBatch : kASValueBool Controls the dialog.

Document Command	Parameters
Document Summary GeneralInfo	kDocInfoCmdKeyTitle : AText kDocInfoCmdKeySubject : AText kDocInfoCmdKeyAuthor : AText kDocInfoCmdKeyKeywords : AText kDocInfoCmdKeyBinding : AText kDocCmdKeyLeaveAsIs : ASCab All values that are to be left untouched. If cabinet is not present, or a key is not present, it is assumed to be a valid value in the cab.
Embed All Thumbnails CreateAllThumbs	None
Print Print	None. If UIPolicy is Interactive, the standard print dialog is displayed; otherwise the default print settings are used.
Remove Embedded Thumbnails DeleteAllThumbs	None
Security DocSecurity	kDocCmdsKeyCryptHandler : kASValueAtom Default is ASAtomNull kDocCmdsKeySecuritySettings : kASValueCab
Format-specific image extraction commands	
Extract Images As JPEG ExtractAsJpeg	kExtractJpegCmdKeyCompression : kASValueInteger, one of: kJpegCompressionLow kJpegCompressionMediumLow kJpegCompressionMedium (default) kJpegCompressionMediumHigh kJpegCompressionHigh kExtractJpegCmdKeyFormat : kASValueInteger, one of: kJpegFormatBaseline (default) kJpegFormatOptimized kJpegFormatProgressive3 kJpegFormatProgressive4 kJpegFormatProgressive5

Document Command	Parameters
Extract Images As PNG ExtractAsPng	kExtractPngCmdKeyInterlace: kASValueInteger , one of: kPngInterlaceNone (default) kPngInterlaceAdam7 kExtractPngCmdKeyFilter: kASValueInteger , one of: kPngFilterNone kPngFilterSub kPngFilterUp kPngFilterAverage kPngFilterPaeth kPngFilterAdaptive (default)
Extract Image As TIFF ExtractAsTiff	kExtractTiffCmdKeyMonoCompression: kASValueInteger kExtractTiffCmdKeyColorCompression: kASValueInteger , one of: kTiffCompressionNone kTiffCompressionCCITT_G3 kTiffCompressionCCITT_G4 (default for mono) kTiffCompressionLzw (default for color) kTiffCompressionZip kTiffCompressionJpegLow kTiffCompressionJpegMediumLow kTiffCompressionJpegMedium kTiffCompressionJpegMediumHigh kTiffCompressionJpegHigh
Image extraction parameters common to all image formats	
Configured Must be set to true to mark the command parameters as being in a valid state.	kExtractImgCmdKeyConfigured: kASValueBool
ConversionFormat Set to the format that you are exporting to.	kExtractImgCmdKeyConvFormat: kASValueInteger , one of: kImgConversionFormatTiff kImgConversionFormatJpeg kImgConversionFormatPng Default is appropriate value—JPEG, TIFF, PNG

Document Command	Parameters
Overwrite Whether files are overwritten. Default is true . If set to false , and output file is found to exist, alert is show if in batch; otherwise the user is given the option to overwrite.	kExtractImgCmdKeyOverwriteFiles: kASValueBool Default is true .
Batch Used to control the format of the settings dialog. true gets the dest folder info.	kExtractImgCmdKeyInBatch: kASValueBool Default is true .
ColorSpace	kExtractImgCmdKeyColorSpace: kASValueInteger , one of: kColorSpaceAuto = 1 (default) kColorSpaceRGB kColorSpaceGrayscale kColorSpaceMonochrome
Resolution	kExtractImgCmdKeyResolution: kASValueInteger , one of: kImgResolutionAuto = -1 (default) kImgResolution72 = 72 kImgResolution96 = 96 kImgResolution150 = 150 kImgResolution300 = 300 kImgResolution600 = 600
DestDirectory Must be passed, or command returns an error.	kExtractImgCmdKeyDestDirectory: kASValuePathName Default is the kAVSCUser/kAVSFDocuments folder
BaseFileName Used to build the output file string. If not set, handler uses the name of the PDDoc parameter. Must be passed in the input's ASCab , not the params.	kExtractImgCmdKeyBaseFileName: kASValueString Default is source PDDoc filename

Document Command	Parameters
Set Open Options	kOpenOptionsCmdKeyPageNum : kASValueText For example, "1"
OpenInfo	
All defaults are "LeaveAsIs"	kOpenOptionsCmdKeyMagnification : kASValueText For example, "100%": Open action
	kOpenOptionsCmdKeyPageLayout : kASValueInteger For example, PDLayoutSinglePage
	kOpenOptionsCmdKeyPageMode : kASValueInteger For example, PDUseThumbs
	kOpenOptionsCmdKeyFullScreen : kASValueBool
	kOpenOptionsCmdKeyHideToolbar : kASValueBool
	kOpenOptionsCmdKeyHideMenubar : kASValueBool
	kOpenOptionsCmdKeyHideWindowUI : kASValueBool
	kOpenOptionsCmdKeyFitWindow : kASValueBool
	kOpenOptionsCmdKeyCenterWindow : kASValueBool
	kOpenOptionsCmdKeyDisplayDocTitle : kASValueBool

JavaScript Command Group

JavaScriptCommand	Parameters
Execute JavaScript	kExecJavaScriptName : kASValueText
JavaScript	Not currently used
	kExecJavaScriptCode : kASValueText

Page Command Group

Page Command	Parameters
Common parameters for all page commands	kPageCmdKeyGroup : kASValueInteger, one of: kAVPagesAll kAVPagesSelected kAVPagesFromTo kPageCmdKeyEvenOdd : kASValueInteger, one of: kAVPagesEvenOdd kAVPagesEven kAVPagesOdd kPageCmdKeyFrom : kASValueText kPageCmdKeyTo : kASValueText
Crop Pages CropPages	• kPageCmdKeyGroup default is kAVPagesAll. • kPageCmdKeyEvenOdd default is kAVPagesEvenOdd. • If cropping to bounding box, the dimension params are ignored. kCropPagesCmdKeyCropType : kASValueInteger, one of: kAVCropCustom (default) kAVCropToBoundingBox kCropPagesCmdKeyLeft : kASValueDouble Default is 0.0 kCropPagesCmdKeyRight : kASValueDouble Default is 0.0 kCropPagesCmdKeyTop : kASValueDouble Default is 0.0 kCropPagesCmdKeyBottom : kASValueDouble Default is 0.0
Delete Pages DeletePages	No specific parameters. • kPageCmdKeyEvenOdd not supported. • kPageCmdKeyGroup default is kAVPagesFromTo. • kAVPagesSelected must be combined with an AVDoc.

Page Command	Parameters
Insert Pages InsertPages	kInsertPagesCmdKeyInsertBefore: kASValueBool Default is false, meaning insert after. kInsertPagesCmdKeyPosition: kASValueInteger, one of: kAVPosRelativeToFirst kAVPosRelativeToLast kAVPosRelativeToPage (default) kInsertPagesCmdKeyPageString: kASValueText Default is 1 kInsertPagesCmdKeySrcFileName: kASValueText No use other than dialogs. kInsertPagesCmdKeySourcePathNames: kASValueCab Contains a series of ASPathNames
Number Pages NumberPages	• kPageCmdKeyGroup Default is kAVPagesAll. • kPageCmdKeyEvenOdd Default is kAVPagesEvenOdd kNumberPagesCmdKeyStyle: kASValueInteger, one of: kPageLabelStyleNone (default) kPageLabelStyleDecimal kPageLabelStyleRomanLower kPageLabelStyleRomanUpper kPageLabelStyleAlphaLower kPageLabelStyleAlphaUpper kNumberPagesCmdKeyStart: kASValueInteger Default is 1 kNumberPagesCmdKeyMergePrevious: kASValueBool Default is false. kNumberPagesCmdKeyPrefix: kASValueString Default is ""
Rotate Pages RotatePages	• kPageCmdKeyGroup default is kAVPagesAll. • kPageCmdKeyEvenOdd default is kAVPagesEvenOdd kRotatePagesCmdKeyDegrees: kASValueInteger Default is pdRotate90 kRotatePagesCmdKeyAction: kASValueInteger one of: kAVRotateLandscapePages kAVRotatePortraitPages kAVRotateEveryPage (default)

PDF Consultant Command Group

PDF Consultant Command	Parameters
Audit Space Usage SpaceAuditAgent	None. Entirely interactive.
Detect And Remove VirusCheck	kVirusCheckCmdKeyOptions : kASValueCab Default is stuff .
	kVirusCheckCmdKeyNumOptions : kASValueInteger Default is 5.
	kVirusCheckCmdKeyRemove : kASValueBool Default is false .
Optimize Space SpaceReductionAgent	kOptSpaceCmdKeyRemoveBmarks : kASValueBool Default is true .
	kOptSpaceCmdKeyRemoveLinks : kASValueBool Default is true .
	kOptSpaceCmdKeyRemoveDests : kASValueBool Default is true .
	kOptSpaceCmdKeyDestConversionType : kASValueInteger , one of: kOptSpaceRemoveUnused (default) kOptSpaceConvertIfSpaceSaved kOptSpaceConvertAll



Enumerators

[ASCabEnum](#)

[ASEnumExtensions](#)

[AVAppEnumActionHandlers](#)

[AVAppEnumAnnotHandlers](#)

[AVAppEnumDocs](#)

[AVAppEnumSystemFonts](#)

[AVAppEnumTools](#)

[AVAppEnumTransHandlers](#)

[AVConversionEnumFromPDFConverters](#)

[AVConversionEnumToPDFConverters](#)

[AVDocEnumSelection](#)

[AVDocSelectionEnumPageRanges](#)

[AVMenubarAcquireMenuByPredicate](#)

[AVMenubarAcquireMenuItemByPredicate](#)

[AVToolBarEnumButtons](#)

[CosDocEnumEOFs](#)

[CosDocEnumIndirect](#)

[CosObjEnum](#)

[PDCharProcEnum](#)

[PDDocEnumFonts](#)

[PDDocEnumLoadedFonts](#)

[PDDocEnumOCGs](#)

[PDEAttrEnumTable](#)

[PDEEnumElements](#)

[PDEClipFlattenedEnumElems](#)

[PDEnumDocs](#)



PDEnumSysFonts
PDXObjectDump
PDFontEnumCharProcs
PDFormEnumPaintProc
PDFormEnumResources
PDNameTreeEnum
PDNumTreeEnum
PDPageEnumContents
PDPageEnumResources
PDPathEnum
PDTextEnum
PDTextSelectEnumQuads
PDTextSelectEnumText
PDTextSelectEnumTextUCS
PDWordFinderEnumWords
PDXObjectEnumFilters
PDXObjectGetData



Font Subtypes

Methods: [PDFontGetSubtype](#)

Subtype	Description
CIDFontType0	Type 0 CID font
CIDFontType2	Type 2 CID font
Type0	Type 0 PostScript font
Type1	Type 1 PostScript font
Type3	Type 3 PostScript font
MMType1	Type 1 multiple master PostScript font
TrueType	TrueType font

Glyph Names of Word Separators

Methods: [PDDocCreateWordFinder](#)
[PDDocCreateWordFinderUCS](#)
[PDWordSplitString](#)

ampersand	comma	greater	parenleft	registered
asciicircum	copyright	guillemotleft	parenright	section
asciitilde	currency	guillemotright	percent	semicolon
asterisk	dagger	guilsinglleft	period	slash
at	daggerdbl	guilsinglright	periodcentered	space
backslash	degree	hyphen	perthousand	sterling
bar	divide	less	plus	threequarters
braceleft	dollar	logicalnot	plusminus	threesuperior
braceright	ellipsis	multiply	question	tilde
bracketleft	emdash	numbersign	questiondown	trademark
bracketright	endash	onehalf	quotedbl	twosuperior
brokenbar	equal	onequarter	quotedblbase	underscore
bullet	exclam	onesuperior	quotedblleft	yen
cent	exclamdown	ordfeminine	quotedblright	
circumflex	florin	ordmasculine	quoteleft	
colon	fraction	paragraph	quotesinglbase	



Key Codes

Header file: `ASKey.h`

Key Code—All platforms	Value
<code>ASKEY_ARROW_D</code>	31
<code>ASKEY_ARROW_L</code>	28
<code>ASKEY_ARROW_R</code>	29
<code>ASKEY_ARROW_U</code>	30
<code>ASKEY_ESCAPE</code>	27
<code>ASKEY_HELP</code>	5
<code>ASKEY_PAGE_D</code>	12
<code>ASKEY_PAGE_U</code>	11
<code>ASKEY_SPACE</code>	32
<code>ASKEY_TAB</code>	9
Key Code—Windows	Value
<code>ASKEY_DEL</code>	127
<code>ASKEY_END</code>	1
<code>ASKEY_ENTER</code>	13
<code>ASKEY_HOME</code>	4
<code>ASKEY_MENU</code>	2
Key Code—Macintosh	Value
<code>ASKEY_CLEAR</code>	27
<code>ASKEY_CR</code>	13
<code>ASKEY_DEL</code>	8
<code>ASKEY_END</code>	4



Key Code—Macintosh	Value
ASKEY_ENTER	3
ASKEY_HOME	1
Key Code—UNIX	Value
ASKEY_CLEAR	27
ASKEY_CR	13
ASKEY_DEL	8
ASKEY_END	4
ASKEY_ENTER	10
ASKEY_HOME	1



Language Codes

The following codes represent the supported languages in the supported formats (see [AVAppLanguageFormat](#)).

Methods: [AVAppGetLanguage](#)
[AVAppGetLanguageWithParams](#)

Language	LCID	ISO4Char	RFC1766
Chinese—Simplified	CHS	zh-cn	zh-cn
Chinese—Traditional	CHT	zh-tw	zh-tw
Danish	DAN	da-dk	da
Dutch	NLD	nl-nl	nl
English	ENU	en-us	en
Finnish	SUO	fi-fi	fi
French	FRA	fr-fr	fr
German	DEU	de-de	de
Italian	ITA	it-it	it
Japanese	JPN	ja-jp	ja
Norwegian	NOR	no-no	no
Portuguese—Brazil	PTB	pt-br	pt-br
Spanish	ESP	es-sp	es
Swedish	SVE	sv-se	sv



Menu and Menu Item Names

The following are the language-independent names of built-in menus and menu items , as returned by the JavaScript command `app.listMenuItems()`.

NOTE: Many of these are not available in Adobe Reader.

Methods: [AVMenuBarAcquireMenuByName](#)
[AVMenuBarAcquireMenuItemByName](#)

File Menu

```
[cName:File, oChildren:[
    [cName:NewDocument, oChildren:
        [ [cName:NewDocFromFile],
          [cName:NewDocFromMultiple],
          [cName:Scan],
          [cName:Web2PDF:OpnURL],
          [cName:ImageConversion:Clipboard]]],
    [cName:Open],
    [cName:&My Bookshelf...],
    [cName:AcroSendMail:SendMail],
    [cName:endSendGroup],
    [cName:Close],
    [cName:Save],
    [cName:SaveAs],
    [cName:DIGSIG:SaveAndAuthenticate],
    [cName:Revert],
    [cName:endSaveGroup],
    [cName:ReduceFileSize],
    [cName:endOptimizeGroup],
    [cName:SendForReview],
    [cName:Annots:BrowserBasedReview],
    [cName:Annots:GoBackOnline],
    [cName:SendCommentsToAuthor],
    [cName:Annots:ExportToWordFileMenu],
    [cName:endFileCommentGroup],
    [cName:GeneralInfo],
    [cName:endDocInfoGroup],
    [cName:PageSetup],
    [cName:Print],
    [cName:Annots:PrintWithComments],
    [cName:EFI:PrintMe],
    [cName:endPrintGroup],
    [cName:RecentFile1],
    [cName:RecentFile2],
    [cName:RecentFile3],
    [cName:RecentFile4],
    [cName:RecentFile5],
    [cName:endRecentFileGroup],
    [cName:Quit]]],
```

Edit Menu

```
[cName:Edit, oChildren:
  [ [cName:Undo] ,
    [cName:Redo] ,
    [cName:endUndoGroup] ,
    [cName:Cut] ,
    [cName:Copy] ,
    [cName:Paste] ,
    [cName:Clear] ,
    [cName:CopyFileToClipboard] ,
    [cName:endEditGroup] ,
    [cName:SelectAll] ,
    [cName:DeselectAll] ,
    [cName:endSelectGroup] ,
    [cName:Spelling:Spelling, oChildren:
      [ [cName:Spelling:Check Spelling] ,
        [cName:Spelling:Edit Dictionary]]] ,
    [cName:endSpellingGroup] ,
    [cName:NewBookmark] ,
    [cName:endEditNewGroup] ,
    [cName:LookUpWord] ,
    [cName:endLookUpGroup] ,
    [cName:Find] ,
    [cName:SearchSubmenu, oChildren:
      [ [cName:FindAgainDoc] ,
        [cName:FindAgain] ,
        [cName:FindPrevious] ,
        [cName:FindPreviousDoc]]] ,
    [cName:endFindGroup] ,
    [cName:GeneralPrefs]]]
```

View Menu

```
[cName:View, oChildren:
  [[cName:Navigation, oChildren:
    [ [cName:ShowHideArticles],
      [cName:ShowHideBookmarks],
      [cName:ShowHideAnnotManager],
      [cName:ShowHideContentPanel],
      [cName:ShowHideDestinations],
      [cName:ShowHideFields],
      [cName:ShowHideInfo],
      [cName:ShowHideOptCont],
      [cName:ShowHideThumbnails],
      [cName:ShowHideSignatures],
      [cName:ShowHideStructurePanel],
    [cName:endShowHideWindowsGroup2],
    [cName:ShowHideNavigationPanel],
    [cName:DockAllPanels],
    [cName:ResetPanels]]],
  [cName:ShowHideTaskPane],
  [cName:TaskButtons, oChildren:
    [ [cName:ShowHideButtonEBookTask],
      [cName:ShowHideButtonNewDocumentTask],
      [cName:ShowHideButtonCommentTask],
      [cName:ShowHideButtonSecureTask],
      [cName:ShowHideButtonSignTask],
      [cName:ShowHideButtonAdvEditingTask],
      [cName:endTaskButtonsGroup],
      [cName:ShowAllTaskButtons]]],
  [cName:Toolbars, oChildren:
    [ [cName:ShowHideAdvCommenting],
      [cName:ShowHideEditing],
      [cName:ShowHideBasicTools],
      [cName:ShowHideCommenting],
      [cName:ShowHideUndoRedo],
      [cName:ShowHideFile],
      [cName:ShowHideHowTo],
      [cName:ShowHideMeasuring],
      [cName:ShowHideNavigation],
      [cName:ShowHideRotate],
      [cName:ShowHideTasks],
      [cName:ShowHideViewing],
      [cName:endToolbarsGroup],
      [cName:PropertyToolbar],
      [cName:endPropertyBarGroup],
      [cName:ToolButtonLabels],
      [cName:ShowHideToolBar],
      [cName:DockToolBars],
      [cName:ResetToolBarLocations],
      [cName:LockToolBars]]],
  [cName:ShowHideMenuBar],
```

```

[cName:endViewPanelsGroup],
[cName:ZoomTo],
[cName:ActualSize],
[cName:FitPage],
[cName:FitWidth],
[cName:FitVisible],
[cName:Reflow],
[cName:endReflowGroup],
[cName:AutoScroll],
[cName:ReadAloud, oChildren:
    [ [cName:ADBE:ReadAloudStart],
      [cName:ADBE:ReadAloudToEnd],
      [cName:ADBE:ReadAloudPauseResume],
      [cName:ADBE:ReadAloudStop]]],
[cName:endDocAccessGroup],
[cName:GoTo, oChildren:
    [ [cName:FirstPage],
      [cName:PrevPage],
      [cName:NextPage],
      [cName:LastPage],
      [cName:GoToPage],
      [cName:endPageNavGroup],
      [cName:GoBack],
      [cName:GoForward],
      [cName:endGoBackGroup],
      [cName:GoBackDoc],
      [cName:GoForwardDoc]]],
[cName:endGoToSubGroup],
[cName:PageLayout, oChildren:
    [ [cName:SinglePage],
      [cName:OneColumn],
      [cName:TwoPages],
      [cName:TwoColumns]]],
[cName:endPageLayoutSubGroup],
[cName:Rotate, oChildren:
    [ [cName:RotateCW],
      [cName:RotateCCW]]],
[cName:endRotateSubGroup],
[cName:ShowGrid],
[cName:SnapToGrid],
[cName:ShowRulers],
[cName:ShowGuides],
[cName:endGuideGroup],
[cName:Annots:Comments, oChildren:
    [ [cName:Annots:ShowButton:CommentPanel],
      [cName:Annots:Separator],
      [cName:Annots:Show All Comments],
      [cName:Annots:Show Types, oChildren:
          [ [cName:Annots:Show all Types],
            [cName:Annots:Comments],
            [cName:Annots:Drawing],

```

```
[cName:Annots:Markup] ,  
[cName:Annots:Stamp] ,  
[cName:Annots:Attachments]]] ,  
[cName:Annots:Show Reviewers] ,  
[cName:Annots:Show Status] ,  
[cName:Annots:Show Marked, oChildren:  
  [cName:Annots:Show all Marked] ,  
  [cName:Annots:Marked] ,  
  [cName:Annots:Unmarked]]] ,  
[cName:Annots:Separator] ,  
[cName:Annots:OpenPopups] ,  
[cName:Annots:ClosePopups] ,  
[cName:Annots:Separator] ,  
[cName:Annots:ConnectorLines] ,  
[cName:Annots:AlignPopups]]] ,  
[cName:Annots:CommentsSummaryPane] ,  
[cName:Annots:ReviewTracker]]]
```

Document Menu

```
[cName:Document, oChildren:
  [ [cName:Pages, oChildren:
    [ [cName:InsertPages],
      [cName:ExtractPages],
      [cName:ReplacePages],
      [cName>DeletePages],
      [cName:CropPages],
      [cName:RotatePages],
      [cName:SetTransitions]]],
    [cName:COMP:AddHeadFoot],
    [cName:COMP:AddBack],
    [cName:endPagesGroup],
    [cName:Annots:AddComment],
    [cName:Annots:Separator],
    [cName:DIGSIG:CompareDocuments],
    [cName:Annots:SummarizeComments],
    [cName:endSummarizeCommentsItem],
    [cName:ImportNotes],
    [cName:ExportNotes],
    [cName:Annots:ExportToWordDocMenu],
    [cName:endCommentExchangeGroup],
    [cName:EmbeddedFiles],
    [cName:endDocAttachmentsGroup],
    [cName:DocSecurity, oChildren:
      [ [cName:SecureApplyPwdSecurity],
        [cName:SecureApplyCertSecurity],
        [cName:ChangeSecurity]]],
    [cName:DIGSIG:DigitalSignatures, oChildren:
      [ [cName:DIGSIG:PlaceSigPullRight],
        [cName:Separator],
        [cName:DIGSIG:ValidSigPullRight],
        [cName:DIGSIG:ValidateAll],
        [cName:Separator],
        [cName:DIGSIG:SignUITaskCreateField],
        [cName:DIGSIG>DeletePullRight],
        [cName:Separator],
        [cName:DIGSIG:RollbackPullRight],
        [cName:DIGSIG:DifferencePullRight],
        [cName:Separator],
        [cName:DIGSIG:PropertiesPullRight]]],
    [cName:endDocSignaturesGroup],
    [cName:PaperCapture, oChildren:
      [ [cName:Cpt:CapturePages],
        [cName:endPaperCaptureGroup],
        [cName:TouchUp:FindCaptureSuspects],
        [cName:TouchUp:ShowCaptureSuspects]]],
    [cName:endDocMiscGroup],
    [cName:CALS:Preflight]]]
```



Tools Menu

```
[cName:Tools, oChildren:
  [cName:Commenting, oChildren:
    [cName:Annots:Tool:TextMenuItem],
    [cName:Annots:Tool:TextEditsMenuItem, oChildren:
      [cName:Annots:TextEditMenu:TextEdits],
      [cName:Annots:Separator],
      [cName:Annots:TextEditMenu:Caret],
      [cName:Annots:TextEditMenu:Replace],
      [cName:Annots:TextEditMenu:HighlightSel],
      [cName:Annots:TextEditMenu:HighlightNote],
      [cName:Annots:TextEditMenu>DeleteSel],
      [cName:Annots:TextEditMenu:UnderlineSel],
      [cName:Annots:Separator],
      [cName:Annots:TextEditMenu:HowToTextEdits]]],
    [cName:Annots:Tools:Stamps, oChildren:
      [cName:dummyItem]]],
    [cName:Commenting:Highlight, oChildren:
      [cName:Annots:Tool:HighlightMenuItem],
      [cName:Annots:Tool:StrikeOutMenuItem],
      [cName:Annots:Tool:UnderlineMenuItem],
      [cName:Commenting:HighlightSeparator],
      [cName:ShowHideCommenting:Highlight]]],
    [cName:SeparatorToolsItem],
    [cName:CommentingSeparator],
    [cName:ShowHideCommenting]]],
  [cName:AdvCommenting, oChildren:
    [cName:Commenting:Ink, oChildren:
      [cName:Annots:Tool:SquareMenuItem],
      [cName:Annots:Tool:CircleMenuItem],
      [cName:Annots:Tool:LineArrowMenuItem],
      [cName:Annots:Tool:LineMenuItem],
      [cName:Annots:Tool:CloudyPolygonMenuItem],
      [cName:Annots:Tool:PolygonMenuItem],
      [cName:Annots:Tool:PolyLineMenuItem],
      [cName:Commenting:InkSeparator],
      [cName:ShowHideCommenting:Ink]]],
    [cName:Annots:Tool:FreeTextMenuItem],
    [cName:Annots:Tool:InkMenuItem],
    [cName:Annots:Tool:EraseInkMenuItem],
    [cName:Commenting:FileAttachment, oChildren:
      [cName:Annots:Tool:FileAttachmentMenuItem],
      [cName:Annots:Tool:SoundMenuItem],
      [cName:Annots:Tool:StampFromClipboardMenuItem],
      [cName:Commenting:FileAttachmentSeparator],
      [cName:ShowHideCommenting:FileAttachment]]],
    [cName:AdvCommentingSeparator],
    [cName:ShowHideAdvCommenting]]]
  [cName:BasicTools, oChildren:
    [cName:HandMenuItem],
```



```

[cName:Selection, oChildren:
    [ [cName:SelectMenuItem],
      [cName:TablePickerSelMenuItem],
      [cName:SelectImageMenuItem],
      [cName:SelectionSeparator],
      [cName:ShowHideSelection]]],
[cName:SelectGraphicsMenuItem],
[cName:endNavToolsGroupToolsItem],
[cName:ShowHideBasicTools]]]
[cName:Viewing, oChildren:
    [ [cName:ZoomIn],
      [cName:ZoomOut],
      [cName:ZoomDrag],
      [cName:Loupe],
      [cName:PanZoom],
      [cName:endZoomNavigationGroup],
      [cName:ViewingToolbar]]],
[cName:Editing, oChildren:[
    [cName:AuthoringMenuItem],
    [cName:ThreadMenuItem],
    [cName:CropMenuItem],
    [cName:LinkMenuItem],
    [cName:AcroForm, oChildren:[
        [cName:ButtonMenuItem],
        [cName:CheckBoxMenuItem],
        [cName:ComboBoxMenuItem],
        [cName:ListBoxMenuItem],
        [cName:RadioButtonMenuItem],
        [cName:TxMenuItem],
        [cName:SigMenuItem],
        [cName:AcroFormSeparator],
        [cName:ShowHideAcroForm]]],
    [cName:AddMovieMenuItem],
    [cName:AddSoundMenuItem],
    [cName:TouchUpTextToolMenuItem],
    [cName:TouchUpObjectToolMenuItem],
    [cName:endToolsGroupToolsItem],
    [cName:ShowHideEditing]]],
[cName:Measuring, oChildren:
    [ [cName:MeasureLineMenuItem],
      [cName:MeasurePolylineMenuItem],
      [cName:MeasurePolygonMenuItem],
      [cName:endMeasuringGroupToolsItem],
      [cName:ShowHideMeasuring]]]]]

```

Advanced Menu

```
[cName:Advanced, oChildren:
  [cName:Accessibility, oChildren:
    [cName:ADBE:QuickAccCheck],
    [cName:QuickCheck],
    [cName:AccCheck:DoCheck],
    [cName:FullCheck],
    [cName:Adobe:MakeAccessible]]],
  [cName:Distiller],
  [cName:BatchEdit],
  [cName:ADBE:Catalog],
  [cName:Metadata],
  [cName:My eBooks, oChildren:
    [cName:Adobe eBook Central],
    [cName:Adobe eBook Activator]]],
  [cName:ImageConversion:Export],
  [cName:AcroForm:Advanced, oChildren:
    [cName:AcroForm:ImportFDF],
    [cName:AcroForm:ExportFDF],
    [cName:AcroForm:Fields, oChildren:
      [cName:AcroForm:CreateTable],
      [cName:AcroForm:Duplicate]]],
    [cName:AcroForm:Template],
    [cName:AcroForm:Separator],
    [cName:AcroForm:CalcOrder]]],
  [cName:EScript:Advanced, oChildren:
    [cName:EScript:JSDebugger],
    [cName:EScript:JSDebuggerStart],
    [cName:EScript:JSEditAll],
    [cName:EScript:JSDocScripts],
    [cName:EScript:JSDocActions]]],
  [cName:Weblink:Tools, oChildren:
    [cName:Weblink:CreateURLs],
    [cName:Weblink:RemoveURLs]]],
  [cName:Security Methods, oChildren:
    [cName:PUBSEC:AddressBook],
    [cName:ppklite:Main, oChildren:
      [cName:ppklite:Login],
      [cName:ppklite:Logout],
      [cName:ppklite:UserSettings]]],
    [cName:PPKMS:CredMgr],
    [cName:PPKMS:ManageDirectories]]],
  [cName:PDFOptimizer],
  [cName:Web2PDF:SpdrSubMnuItm, oChildren:
    [cName:Web2PDF:AppURL],
    [cName:Web2PDF:AppLinks],
    [cName:Web2PDF:ViewLinks],
    [cName:Web2PDF:Update],
    [cName:Web2PDF:SepAfterUpdt],
    [cName:Web2PDF:PgInfo],
```

```

        [cName:Web2PDF:SepPgInfo] ,
        [cName:Web2PDF:OpenInBrowser] ,
        [cName:Web2PDF:FrontStsDlgs]]] ,
[cName:endToolsSubGroup1] ,
[cName:LocalFonts] ,
[cName:endLocalFontsGroup] ,
[cName:ProofSetup, oChildren:
    [ [cName:ProofCustom] ,
      [cName:endProofCustomGroup] ,
      [cName:ProofInkBlack] ,
      [cName:ProofPaperWhite]]] ,
[cName:ProofColors] ,
[cName:Overprint] ,
[cName:SEPSV:SepItem] ,
[cName:ADBE:FlatView]]]

```

Window Menu

```

[cName:Window, oChildren:
    [ [cName:Cascade] ,
      [cName:Tile, oChildren:
          [ [cName:TileHorizontal] ,
            [cName:TileVertical]]] ,
      [cName:CloseAll] ,
      [cName:endWindowLayoutGroup] ,
      [cName:SplitWindow] ,
      [cName:endSplitGroup] ,
      [cName:ShowHideClipboard] ,
      [cName:endShowHideGroup] ,
      [cName:FullScreen]]]

```



Help Menu

```
[cName:Help, oChildren:
  [ [cName:HelpHowTo, oChildren:
    [ [cName:HelpHowToTaskHome],
      [cName:endHowToWindowMI],
      [cName:NewDocumentTask],
      [cName:CommentTask],
      [cName:SecureTask],
      [cName:SignTask],
      [cName:AdvEditingTask],
      [cName:PrepOutputTask],
      [cName:HowToTask],
      [cName:endHelpHowToTasks],
      [cName:HelpHowToTaskShowAtStartup]]],
    [cName:endHowToGroup],
    [cName:HelpUserGuide],
    [cName:endGuideGroup],
    [cName:About],
    [cName>AboutAdobeExtensions],
    [cName>AboutExtensions, oChildren:
      [ [cName:EFI:PrintMeHelp]]],
    [cName:endAboutPlugInGroup],
    [cName:SystemInformation],
    [cName:OnlineSupport],
    [cName:Updates],
    [cName:RegisterProduct],
    [cName:AdobeOnline],
    [cName:DetectAndRepair]]]
```

Replaceable Methods

These methods are replaceable in Adobe Reader, except as noted.

[AVAlert](#)

[AVAppCanQuit](#)

[AVAppHandleAppleEvent](#)

[AVDocClose](#)

[AVDocDoPrint](#) (not replaceable in Adobe Reader)

[AVDocDoSave](#) (not replaceable in Adobe Reader)

[AVDocDoSaveAs](#) (not replaceable in Adobe Reader)

[AVDocDoSaveAsWithParams](#) (not replaceable in Adobe Reader)

[AVDocOpenFromASFileWithParams](#)

[AVDocPrintPages](#)

[AVDocPrintPagesWithParams](#)

[AVPageViewGetNextView](#)

[PDDocSave](#) (not replaceable in Adobe Reader)

[PDDocSaveWithParams](#) (not replaceable in Adobe Reader)

[PDImageSelectAlternate](#)

Selection Types

Selection types that can be specified when calling [AVDocSetSelection](#). The “Details” column specifies what should be used for the **data** parameter.

Selection Type	Description	Data Type	Details
Text	Text in the document	PDTextSelect	Use PDDocCreateTextSelect to create the selection.
Bitmap	Graphics	AVGrafSelect	Use AVGrafSelectCreate to create the selection.
Annotation	Annotation (text annotation, link, and so on)		Allocate memory using ASmallloc (sizeof (PDAnnot)) . Copy the desired PDAnnot into the allocated memory and pass a pointer to it. The annotation selection server assumes responsibility for freeing the memory.
Thumbnail	Thumbnail image	ASInt32	Pass the page number (the first page in a document is page 0).

NOTE: To get information about a selected annotation, use [AVPageViewGetFocusAnnot](#).

Toolbar and Toolbar Button Names

Methods: [AVAppGetToolBarByName](#)
[AVToolBarGetButtonByName](#)

The following are the language-independent names of the built-in toolbars and toolbar buttons, as returned by the JavaScript command **app.listToolBarButtons()**.

```
[cName:JSToolBar]

[cName:AdvCommenting, oChildren:
  [ [cName:Annots:Tool:Square, oChildren:
    [ [cName:Annots:Tool:Square],
      [cName:Annots:Tool:Circle],
      [cName:Annots:Tool:LineArrow],
      [cName:Annots:Tool:Line],
      [cName:Annots:Tool:CloudyPolygon],
      [cName:Annots:Tool:Polygon],
      [cName:Annots:Tool:PolyLine]]],
    [cName:Annots:Tool:FreeText],
  [cName:Annots:Tool:Ink, oChildren:
    [ [cName:Annots:Tool:Ink],
      [cName:Annots:Tool:EraseInk]]],
  [cName:Annots:Tool:FileAttachment, oChildren:
    [ [cName:Annots:Tool:FileAttachment],
      [cName:Annots:Tool:Sound],
      [cName:Annots:Tool:StampFromClipboard]]]]]

[cName:Commenting, oChildren:
  [ [cName:Annots:Tool:Text],
    [cName:Annots:Tool:TextEdits],
    [cName:Annots:Tool:Stamp],
    [cName:Annots:Tool:Highlight, oChildren:
      [ [cName:Annots:Tool:Highlight],
        [cName:Annots:Tool:StrikeOut],
        [cName:Annots:Tool:Underline]]],
    [cName:Separator],
    [cName:Annots:Tool:Filter]]]

[cName:Measuring, oChildren:
  [ [cName:MeasureLine],
    [cName:MeasurePolyline],
    [cName:MeasurePolygon],
    [cName:endMeasuringGroup]]]

[cName:Editing, oChildren:
  [ [cName:Authoring],
    [cName:Thread],
    [cName:Crop],
    [cName:Link],
    [cName:Button, oChildren:
```

```

        [ [cName:Button],
          [cName:CheckBox],
          [cName:ComboBox],
          [cName:ListBox],
          [cName:RadioButton],
          [cName:Tx],
          [cName:Sig]]],
[cName:AddMovie, oChildren:
  [ [cName:AddMovie],
    [cName:AddSound]]],
[cName:TouchUpTextTool, oChildren:
  [ [cName:TouchUpTextTool],
    [cName:TouchUpObjectTool]]]
[cName:endToolsGroup]]]

[cName:Properties, oChildren:
  [ [cName:MainView],
    [cName:endPropertiesGroup]]]

[cName:Navigation, oChildren:
  [ [cName:FirstPage],
    [cName:PreviousPage],
    [cName:NextPage],
    [cName:LastPage],
    [cName:endPageNavGroup],
    [cName:GoBack],
    [cName:GoForward],
    [cName:endPageStackGroup]]]

[cName:HowTo, oChildren:
  [ [cName:HowTo]]]

[cName:Rotate, oChildren:
  [ [cName:RotateCW],
    [cName:RotateCCW],
    [cName:endRotateViewGroup]]]

[cName:Viewing, oChildren:
  [ [cName:ZoomIn, oChildren:
    [ [cName:ZoomIn],
      [cName:ZoomOut],
      [cName:ZoomDrag],
      [cName:Loupe]]],
    [cName:Zoom100],
    [cName:FitPage],
    [cName:FitVisible],
    [cName:ZoomViewOut],
    [cName:ZoomTo],
    [cName:ZoomViewIn]]]

[cName:BasicTools, oChildren:

```



```
[ [cName:Hand] ,
[cName:Select, oChildren:
    [ [cName:Select] ,
      [cName:TablePickerSel] ,
      [cName:SelectImage]]] ,
[cName:SelectGraphics] ,
[cName:endNavToolsGroup]]]

[cName:Tasks, oChildren:
    [ [cName:NewDocumentTask] ,
      [cName:CommentTask] ,
      [cName:SecureTask] ,
      [cName:SignTask] ,
      [cName:AdvEditingTask]]]

[cName:UndoRedo, oChildren:
    [ [cName:Spelling:Tool:SpellTool] ,
      [cName:Undo] ,
      [cName:Redo] ,
      [cName:Copy] ,
      [cName:endUndoRedoGroup]]]

[cName:File, oChildren:
    [ [cName:Open] ,
      [cName:ADBE:SPDR:OpStatTlButton] ,
      [cName:Save] ,
      [cName:Print] ,
      [cName:AcroSendMail:SendMail] ,
      [cName:FindDialog] ,
      [cName:endDialogGroup]]]
```

Tool Names

Methods: [AVAppGetToolByName](#)

Tool Name	UI Label
Movie	"Movie Tool (M)"
Link	"Link Tool (L)"
Thread	"Article Tool (A)"
Crop	"Crop Tool (C)"
Widget	"Form Tool (F)"
Touch-Up Text Selection	"TouchUp Text Tool (T)"
Object Selection Type	"TouchUp Object Tool (T)" / "TouchUp Order Tool (T)"
Hand	"Hand Tool (H)"
Zoom	"Zoom In Tool (Z)" / "Zoom Out Tool (Z)"
Select	"Text Select Tool (V)" / "Column Select Tool (V)"
BCLC:Table/Formatted_TextZoneTool	"Table/Formatted Text Select Tool (V)"
SelectGraphics	"Graphics Select Tool (G)"
Text	"Note Tool (S)"
FreeText	"FreeText Tool (S)"
Sound	"Sound Attachment Tool (S)"
Stamp	"Stamp Tool (S)"
FileAttachment	"File Attachment Tool (S)"
Ink	"Pencil Tool (N)"
Square	"Square Tool (N)"
Circle	"Circle Tool (N)"
Line	"Line Tool (N)"
Highlight	"Highlight Tool (U)"
StrikeOut	"Strikeout Tool (U)"
Underline	"Underline Tool (U)"
Squiggly	Not exposed through UI.
DigSigTool	"Digital Signature Tool (D)"

Type/Creator Codes

Type/Creator codes for [ASFileSysSetTypeAndCreator](#) method.

Creators		
kAcrobatCreatorCode	ASFourCharCode ('CARO')	Acrobat Creator Code
kPhotoshopCreatorCode	ASFourCharCode ('8BIM')	Photoshop Creator Code
kImageReadyCreatorCode	ASFourCharCode ('MeSa')	ImageReady Creator Code
kIllustratorCreatorCode	ASFourCharCode ('ART5')	Illustrator Creator Code

Acrobat Types	Creator=kAcrobatCreatorCode	
kPDFTypeCode	ASFourCharCode ('PDF ')	Portable Document Format
kFDFTypeCode	ASFourCharCode ('FDF ')	Forms Data Format
kXFDFTypeCode	ASFourCharCode ('XFDF')	XML Forms Data Format
kPrefsTypeCode	ASFourCharCode ('PREF')	Preferences File
kPDXTypeCode	ASFourCharCode ('PDX ')	Acrobat Catalog Index file
kRMFTypeCode	ASFourCharCode ('RMF ')	WebBuy Rights Management File
kAPFTypeCode	ASFourCharCode ('APF ')	Acrobat Profile Format (PPKLite)
kSequenceTypeCode	SFourCharCode ('SEQU')	Acrobat Sequence File
kDictionaryTypeCode	ASFourCharCode ('DICT')	Spelling Dictionary File
kWHATypeCode	ASFourCharCode ('WHA ')	Web-Hosted Apps File
kLocaleTypeCode	ASFourCharCode ('LANG')	Locale File
kPluginTypeCode	ASFourCharCode ('XTND')	client File

Acrobat Types		Creator=kAcrobatCreatorCode
kPluginNewTypeCode	ASFourCharCode ('XTNc')	client File (Preferred, supported by 5.0.5 and later.) Using this filetype allows a client developer to ship a Carbonized client that will not try to load and show an error when installed under Acrobat 4 or earlier.

Photoshop Types		Creator=kPhotoshopCreatorCode
kPSDTypeCode	ASFourCharCode ('8BIM')	Photoshop PSD File
kPICTTypeCode	ASFourCharCode ('PICT')	Mac PICT File
kTIFFTypeCode	ASFourCharCode ('TIFF')	TIFF File
kGIFTypeCode	ASFourCharCode ('GIFf')	GIF File
kJPEGTypeCode	ASFourCharCode ('JPEG')	JPEG File
kPNGTypeCode	ASFourCharCode ('PNGf')	PNG File

Illustrator Types		Creator=kIllustratorCreatorCode
KAITypeCode	ASFourCharCode ('TEXT')	Illustrator AI File
kEPSTypeCode	ASFourCharCode ('EPSF')	EPS File

Other Types/Creators		
kTextTypeCode	ASFourCharCode ('TEXT')	Text File
kTextCreatorCode	ASFourCharCode ('ttxt')	SimpleText
kQuickTimeTypeCode	ASFourCharCode ('MooV')	QuickTime File
kQuickTimeCreatorCode	ASFourCharCode ('TVOD')	QuickTime Player
kHTMLTypeCode	ASFourCharCode ('TEXT')	HTML File
kHTMLCreatorCode	ASFourCharCode ('MSIE')	Microsoft IE



View Destination Fit Types

XYZ	Destination specified as upper-left corner point and a zoom factor.
Fit	Fits the page into the window, corresponding to the Acrobat viewer's "FitPage" menu item.
FitH	Fits the widths of the page into the window, corresponding to the Acrobat viewer's "Fit Width" menu item.
FitV	Fits the height of the page into a window.
FitR	Fits the rectangle specified by its upper-left and lower-right corner points into the window.
FitB	Fits the rectangle containing all visible elements on the page (known as the bounding box) into the window, corresponds to the Acrobat viewer's "Fit Visible" menu item.
FitBH	Fits the width of the bounding box into the window.
FitBV	Fits the height of the bounding box into the window.



Notifications

AVAppDidInitialize

```
ACCB1 void ACCB2 AVAppDidInitialize (void* clientData);
```

Description

The Acrobat viewer has finished initializing and is about to enter its event loop.

Parameters

clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVAppWillQuit](#)

Methods

None



AVAppDidInitExtensions

```
ACCB1 void ACCB2 AVAppDidInitExtensions (void* clientData);
```

Description

The Acrobat viewer has finished initializing extensions.

Parameters

clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVAppWillTerminateExtensions](#)

Methods

None

AVAppFrontDocDidChange

```
ACCB1 void ACCB2 AVAppFrontDocDidChange (AVDoc doc,  
void* clientData);
```

Description

The front-most **AVDoc** has changed.

NOTE: This notification is not broadcast for external windows, such as OLE applications or PDF files being displayed in Netscape.

Parameters

doc	The document that was brought to the front. NULL if there is no front-most document (for example, the previous front-most document was just closed).
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocDidActivate](#)
[AVDocDidDeactivate](#)

Methods

[AVWindowBringToFront](#)
[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromFile](#)
[AVDocOpenFromASFileWithParams](#)
[AVDocOpenFromPDDoc](#)
[AVDocOpenFromPDDocWithParams](#)

AVAppOldPrefDidChange

```
ACCB1 void ACCB2 AVAppOldPrefDidChange  
(AVPrefsType preference, void* clientData);
```

Description

An old-style [AVPrefsType](#) was changed.

Parameters

preference	Preference type that was changed. You can call AVAppGetPreference to find out the new value.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVAppPrefDidChange](#)

Methods

[AVAppSetPreference](#)

AVAppPrefDidChange

```
ACCB1 void ACCB2 AVAppPrefDidChange (const char* section,
const char* key, void* clientData);
```

Description

A new-style preference changed.

NOTE: As of Acrobat 5.0, most of the methods used to set and get the new style preferences have not been exported to the public API.

Parameters

section	The section.
key	The key.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVAppOldPrefDidChange](#)

Methods

Numerous

AVAppSystemLogonSessionSwitched

```
ACCB1 void ACCB2 AVAppSystemLogonSessionSwitched  
(const char* switchState, void* clientData);
```

Description

Sent on a system running Windows XP when the user does a Switch User (Fast User Switching) operation to log on or switch to another user account. Typically used to release shared resources, such as fonts or multimedia ports.

Parameters

switchState	The switch state. ● WTS_CONSOLE_CONNECT when a switched out session is switched back in. ● WTS_CONSOLE_DISCONNECT when a session is switched out.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .



AVAppToolDidChange

```
ACCB1 void ACCB2 AVAppToolDidChange (AVDoc activeDoc,
AVTool prevTool, AVTool curTool, void* clientData);
```

Description

A document’s active tool changed.

Parameters

activeDoc	The document
prevTool	The previously active tool.
curTool	The currently active tool.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Methods

[AVDocGetActiveTool](#)
[AVDocSetActiveTool](#)

AVAppUsingPDDocForBatch

```
void AVAppUsingPDDocForBatch (AVBatchContext batchContext,
PDDoc pdDoc, ASBool isUsing, void* clientData);
```

Description

Used to let clients know when **pdDoc** is undergoing batch processing.

NOTE: This notification is set for *any* **pdDoc** that serves as input to batch processing, including **pdDocs** that are attached to **AVDocs** when batching is invoked on the currently open documents.

Parameters

batchContext	Placeholder. Reserved for future releases. Always NULL .
pdDoc	Document undergoing batch processing.
isUsing	When the document is about to be batch processed, the notification is sent with isUsing set to true . When batch processing has finished with the document, the notification is again sent but with isUsing set to false .
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

None

Methods

Numerous

AVAppWillCloseAllInternalDocs

```
ACCB1 void ACCB2 AVAppWillCloseAllInternalDocs  
(void* clientData);
```

Description

All **AVDocs** will be closed.

Parameters

clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[AVDocDidClose](#)
[AVDocWillClose](#)

Methods

None



AVAppWillQuit

```
ACCB1 void ACCB2 AVAppWillQuit (void* clientData);
```

Description

The Acrobat viewer is quitting. All documents have been closed. To access or enumerate documents when the application is quitting, replace the [AVAppCanQuit](#) method, access or enumerate documents in your replacement for that procedure, and return **true** to allow the Acrobat viewer to continue quitting.

Parameters

clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVAppDidInitialize](#)

Methods

None

AVAppWillTerminateExtensions

```
ACCB1 void ACCB2 AVAppWillTerminateExtensions  
(void* clientData);
```

Description

The Acrobat viewer will terminate extensions.

Parameters

clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVAppDidInitExtensions](#)
[AVAppWillQuit](#)

Methods

None

AVDocActivePageViewDidChange

```
ACCB1 void ACCB2 AVPageViewDidChange (AVDoc avDoc,  
AVPageView newPageView, AVPageView oldPageView,  
void* clientData);
```

Description

Called when the document's page view changes.

Parameters

avDoc	The AVDoc whose page view has changed.
newPageView	The new AVPageView .
oldPageView	The old AVPageView .
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVPageViewDocDidChange](#)
[AVPageViewWasCreated](#)
[AVPageViewWillDestroy](#)

Methods

[AVDocGetNthPageView](#)
[AVDocGetNumPageViews](#)
[AVDocGetPageView](#)

AVDocDidActivate

```
ACCB1 void ACCB2 AVDocDidActivate (AVDoc doc,  
void* clientData);
```

Description

An **AVDoc** has activated. At the time this notification is broadcast, it is possible that the window being activated has not yet been brought to the front. For this reason, the [AVAppFrontDocDidChange](#) notification is often more useful.

NOTE: [AVAppGetActiveDoc](#) will not necessarily return the AVDoc returned in this notification. For instance, if there is an AVDoc in an external window (such as a Web browser's) that becomes active, the AVDoc returned by this notification will not match what [AVAppGetActiveDoc](#) returns.

NOTE: This notification is not broadcast for external windows, such as OLE applications or PDF files being displayed in Netscape.

Parameters

doc	The document that was activated.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocDidDeactivate](#)
[AVAppFrontDocDidChange](#)

Methods

[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromFile](#)
[AVDocOpenFromASFileWithParams](#)
[AVDocOpenFromPDDoc](#)
[AVDocOpenFromPDDocWithParams](#)

AVDocDidAddToSelection

```
ACCB1 void ACCB2 AVDocDidAddToSelection (AVDoc doc,
ASAtom selType, void *selData, void *addData,
void* clientData);
```

Description

The document's selection has been added to or had something removed.

Parameters

doc	The document containing the selection.
selType	The ASAtom corresponding to the current selection type. See Selection Types for a list of selection types.
selData	Pointer to the current selection data <i>after</i> the selection has been added. The format and contents of selData depend on selType .
addData	Pointer to the added selection data. The format and contents of addData depend on selType .
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocDidClearSelection](#)
[AVDocDidSetSelection](#)
[AVDocDidRemoveFromSelection](#)
[AVDocWillClearSelection](#)

Methods

[AVDocClearSelection](#)
[AVDocDeleteSelection](#)
[AVDocSetSelection](#)

AVDocDidClearSelection

```
ACCB1 void ACCB2 AVDocDidClearSelection (AVDoc doc,  
void* clientData);
```

Description

A document's selection has been cleared.

Parameters

doc	The document whose selection was cleared.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocWillClearSelection](#)

Methods

[AVDocClearSelection](#)
[AVDocDeleteSelection](#)
[AVDocSetSelection](#)

AVDocDidClickName

```
ACCB1 void ACCB2 AVDocDidClickName (AVDoc doc, CosObj nameObj,  
void* clientData);
```

Description

Acrobat executed an action to go to a named destination.

Parameters

doc	The document containing the named destination.
nameObj	The Cos object corresponding to the named destination.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

None

Methods

None



AVDocDidClose

```
ACCB1 void ACCB2 AVDocDidClose (AVDoc doc, void* clientData);
```

Description

A document has been closed. Although an **AVDoc** is passed to the routine called by this notification, the document has already been closed but not freed. As a result, all the routine can really do is manipulate any private data in the underlying PDF file at the time this notification occurs.

Parameters

doc	The document that was closed.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocWillClose](#)

Methods

[AVDocClose](#)



AVDocDidDeactivate

```
ACCB1 void ACCB2 AVDocDidDeactivate (AVDoc doc,  
void* clientData);
```

Description

A document was deactivated.

NOTE: This notification is not broadcast for external windows, such as OLE applications or PDF files being displayed in Netscape.

Parameters

doc	The document that was deactivated.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocDidActivate](#)
[AVAppFrontDocDidChange](#)

Methods

[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromFile](#)
[AVDocOpenFromASFileWithParams](#)
[AVDocOpenFromPDDoc](#)
[AVDocOpenFromPDDocWithParams](#)

AVDocDidDeleteSelection

```
ACCB1 void ACCB2 AVDocDidDeleteSelection (AVDoc doc, ASAtom
selType, void* selData, void* clientData);
```

Description

Called when a user deletes the current selection.

Parameters

doc	The document containing the selection.
selType	The selection's type. See Selection Types for a list of the those available.
selData	Server-dependent selection data.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocDidAddToSelection](#)
[AVDocDidSetSelection](#)

Related Methods

[AVDocDeleteSelection](#)

AVDocDidOpen

```
ACCB1 void ACCB2 AVDocDidOpen (AVDoc doc, ASInt32 error,  
void* clientData);
```

Description

A document has been opened.

Calling [AVDocClose](#) within this notification is forbidden.

Parameters

doc	The document that was opened.
error	Error code. error is set to 0 if no errors occurred while opening the file. If an error occurred, error contains the error code.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocWillOpenFromFile](#)
[AVDocWillOpenFromPDDoc](#)

Methods

[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromFile](#)
[AVDocOpenFromASFileWithParams](#)

AVDocDidPerformAction

```
ACCB1 void ACCB2 AVDocDidPerformAction (AVDoc doc,  
PDAction action, ASInt32 err, void* clientData);
```

Description

An action was performed.

Parameters

doc	The document containing the action that was performed.
action	The action that was performed.
err	Error code. error is set to 0 if no errors occurred while performing the action. If an error occurred, error contains the error code.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocWillPerformAction](#)

Methods

[AVDocPerformAction](#)

The following methods broadcast this notification if the document has an open action:

[AVDocOpenFromASFileWithParamString](#)

[AVDocOpenFromFile](#)

[AVDocOpenFromASFileWithParams](#)

[AVDocOpenFromPDDoc](#)

[AVDocOpenFromPDDocWithParams](#)

AVDocDidRemoveFromSelection

```
ACCB1 void ACCB2 AVDocDidRemoveFromSelection (AVDoc doc,
ASAtom selType, void* selData, void* remData,
void* clientData);
```

Description

The document's selection has had something removed.

Parameters

doc	The document whose selection was removed.
selType	The ASAtom corresponding to the current selection type. See Selection Types for a list of selection types.
selData	Pointer to the current selection data <i>after</i> the selection has been deleted. The format and contents of selData depend on selType .
remData	The item removed from the selection. The format and contents of addData depend on selType .
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocDidSetSelection](#)
[AVDocDidAddToSelection](#)
[AVDocDidClearSelection](#)
[AVDocWillClearSelection](#)

Methods

[AVDocClearSelection](#)
[AVDocDeleteSelection](#)
[AVDocSetSelection](#)



AVDocDidPrint

```
ACCB1 void ACCB2 AVDocDidPrint (AVDoc doc, void* clientData);
```

Description

This notification is broadcast after printing ends.

Parameters

doc	The document that was printed.
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[AVDocWillPrint](#)

Methods

[AVDocPrintPages](#)

[AVDocPrintPagesWithParams](#)

AVDocDidSetSelection

```
ACCB1 void ACCB2 AVDocDidSetSelection (AVDoc doc,
ASAtom selType, void *selData, void* clientData);
```

Description

The document's selection has been set.

Parameters

doc	The document whose selection was set.
selType	The ASAtom corresponding to the current selection type. See Selection Types for a list of selection types.
selData	Pointer to the current selection data. The format and contents of selData depend on selType .
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocDidAddToSelection](#)
[AVDocDidClearSelection](#)
[AVDocWillClearSelection](#)

Methods

[AVDocClearSelection](#)
[AVDocDeleteSelection](#)
[AVDocSetSelection](#)



AVDocWantsToDie

```
ACCB1 void ACCB2 AVDocWantsToDie (AVDoc doc,  
void* clientData);
```

Description

An **AVDoc**'s file stream has been terminated by the [AVDocSetDead](#) method.

Parameters

doc	The AVDoc whose file stream has been terminated.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocDidClose](#)

Methods

[AVDocSetDead](#)

AVDocWillClearSelection

```
ACCB1 void ACCB2 AVDocWillClearSelection (AVDoc doc,
ASAtom selType, void *selData, void* clientData);
```

Description

A document's selection is about to be cleared.

Parameters

doc	The document whose selection will be cleared.
selType	The ASAtom corresponding to the current selection type.
selData	Pointer to the current selection data. The format and contents of selData depend on selType .
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocDidAddToSelection](#)
[AVDocDidClearSelection](#)
[AVDocDidSetSelection](#)

Methods

[AVDocClearSelection](#)
[AVDocDeleteSelection](#)
[AVDocSetSelection](#)



AVDocWillClose

```
ACCB1 void ACCB2 AVDocWillClose (AVDoc doc, void* clientData);
```

Description

An **AVDoc** will be closed. Neither this notification nor [AVDocDidClose](#) are broadcast if the user selects “Cancel” when prompted to save a modified document as it is closed.

NOTE: Reads made from **AVDocWillClose** on an in-browser document will fail if the data is not already downloaded.

Parameters

doc	The document that will be closed.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocDidClose](#)

Methods

[AVDocClose](#)

AVDocWillOpenFromFile

```
ACCB1 void ACCB2 AVDocWillOpenFromFile (ASPathName fileName,
ASFileSys fileSys, void* clientData);
```

Description

An **AVDoc** will be opened from a file.

Parameters

fileName	The ASPathName for the file that will be opened.
fileSys	The file system responsible for the file to open.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocDidOpen](#)

Methods

[AVDocOpenFromASFileWithParamString](#)
[AVDocOpenFromFile](#)
[AVDocOpenFromASFileWithParams](#)

AVDocWillOpenFromPDDoc

```
ACCB1 void ACCB2 AVDocWillOpenFromPDDoc (PDDoc pdDoc,  
void* clientData);
```

Description

An **AVDoc** will be opened from a PDF file.

Parameters

pdDoc	The PDDoc for the file that will be opened.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocDidOpen](#)

Methods

[AVDocOpenFromPDDoc](#)

[AVDocOpenFromPDDocWithParams](#)

AVDocWillPerformAction

```
ACCB1 void ACCB2 AVDocWillPerformAction (AVDoc doc,
PDAction action, void* clientData);
```

Description

An action is about to be performed.

Parameters

doc	The document containing the action that will be performed.
action	The action that will be performed.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocDidPerformAction](#)

Methods

[AVDocPerformAction](#)

The following methods broadcast this notification if the document has an open action:

[AVDocOpenFromASFileWithParamString](#)

[AVDocOpenFromFile](#)

[AVDocOpenFromASFileWithParams](#)

[AVDocOpenFromPDDoc](#)

[AVDocOpenFromPDDocWithParams](#)



AVDocWillPrint

```
ACCB1 void ACCB2 AVDocWillPrint (AVDoc doc, void* clientData);
```

Description

This notification is broadcast before a document is printed, before any marks are made on the first page.

Parameters

doc	The document that is about to be printed.
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[AVDocDidPrint](#)

Methods

[AVDocPrintPages](#)

[AVDocPrintPagesWithParams](#)

AVPageViewAnnotDidPerformOp

```
ACCB1 void ACCB2 AVPageViewAnnotDidPerformOp  
(AVPageView pageView, PDAnnot pdAnnot, AVAnnotOp annotOp,  
void* clientData);
```

Description

Goes out when an annotation has performed a focus operation—for example, **kAVAnnotAcceptFocus** or **kAVAnnotLostFocus**. No notification is issued for **kAVAnnotDefaultAction** or **kAVAnnotShowMenu**.

Parameters

pageView	The page view containing the annotation.
pdAnnot	The annotation that performed the operation.
annotOp	The operation.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

None

Methods

[AVPageViewFocusAnnotPerformOp](#)

AVPageViewDocDidChange

```
ACCB1 void ACCB2 AVPageViewDocDidChange (AVPageView pageView,  
AVDoc newDoc, AVDoc oldDoc, void* clientData);
```

Description

Sent when a page view becomes associated with an **AVDoc**. When a cross-document link is being performed, the same page view may be re-used with a different **AVDoc**. In this case this notification is sent twice, once when the old **AVDoc** is closed and the page view's **AVDoc** becomes **NULL**, and again when the new **AVDoc** is opened and associated with the page view.

Parameters

pageView	The AVPageView for which the document changed.
newDoc	The new document associated with the page view.
oldDoc	The document that was previously associated with the page view.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocActivePageViewDidChange](#)
[AVPageViewDidChange](#)
[AVPageViewWasCreated](#)
[AVPageViewWillDestroy](#)

Methods

[AVDocGetNthPageView](#)
[AVDocGetNumPageViews](#)
[AVDocGetPageView](#)
[AVPageViewGetAVDoc](#)

AVPageViewDidChange

```
ACCB1 void ACCB2 AVPageViewDidChange (AVPageView pageView,
ASInt16 how, void* clientData);
```

Description

The page view has changed. Zero or more of the following events has occurred:

- The page number has changed.
- The zoom factor has changed.
- The window has been resized.
- The page has been scrolled.

NOTE: If continuous scrolling is turned on (available in Acrobat 3.0 or later) and more than one page is displayed in the **AVPageView**, alternating mouse clicks in the different pages displayed does not constitute a change to the **AVPageView**.

Parameters

pageView	The AVPageView that has changed.
how	Specifies how the page view did change. how is an OR of zero or more of the following (see AVExpT.h): PAGEVIEW_UPDATE_SCROLL — The view has been scrolled. PAGEVIEW_UPDATE_PAGENUM — The page number has changed. PAGEVIEW_UPDATE_PAGESIZE — A new view has been created. PAGEVIEW_UPDATE_ZOOM — The zoom has been changed.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocActivePageViewDidChange](#)
[AVPageViewDidDraw](#)
[AVPageViewWasCreated](#)
[AVPageViewWillDestroy](#)

Methods

[AVPageViewZoomTo](#)
[AVPageViewScrollTo](#)
[AVPageViewScrollToRect](#)
[AVPageViewGoTo](#)
[AVPageViewReadPageDown](#)
[AVPageViewReadPageUp](#)



`AVPageViewGoBack`
`AVPageViewGoForward`
`AVPageViewUseDestInfo`
`AVPageViewUseThisDestination`



AVPageViewDidDraw

```
ACCB1 void ACCB2 AVPageViewDidDraw (AVPageView pageView,  
void* clientData);
```

Description

Redrawing occurred in the page view section of the window.

Parameters

pageView	The AVPageView in which drawing occurred.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVPageViewDidChange](#)
[AVPageViewWillDraw](#)

Methods

[AVPageViewDrawNow](#)

AVPageViewWasCreated

```
ACCB1 void ACCB2 AVPageViewWasCreated (AVPageView pageView,  
void* clientData);
```

Description

Sent when a page view is created.

Parameters

pageView	The AVPageView that was created.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocActivePageViewDidChange](#)
[AVPageViewDocDidChange](#)
[AVPageViewDidChange](#)
[AVPageViewWillDestroy](#)

Methods

[AVDocGetNthPageView](#)
[AVDocGetNumPageViews](#)
[AVDocGetPageView](#)
[AVPageViewGetAVDoc](#)

AVPageViewWillDestroy

```
ACCB1 void ACCB2 AVPageViewWillDestroy (AVPageView pageView,  
void* clientData);
```

Description

Sent before a page view is destroyed.

Parameters

pageView	The AVPageView that will be destroyed.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVDocActivePageViewDidChange](#)
[AVPageViewDocDidChange](#)
[AVPageViewDidChange](#)
[AVPageViewWasCreated](#)

Methods

[AVDocGetNthPageView](#)
[AVDocGetNumPageViews](#)
[AVDocGetPageView](#)
[AVPageViewGetAVDoc](#)

AVPageViewWillDraw

```
ACCB1 void ACCB2 AVPageViewWillDraw (AVPageView pageView,  
void* clientData);
```

Description

Redrawing will occur in the page view section of the window.

Parameters

pageView	The AVPageView in which drawing will occur.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[AVPageViewDidChange](#)
[AVPageViewDidDraw](#)

Methods

[AVPageViewDrawNow](#)

PagePDEContentDidChange

```
ACCB1 void ACCB2 PagePDEContentDidChange (PDPPage page,  
PDEContent pagesPDEContent) ;
```

Description

The **PDEContent** for a page has changed.

Parameters

page	The page whose PDEContent changed.
pagesPDEContent	The PDEContent that changed.

Related Notifications

[PagePDEContentNotCached](#)

Methods

[PDPPagePDEContentWasChanged](#)

PagePDEContentNotCached

```
ACCB1 void ACCB2 PagePDEContentNotCached ( PDPPage page,
PDEContent pagesPDEContent ) ;
```

Description

The **PDEContent** for a page is no longer valid.

This notification is also sent when others change (or delete) a **PDPPage**'s contents without using PDFEdit methods. For instance, rotating or deleting a page in the viewer results in this notification being sent.

You should free up any tag data you might have attached to the **PDEContent**.

PDFEdit registers for almost a half dozen different notifications for the different ways Acrobat can alter page contents—you may need only this notification.

Parameters

page	The page whose PDEContent is no longer valid.
pagesPDEContent	The PDEContent that is no longer valid.

Related Notifications

[PagePDEContentDidChange](#)

Methods

[PDDocDeletePages](#)
[PDPPageAddCosContents](#)
[PDPPageAddCosResource](#)
[PDPPageRemoveCosContents](#)
[PDPPageRemoveCosResource](#)
[PDPPageSetRotate](#)

PDAnnotDidChange

```
ACCB1 void ACCB2 PDAnnotDidChange (PDAnnot annot, ASAtom key,
ASInt32 error, void* clientData);
```

Description

An annotation changed in the specified way.

Parameters

annot	The annotation that changed.
key	The ASAtom specifying how the annotation changed. The ASAtom corresponding to the key that changed in the annotation's Cos dictionary. See Section 8.4 on annotations in the <i>PDF Reference</i> for information on the keys.
error	Error code. error is set to 0 if no errors occurred while changing the annotation. If an error occurred, error contains the error code.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDAnnotWillChange](#)

Methods

[PDAnnotNotifyDidChange](#)
[PDAnnotSetRect](#)
[PDTextAnnotSetOpen](#)
[PDAnnotSetColor](#)
[PDAnnotSetTitle](#)
[PDAnnotSetDate](#)
[PDAnnotSetFlags](#)
[PDTextAnnotSetContents](#)
[PDLinkAnnotSetBorder](#)
[PDLinkAnnotSetAction](#)
[AVPageViewSetAnnotLocation](#)



PDAannotWasCreated

```
ACCB1 void ACCB2 PDAannotWasCreated (PDAannot annot,  
PDPage page, void* clientData);
```

Description

An annotation was created.

Parameters

annot	The annotation that was created.
page	The page to which the annotation was added.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDAannotDidChange](#)
[PDAannotWillChange](#)

Methods

[PDPageCreateAnnot](#)
[PDPageAddNewAnnot](#)

PDAnnotWillChange

```
ACCB1 void ACCB2 PDAnnotWillChange (PDAnnot annot, ASAtom key,
void* clientData);
```

Description

An annotation will change in the specified way.

Parameters

annot	The annotation that will change.
key	The ASAtom specifying how the annotation will change. The ASAtom corresponding to the key that changed in the annotation's Cos dictionary. See Section 8.4 on annotations in the <i>PDF Reference</i> for information on the keys.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDAnnotDidChange](#)

Methods

[PDAnnotNotifyWillChange](#)
[PDAnnotSetRect](#)
[PDTextAnnotSetOpen](#)
[PDAnnotSetColor](#)
[PDAnnotSetTitle](#)
[PDAnnotSetDate](#)
[PDAnnotSetFlags](#)
[PDTextAnnotSetContents](#)
[PDLinkAnnotSetBorder](#)
[PDLinkAnnotSetAction](#)
[AVPageViewSetAnnotLocation](#)

PDBookmarkDidChange

```
ACCB1 void ACCB2 PDBookmarkDidChange (PDBookmark bookmark,
ASAtom key, ASInt32 err, void* clientData);
```

Description

A bookmark has been opened/closed, its action has been changed, its title has been changed, or children have been added to it.

Parameters

bookmark	The bookmark that will be changed.
key	The ASAtom specifying the change that will occur. See PDBookmarkWillChange for a list of the ASAtoms and their meaning.
err	Error code. error is set to 0 if no errors occurred while changing the bookmark. If an error occurred, error contains the error code.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDBookmarkWillChange](#)
[PDBookmarkDidChangePosition](#)

Methods

[PDBookmarkSetOpen](#)
[PDBookmarkSetAction](#)
[PDBookmarkSetTitle](#)
[PDBookmarkAddSubtree](#)

PDBookmarkDidChangeOpenState

```
ACCB1 void ACCB2 PDBookmarkDidChangePosition  
(PDBookmark bookmark, ASBool open, void* clientData);
```

Description

A bookmark was opened or closed.

Parameters

bookmark	The bookmark that was opened or closed.
open	When true , the bookmark was opened. When false , it was closed.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDBookmarkWillChange](#)
[PDBookmarkDidChange](#)

Methods

[PDBookmarkSetOpen](#)

PDBookmarkDidChangePosition

```
ACCB1 void ACCB2 PDBookmarkDidChangePosition  
(PDBookmark bookmark, void* clientData);
```

Description

One or more bookmarks have been moved.

Parameters

bookmark	The bookmark that was moved.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDBookmarkWillChange](#)
[PDBookmarkDidChange](#)

Methods

[PDBookmarkAddNext](#)
[PDBookmarkAddPrev](#)
[PDBookmarkAddNewChild](#)
[PDBookmarkAddNewSibling](#)
[PDBookmarkAddChild](#)
[PDBookmarkAddSubtree](#)

PDBookmarkDidDestroy

```
ACCB1 void ACCB2 PDBookmarkDidDestroy (PDBookmark bookmark,  
ASInt32 err, void* clientData);
```

Description

A bookmark was destroyed.

Parameters

bookmark	The bookmark that was destroyed. Because the bookmark has been destroyed, it is not possible to access it.
err	Error code. error is set to 0 if no errors occurred while destroying the bookmark. If an error occurred, error contains the error code.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDBookmarkWillDestroy](#)

Methods

[PDBookmarkDestroy](#)

PDBookmarkDidUnlink

```
ACCB1 void ACCB2 PDBookmarkDidUnlink (PDBookmark bookmark,  
void* clientData);
```

Description

A bookmark was unlinked from the bookmark tree.

Parameters

bookmark	The bookmark that was unlinked. Because the bookmark has not been destroyed, it is possible to access it.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDBookmarkDidChangePosition](#)

Methods

[PDBookmarkUnlink](#)

PDBookmarkWasCreated

```
ACCB1 void ACCB2 PDBookmarkWasCreated (PDBookmark bookmark,  
void* clientData);
```

Description

A bookmark was created.

Parameters

bookmark	The bookmark that was created.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDBookmarkDidDestroy](#)
[PDBookmarkWillDestroy](#)

Methods

[PDBookmarkAddNewChild](#)
[PDBookmarkAddNewSibling](#)

PDBookmarkWillChange

```
ACCB1 void ACCB2 PDBookmarkWillChange (PDBookmark bookmark,  
ASAtom key, void* clientData);
```

Description

A bookmark will be opened/closed, its action will be changed, its title will be changed, or children will be added to it.

Parameters

bookmark	The bookmark that will be changed.
key	The ASAtom specifying the change that will occur. key will be an ASAtom corresponding to one of the following: ● Count — Children will be added, or bookmark will be opened/closed. ● A — Action will be changed. ● Title — Title will be changed. ● C — Color will be changed (PDF 1.4) ● F — Flags will be changed (PDF 1.4)
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDBookmarkDidChange](#)
[PDBookmarkDidChangePosition](#)

Methods

[PDBookmarkSetOpen](#)
[PDBookmarkSetAction](#)
[PDBookmarkSetTitle](#)
[PDBookmarkAddSubtree](#)



PDBookmarkWillDestroy

```
ACCB1 void ACCB2 PDBookmarkWillDestroy (PDBookmark bookmark,  
void* clientData);
```

Description

A bookmark will be destroyed.

Parameters

bookmark	The bookmark that will be destroyed.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDBookmarkDidDestroy](#)

Methods

[PDBookmarkDestroy](#)

PDDocCalculateMetadata

```
ACCB1 void ACCB2 PDDocCalculateMetadata (PDDoc pdDoc,  
void* clientData);
```

Description

The client is requested to calculate and set metadata items that depend on the state of the document. Issued when a document is saved. Can also be issued explicitly via [PDDocCalculateImplicitMetadata](#).

Parameters

pdDoc	The document for which implicit metadata is to be calculated.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

None

Methods

Numerous, but especially [PDDocCalculateImplicitMetadata](#)



PDDocDidAddThread

```
ACCB1 void ACCB2 PDDocDidAddThread (PDDoc doc,  
PDThread thread, void* clientData);
```

Description

A thread has been added to a document.

Parameters

doc	The document to which a thread was added.
thread	The thread that was added.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDThreadDidChange](#)

Methods

[PDDocAddThread](#)
[PDDocInsertPages](#)

PDDocDidChangePageAreas

```
ACCB1 void ACCB2 PDDocDidChangePageAreas (PDDoc pdDoc,  
ASInt32 areaMask, ASInt32 firstPage, ASInt32 lastPage,  
void* clientData);
```

Description

Page areas changed.

Parameters

pdDoc	The document in which the page areas changed.
areaMask	The area mask.
firstPage	The first page.
lastPage	The last page.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocDidChangePages](#)

Methods

Numerous

PDDocDidChangePages

```
ACCB1 void ACCB2 PDDocDidChangePages (PDDoc doc,
PDDocOperation op, ASInt32 fromPage, ASInt32 toPage,
ASInt32 error, void* clientData);
```

Description

Pages have been inserted, deleted, moved, or modified.

Parameters

doc	The document in which pages have been changed.
op	The change that was made. op will be one of the PDDocOperation values.
fromPage	The page number of the first page that was modified. For page insertion, this is the number of the page <i>before</i> the first inserted page. For page deletion, this is the page number of the first deleted page.
toPage	The page number of the last page that was modified. If broadcast by PDDocCreatePage , this is the number of the newly created page. When broadcast by PDDocInsertPages , toPage is the number of pages in the document post-insertion; that is, it points to a page not in the document. If broadcast by PDDocDeletePages , toPage is the number of pages in the PDDoc prior to deletion, pointing to a page not in the document.
error	Error code. error is set to 0 if no errors occurred while changing the pages. If an error occurred, error contains the error code.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocWillChangePages](#)

Methods

[PDDocCreatePage](#)
[PDDocInsertPages](#)
[PDDocReplacePages](#)
[PDDocMovePage](#)
[PDPageAddCosResource](#)
[PDPageRemoveCosResource](#)
[PDPageAddCosContents](#)
[PDPageRemoveCosContents](#)



PDDocDeletePages
 PDPageSetRotate
 PDPageSetMediaBox
 PDPageSetCropBox

PDDocDidChangeThumbs

```
ACCB1 void ACCB2 PDDocDidChangeThumbs (PDDoc doc,
void* clientData);
```

Description

Thumbnail images have been added or removed. In addition to the expected ways in which this can occur, it can also occur if pages are inserted into a file.

Parameters

doc	The document in which thumbnail images have been changed.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocDidChangePages](#)
[PDDocDidDeletePages](#)
[PDDocDidInsertPages](#)
[PDDocDidReplacePages](#)
[PDDocWillChangePages](#)
[PDDocWillDeletePages](#)
[PDDocWillInsertPages](#)
[PDDocWillReplacePages](#)

Methods

[PDDocCreatePage](#)
[PDDocCreateThumbs](#)
[PDDocDeleteThumbs](#)
[PDDocInsertPages](#)

PDDocDidClose

```
ACCB1 void ACCB2 PDDocDidClose (PDDoc doc, void* clientData);
```

Description

A **PDDoc** closed. A **PDDoc** is closed only if its reference count is zero.

Neither this notification, [PDDocWillClose](#), [AVDocWillClose](#), nor [AVDocDidClose](#) are broadcast if the user selects **Cancel** when prompted to save a modified document as it is closed.

Parameters

doc	The document that closed.
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PDDocWillClose](#)

Methods

[AVDocClose](#)

[PDDocClose](#)



PDDocDidDeletePages

```
ACCB1 void ACCB2 PDDocDidDeletePages (PDDoc doc,  
ASInt32 fromPage, ASInt32 toPage, ASInt32 error,  
void* clientData);
```

Description

One or more pages were deleted.

Parameters

doc	The document from which pages were deleted.
fromPage	The page number of the first page that was deleted.
toPage	The page number of the last page that was deleted.
error	Error code. error is set to 0 if no errors occurred while deleting the pages. If an error occurred, error contains the error code.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocWillDeletePages](#)
[PDDocDidChangePages](#)

Methods

[PDDocDeletePages](#)



PDDocDidExportAnnots

```
ACCB1 void ACCB2 PDDocDidExportAnnots (PDDoc doc,  
void* clientData);
```

Description

The annotations of a document were exported.

Parameters

doc	The document whose annotations were exported.
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PDDocDidImportAnnots](#)
[PDDocWillExportAnnots](#)
[PDDocWillImportAnnots](#)

Methods

[PDDocExportNotes](#)

PDDocDidImportAnnots

```
ACCB1 void ACCB2 PDDocDidImportAnnots (PDDoc doc,  
void* clientData);
```

Description

The annotations from one document were imported into another document.

Parameters

doc	The document into which annotations were imported.
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PDDocDidImportAnnots](#)
[PDDocWillExportAnnots](#)
[PDDocWillImportAnnots](#)

Methods

[PDDocImportCosDocNotes](#)
[PDDocImportNotes](#)

PDDocDidInsertPages

```
ACCB1 void ACCB2 PDDocDidInsertPages (PDDoc doc,
ASInt32 insertAfterThisPage, PDDoc srcDoc,
ASInt32 srcFromPage, ASInt32 srcToPage, ASInt32 error,
void* clientData);
```

Description

One or more pages have been inserted.

Parameters

doc	The document into which pages were inserted.
insertAfterThisPage	Page number (in doc) after which pages were inserted.
srcDoc	The document that provided the pages that were inserted. This is NULL when a new blank page is created and inserted into a document. This is NULL for a notification broadcast by PDDocCreatePage .
srcFromPage	The page number (in srcDoc) of the first page that was inserted. Not valid when a new blank page is created and inserted into a document. This is NULL for a notification broadcast by PDDocCreatePage .
srcToPage	The page number (in srcDoc) of the last page that was inserted. Not valid when a new blank page is created and inserted into a document. This is NULL for a notification broadcast by PDDocCreatePage .
error	Error code. error is set to 0 if no errors occurred while inserting the pages. If an error occurred, error contains the error code.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocWillInsertPages](#)
[PDDocDidChangePages](#)

Methods

[PDDocCreatePage](#)
[PDDocInsertPages](#)

PDDocDidInsertPagesEx

```
ACCB1 void ACCB2 PDDocDidInsertPagesEx
(PDDocInsertPagesParams params, void* clientData);
```

Description

Pages were inserted into a document. This notification occurs after the **PDDocDidInsertPages** notification.

Parameters

params	A structure describing how pages were inserted.
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PDDocDidInsertPages](#)
[PDDocWillInsertPages](#)
[PDDocWillInsertPagesEx](#)

Methods

[PDDocCreatePage](#)
[PDDocInsertPages](#)

PDDocDidMovePages

```
ACCB1 void ACCB2 PDDocDidMovePages (PDDoc doc,
ASInt32 moveAfterThisPage, ASInt32 fromPage, ASInt32 toPage,
ASInt32 error, void* clientData);
```

Description

One or more pages were moved.

Parameters

doc	The document in which pages were moved.
moveAfterThisPage	The page number after which the moved pages were placed.
fromPage	The page number of the first page that was moved.
toPage	The page number of the last page that was moved.
error	Error code. error is set to 0 if no errors occurred while moving pages. If an error occurred, error contains the error code.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocWillMovePages](#)
[PDDocDidChangePages](#)

Methods

[PDDocMovePage](#)



PDDocDidOpen

```
ACCB1 void ACCB2 PDDocDidOpen (PDDoc doc, void* clientData);
```

Description

A document was opened.

Parameters

doc	The document.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocPermsReady](#)

Methods

Numerous



PDDocDidPrintPage

```
ACCB1 void ACCB2 PDDocDidPrintPage (PDDoc doc, ASInt32 page,
ASStm stm, ASInt32 error, void* clientData);
```

Description

This notification is broadcast once per page that is printed, after all marks have been made on the page. When printing to a PostScript printer, printing commands can also be sent that will be placed on the page after all other marks.

Parameters

doc	The document from which a page was printed.
page	The page number of the page that was printed.
stm	The PostScript print stream when printing to a PostScript printer, and NULL when printing to a non-PostScript printer. When printing to a PostScript printer, clients can write printing commands into stm (using ASStmWrite) to add marks to the printed page after all other marks have been made. See PDDocWillPrintPage for a description of the sequence of operations when printing a page to a PostScript printer.
error	Error code. error is set to 0 if no errors occurred while printing the page. If an error occurred, error contains the error code.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocWillPrintPage](#)
[PDDocDidPrintPages](#)

Methods

[AVDocPrintPages](#)
[AVDocPrintPagesWithParams](#)

PDDocDidPrintPages

```
ACCB1 void ACCB2 PDDocDidPrintPages (PDDoc doc,
ASInt32 fromPage, ASInt32 toPage, ASInt32 error,
void* clientData);
```

Description

This notification is broadcast after printing ends.

Parameters

doc	The document from which pages were printed.
fromPage	The page number of the first page that was printed.
toPage	The page number of the last page that was printed.
error	Error code. error is set to 0 if no errors occurred while printing the pages. If an error occurred, error contains the error code.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocWillPrintPages](#)
[PDDocDidPrintPage](#)

Methods

[AVDocPrintPages](#)
[AVDocPrintPagesWithParams](#)

PDDocDidPrintTiledPage

```
ACCB1 void ACCB2 PDDocDidPrintTiledPage (PDDoc doc,
ASInt32 page, ASStm stm, ASInt32 error, ASInt32 whichSheet,
ASInt32 totalSheets, void* clientData);
```

Description

Clients who register for **PDDocDidPrintTiledPage** will be called after the tile marks (if any) have been emitted for this tiled page. Tiled pages can be requested by the user from the Advanced print dialog; clients are made aware of the selection via the tiled page notifications.

Parameters

doc	The document containing the tiled page.
page	The page being printed.
stm	The PostScript print stream when printing to a PostScript printer, and NULL when printing to a non-PostScript printer. When printing to a PostScript printer, clients can write printing commands into stm (using ASStmWrite) to add marks to pages before any other marks have been made.
error	Error code. error is set to 0 if no errors occurred while printing the pages. If an error occurred, error contains the error code.
whichSheet	The sheet to be printed.
totalSheets	The number of sheets to be printed.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocPrintingTiledPage](#)
[PDDocWillPrintTiledPage](#)

Methods

[AVDocPrintPages](#)
[AVDocPrintPagesWithParams](#)



PDDocDidRemoveThread

```
ACCB1 void ACCB2 PDDocDidRemoveThread (PDDoc doc,
ASInt32 index, void* clientData);
```

Description

A thread was removed from a document.

Parameters

doc	The document from which a thread was removed.
index	The index of the thread that was removed. Because the thread has already been removed, it is not possible to access it using index .
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocWillRemoveThread](#)

Methods

[PDDocRemoveThread](#)

PDDocDidReplacePages

```
ACCB1 void ACCB2 PDDocDidReplacePages (PDDoc doc,
ASInt32 fromPage, ASInt32 toPage, PDDoc srcDoc,
ASInt32 srcFromPage, ASInt32 srcToPage, ASInt32 error,
void* clientData);
```

Description

One or more pages have been replaced.

Parameters

doc	The document in which pages have been replaced.
fromPage	The page number (in doc) of the first page that was replaced.
toPage	The page number (in doc) of the last page that was replaced.
srcDoc	The document that provided the replacement pages.
srcFomPage	The page number (in srcDoc) of the first replacement page.
srcToPage	The page number (in srcDoc) of the last replacement page.
error	Error code. error is set to 0 if no errors occurred while replacing pages. If an error occurred, error contains the error code.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocWillReplacePages](#)
[PDDocDidChangePages](#)

Methods

[PDDocReplacePages](#)

PDDocDidSave

```
ACCB1 void ACCB2 PDDocDidSave (PDDoc doc, ASInt32 err,  
void* clientData);
```

Description

A document has been saved.

The **PDDocDidSave** notification takes place just after the save operation finishes. At the time of a **PDDocDidSave** notification, a client or application can reacquire resources from the **PDDoc** if needed. It should examine the error code **err** associated with the save. If the save was not successful, the error code is non-zero. In the case of a unsuccessful save, a client or application should *not* attempt to do anything further with this **PDDoc**.

See [PDDocWillSaveEx](#) for important information on releasing objects derived from the **PDDoc** before it is saved.

Parameters

doc	The document that was saved.
err	Error code. error is set to 0 if no errors occurred while saving the file. If an error occurred, error contains the error code.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocWillSave](#)

[PDDocWillSaveEx](#)

Methods

[CosObjRefreshAfterLinearizedSave](#)

[PDDocSave](#)

PDDocOCDidChange

```
ACCB1 void ACCB2 PDDocOCDidChange (PDDoc doc,  
PDDocOCChangeType whatHappened, void* object,  
void* clientData);
```

Description

An optional-content context changed in a **PDDoc** in a way that could affect the visibility state of content.

Parameters

doc	The document containing the optional-content context whose visibility state changed.
whatHappened	The type of change that occurred.
object	A pointer to the object that changed. The kind of object this can be depends on the type of change that occurred: kPDOCGCreate: PDOCG kPDOCGDestroy: NULL kPDOCGProperties: PDOCG kPDOCConfigChange: PDOCConfig kPDOCConfigCreate: PDOCConfig kPDOCConfigDestroy: NULL kPDOCGReplace: PDOCG (the replacement) kPDDocRemoveOC: NULL
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PDDocOCWillChange](#)
[PDOCContextDidChange](#)
[PDOCContextWillChange](#)

Methods

numerous

PDDocOCWillChange

```
ACCB1 void ACCB2 PDDocOCWillChange (PDDoc doc,
PDDocOCChangeType whatWillHappen, void* object,
void* clientData);
```

Description

An optional-content context is changing in a **PDDoc** in a way that could affect the visibility state of content.

Parameters

doc	The document containing the optional-content context whose visibility state is changing.
whatWillHappen	The type of change that will occur.
object	A pointer to the object that will change. The kind of object this can be depends on the type of change that will occur: kPDOCGCreate: NULL kPDOCGDestroy: PDOCG kPDOCGProperties: PDOCG kPDOCConfigChange: PDOCConfig kPDOCConfigCreate: NULL kPDOCConfigDestroy: PDOCConfig kPDOCGReplace: PDOCG (the one being replaced) kPDDocRemoveOC: NULL
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PDDocOCDidChange](#)
[PDOCContextDidChange](#)
[PDOCContextWillChange](#)

Methods

numerous



PDDocPermsReady

```
ACCB1 void ACCB2 PDDocPermsReady (PDDoc doc,  
void* clientData);
```

Description

A document was opened and its permissions (if present) have been validated.

Parameters

doc	The document.
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PDDocDidOpen](#)

Methods

numerous

PDDocPageLabelDidChange

```
ACCB1 void ACCB2 PDDocPageLabelDidChange (PDDoc doc,  
ASInt32 firstPage, ASInt32 lastPage, void* clientData);
```

Description

A range of pages' labels changed in a **PDDoc**.

Parameters

doc	The document containing the pages whose labels changed.
firstPage	The number of the first page whose label changed.
lastPage	The number of the last page whose label changed.
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

None

Methods

[PDDocSetPageLabel](#)

PDDocPrintingTiledPage

```
ACCB1 void ACCB2 PDDocPrintingTiledPage (PDDoc doc,
ASInt32 page, ASStm stm, ASInt32 whichRow,
ASInt32 whichColumn, ASInt32 numRows, ASInt32 numColumns,
ASFixedRect* cropRegion, PDTile tile, void* clientData);
```

Description

Clients who register for **PDDocPrintingTilePage** will be called just after the page contents have been emitted for this tile. Tiled pages can be requested by the user from the Advanced print dialog; clients are made aware of the selection via the tiled page notifications.

Parameters

doc	The document containing the tiled page.
pageNum	The page being printed.
stm	The PostScript print stream when printing to a PostScript printer, and NULL when printing to a non-PostScript printer. When printing to a PostScript printer, clients can write printing commands into stm (using ASStmWrite) to add marks to pages before any other marks have been made.
whichRow	The row that is being printed.
whichColumn	The column that is being printed.
numRows	The number of rows to be printed.
numColumns	The number of columns to be printed.
cropRegion	A pointer to an ASFixedRect that represents the region of the PDPage being "cropped to" for this sheet.
tile	The PDTile currently in use. The fields in this structure can be modified as necessary to affect tiling output.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocDidPrintTiledPage](#)
[PDDocWillPrintTiledPage](#)



Methods

[AVDocPrintPages](#)

[AVDocPrintPagesWithParams](#)

PDDocWillChangePages

```
ACCB1 void ACCB2 PDDocWillChangePages (PDDoc doc,
PDOperation op, ASInt32 fromPage, ASInt32 toPage,
void* clientData);
```

Description

Pages will be inserted, deleted, moved, or modified.

Parameters

doc	The document in which pages will be changed.
op	The change that will be made. op will be one of the PDOperation values.
fromPage	The page number of the first page that will be modified.
toPage	The page number of the last page that will be modified.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocDidChangePages](#)

Methods

[PDDocDeletePages](#)
[PDPageSetRotate](#)
[PDPageSetMediaBox](#)
[PDPageSetCropBox](#)

PDDocWillClose

```
ACCB1 void ACCB2 PDDocWillClose (PDDoc doc, void* clientData);
```

Description

A **PDDoc** will be closed. A **PDDoc** is closed only if its reference count is zero.

Neither this notification, [PDDocDidClose](#), [AVDocWillClose](#), nor [AVDocDidClose](#) are broadcast if the user selects “Cancel” when prompted to save a modified document as it is closed.

Parameters

doc	The document to close.
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PDDocDidClose](#)

Methods

[AVDocClose](#)

[PDDocClose](#)

PDDocWillDeletePages

```
ACCB1 void ACCB2 PDDocWillDeletePages (PDDoc doc,
ASInt32 fromPage, ASInt32 toPage, void* clientData);
```

Description

One or more pages will be deleted.

Parameters

doc	The document from which pages will be deleted.
fromPage	The page number of the first page that will be deleted.
toPage	The page number of the last page that will be deleted.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocDidDeletePages](#)
[PDDocDidChangePages](#)

Methods

[PDDocDeletePages](#)

PDDocWillExportAnnots

```
ACCB1 void ACCB2 PDDocWillExportAnnots (PDDoc doc,  
void* clientData);
```

Description

The annotations of a document will be exported.

Parameters

doc	The document whose annotations will be exported.
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PDDocDidExportAnnots](#)
[PDDocDidImportAnnots](#)
[PDDocWillImportAnnots](#)

Methods

[PDDocExportNotes](#)



PDDocWillImportAnnots

```
ACCB1 void ACCB2 PDDocWillImportAnnots (PDDoc doc,  
void* clientData);
```

Description

The annotations from one document will be imported into another document.

Parameters

doc	The document into which annotations will be imported.
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PDDocDidExportAnnots](#)
[PDDocDidImportAnnots](#)
[PDDocWillImportAnnots](#)

Methods

[PDDocImportCosDocNotes](#)
[PDDocImportNotes](#)

PDDocWillInsertPages

```
ACCB1 void ACCB2 PDDocWillInsertPages (PDDoc doc,
ASInt32 insertAfterThisPage, PDDoc srcDoc,
ASInt32 srcFromPage, ASInt32 srcToPage, void* clientData);
```

Description

One or more pages will be inserted.

Parameters

doc	The document into which pages will be inserted.
insertAfterThisPage	Page number (in doc) after which pages will be inserted.
srcDoc	The document that provides the pages to insert. This is NULL when a new blank page is created and inserted into a document. This is NULL for a notification broadcast by PDDocCreatePage .
srcFromPage	The page number (in srcDoc) of the first page that will be inserted. Not valid when a new blank page is created and inserted into a document. This is NULL for a notification broadcast by PDDocCreatePage .
srcToPage	The page number (in srcDoc) of the last page that will be inserted. Not valid when a new blank page is created and inserted into a document. This is NULL for a notification broadcast by PDDocCreatePage .
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocDidInsertPages](#)
[PDDocWillChangePages](#)

Methods

[PDDocCreatePage](#)
[PDDocInsertPages](#)

PDDocWillInsertPagesEx

```
ACCB1 void ACCB2 PDDocWillInsertPagesEx
(PDDocInsertPagesParams params, void* clientData);
```

Description

Pages will be inserted into a document. This notification occurs after the [PDDocWillInsertPages](#) notification.

Parameters

params	A structure describing how pages will be inserted.
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PDDocDidInsertPages](#)
[PDDocDidInsertPagesEx](#)
[PDDocWillInsertPages](#)

Methods

[PDDocCreatePage](#)
[PDDocInsertPages](#)



PDDocWillMovePages

```
ACCB1 void ACCB2 PDDocWillMovePages (PDDoc doc,
ASInt32 moveAfterThisPage, ASInt32 fromPage, ASInt32 toPage,
void* clientData);
```

Description

One or more pages will be moved.

Parameters

doc	The document in which pages will be moved.
moveAfterThisPage	The page number after which the moved pages will be placed.
fromPage	The page number of the first page to move.
toPage	The page number of the last page to move.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocDidMovePages](#)
[PDDocWillChangePages](#)

Methods

[PDDocMovePage](#)

PDDocWillPrintDoc

```
ACCB1 void ACCB2 PDDocWillPrintDoc (PDDoc doc, ASStm stm,
ASInt32 psLevel, void* clientData);
```

Description

This notification is broadcast before a document is printed, before any marks are made on the first page. When printing to a PostScript printer, printing commands can also be sent that are placed on the page before any other marks. For example, a **setpagedevice** operator could be placed in the print stream.

NOTE: Page resources and contents cannot be modified reliably at the time this notification is broadcast.

Parameters

doc	The document that is about to be printed.
stm	The PostScript print stream when printing to a PostScript printer, and NULL when printing to a non-PostScript printer. When printing to a PostScript printer, clients can write printing commands into stm (using ASStmWrite) to add marks to pages before any other marks have been made. In the 2.x Acrobat viewers, the page printing sequence to a PostScript printer is: <i>page setup (including setpagedevice)...save... gsave...save...begin... begin...begin... PDDocWillPrintPage... page contents... PDDocDidPrintPage... end... end... end... restore... restore... showpage.</i> This sequence must not be relied on, and it is to some extent dependent on the printer driver in use. Nevertheless, it is true that by the time the PDDocWillPrintPage notification is broadcast, it is too late to perform any setpagedevice operations.
psLevel	When printing to a PostScript printer, psLevel is either 1, 2, or 3, representing the PostScript level available on the printer. When printing to a non-PostScript printer, psLevel is 0. psLevel is useful in determining whether the output device is a PostScript printer. In addition, when printing to a PostScript printer, psLevel is useful to determine the operators that can be sent in any printing code downloaded using the PDDocWillPrintDoc, PDDocWillPrintPage, and PDDocDidPrintPage notifications.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification.

**Related Notifications**[PDDocDidPrintPage](#)[PDDocWillPrintPages](#)**Methods**[AVDocPrintPages](#)[AVDocPrintPagesWithParams](#)

PDDocWillPrintDocInMode

```
ACCB1 void ACCB2 PDDocWillPrintDocInMode (PDDoc doc,  
PDPrintWhat printMode, void* clientData);
```

Description

Sent before a document is printed, before any marks are made on the first page. Page resources and contents may be modified at the time this notification is broadcast.

Parameters

doc	The document that is about to be printed.
printMode	A constant that describes what is being printed—the document only, the document with annotations, or the form data only.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocDidPrintPages](#)
[PDDocWillPrintDoc](#)

Methods

[AVDocPrintPages](#)
[AVDocPrintPagesWithParams](#)

PDDocWillPrintPage

```
ACCB1 void ACCB2 PDDocWillPrintPage (PDDoc doc, ASInt32 page,
ASStm stm, void* clientData);
```

Description

This notification is broadcast once per page that is printed, before any marks are made on the page. When printing to a PostScript printer, printing commands can also be sent that will be placed on the page before any other marks.

NOTE: Page resources and contents cannot be modified reliably at the time this notification is broadcast.

Parameters

doc	The document from which a page is about to be printed.
page	The page number of the page that is about to be printed.
stm	The PostScript print stream when printing to a PostScript printer, or NULL when printing to a non-PostScript printer. When printing to a PostScript printer, clients can write printing commands into stm with ASStmWrite to add marks to the printed page before any other marks have been made. In the 2.x Acrobat viewers, the page printing sequence to a PostScript printer is: <i>page setup (including setpagedevice)...save... gsave...save...begin... begin...begin... PDDocWillPrintPage... page contents... PDDocDidPrintPage... end... end... end... restore... restore... showpage.</i> This sequence must not be relied on, and depends on the printer driver in use. However, by the time the PDDocWillPrintPage notification is broadcast, it is too late to perform any setpagedevice operations.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocDidPrintPage](#)
[PDDocWillPrintPages](#)

Methods

[AVDocPrintPages](#)
[AVDocPrintPagesWithParams](#)

PDDocWillPrintPages

```
ACCB1 void ACCB2 PDDocWillPrintPages (PDDoc doc,
ASInt32 fromPage, ASInt32 toPage, ASInt32 psLevel,
ASBool binaryOK, void* clientData);
```

Description

This notification is broadcast when printing begins, before any pages are printed.

NOTE: Page resources and contents cannot be modified reliably at the time this notification is broadcast.

Parameters

doc	The document from which pages will be printed.
fromPage	The page number of the first page that will be printed.
toPage	The page number of the last page that will be printed.
psLevel	When printing to a PostScript printer, psLevel is either 1, 2, or 3, representing the PostScript level available on the printer. When printing to a non-PostScript printer, psLevel is 0. psLevel is useful in determining whether the output device is a PostScript printer. In addition, when printing to a PostScript printer, psLevel is useful to determine the operators that can be sent in any printing code downloaded using the PDDocWillPrintDoc , PDDocWillPrintPage , and PDDocDidPrintPage notifications.
binaryOK	Valid only when printing to a PostScript printer. Indicates whether binary data can be sent.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocWillPrintDoc](#)
[PDDocWillPrintPage](#)
[PDDocDidPrintPages](#)

Methods

[AVDocPrintPages](#)
[AVDocPrintPagesWithParams](#)

PDDocWillPrintTiledPage

```
ACCB1 void ACCB2 PDDocWillPrintTiledPage (PDDoc doc,
ASInt32 pageNum, ASStm stm, ASInt32 whichSheet,
ASInt32 totalSheets, ASFixedRect* cropRegion, PDTile tile,
void* clientData);
```

Description

This notification is broadcast before a tiled page is printed. Tiled pages can be requested by the user from the Advanced print dialog; clients are made aware of the selection via the tiled page notifications.

Parameters

doc	The document containing the tiled page.
pageNum	The page to be printed.
stm	The PostScript print stream when printing to a PostScript printer, and NULL when printing to a non-PostScript printer. When printing to a PostScript printer, clients can write printing commands into stm (using ASStmWrite) to add marks to pages before any other marks have been made.
whichSheet	The sheet to be printed.
totalSheets	The total number of sheets to be printed.
cropRegion	A pointer to an ASFixedRect that represents the region of the PDPage being "cropped to" for this sheet.
tile	The PDTile currently in use. The fields in this structure can be modified as necessary to affect tiling output.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocDidPrintTiledPage](#)
[PDDocPrintingTiledPage](#)

Methods

[AVDocPrintPages](#)
[AVDocPrintPagesWithParams](#)

PDDocWillRemoveThread

```
ACCB1 void ACCB2 PDDocWillRemoveThread (PDDoc doc,
ASInt32 index, void* clientData);
```

Description

A thread will be removed from a document.

Parameters

doc	The document from which a thread will be removed.
index	The index of the thread that will be removed. Use PDDocGetThread to obtain the thread from its index.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocDidRemoveThread](#)

Methods

[PDDocRemoveThread](#)

PDDocWillReplacePages

```
ACCB1 void ACCB2 PDDocWillReplacePages (PDDoc doc,
ASInt32 fromPage, ASInt32 toPage, PDDoc srcDoc,
ASInt32 srcFromPage, ASInt32 srcToPage, void* clientData);
```

Description

One or more pages will be replaced.

Parameters

doc	The document in which pages will be replaced.
fromPage	The page number (in doc) of the first page that will be replaced.
toPage	The page number (in doc) of the last page that will be replaced.
srcDoc	The document that provides the replacement pages.
srcFomPage	The page number (in srcDoc) of the first replacement page.
srcToPage	The page number (in srcDoc) of the last replacement page.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocDidReplacePages](#)
[PDDocWillChangePages](#)

Methods

[PDDocReplacePages](#)



PDDocWillSave

```
ACCB1 void ACCB2 PDDocWillSave (PDDoc doc, void* clientData);
```

Description

A document will be saved.

The **PDDocWillSave** notification takes place just before the save operation begins. At the time of a **PDDocWillSave** notification, the current **PDDoc** is valid.

See [PDDocWillSaveEx](#) for important information on releasing objects derived from the **PDDoc** before it is saved, and [PDDocDidSave](#) for information on reacquiring objects from the **PDDoc** after it has been saved.

Parameters

doc	The document that will be saved.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocDidSave](#)

[PDDocWillSaveEx](#)

Methods

[PDDocSave](#)

[PDDocSaveWithParams](#)

PDDocWillSaveEx

```
ACCB1 void ACCB2 PDDocWillSaveEx (PDDoc doc,  
PDDocSaveParams params, void* clientData);
```

Description

A document will be saved.

The **PDDocWillSaveEx** notification takes place just before the save operation begins. At the time of a **PDDocWillSaveEx** notification, the current **PDDoc** is valid.

At this time, clients should look at the save flags field **saveFlags** in **params**. In the case of a full save, they should release any objects that they have acquired from the **PDDoc doc** with methods, such as [PDPageRelease](#). During this notification, plug-in's should also forget any PD-level or Cos-level objects that had been derived from the **PDDoc**; these will not be valid after the save.

See [PDDocDidSave](#) for information on reacquiring objects from the **PDDoc** after it has been saved.

Parameters

doc	The document that will be saved.
params	The PDDocSaveParams parameters used when saving a file using PDDocSaveWithParams .
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocDidSave](#)
[PDDocWillSave](#)

Methods

[PDDocSave](#)
[PDDocSaveWithParams](#)

PDNameTreeNameAdded

```
ACCB1 void ACCB2 PDNameTreeNameAdded (PDNameTree nameTree,
CosObj key, CosObj value, void* clientData);
```

Description

An entry was added to a name tree.

Parameters

nameTree	The name tree to which an entry was added.
key	Cos object of the key for the entry. This object is a Cos integer.
value	Cos object for the value associated with key .
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PDNameTreeNameRemoved](#)

Related Methods

[PDNameTreeNotifyNameAdded](#)

PDNameTreeNameRemoved

```
ACCB1 void ACCB2 PDNameTreeNameRemoved (PDNameTree nameTree,  
CosObj key, void* clientData);
```

Description

An entry was removed from a name tree.

Parameters

nameTree	The name tree from which an entry was removed.
key	The Cos object of the key for the entry. This object is a Cos integer.
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Notifications

[PDNameTreeNameAdded](#)

Methods

[PDNameTreeNotifyNameRemoved](#)

PDNumTreeNumAdded

```
ACCB1 void ACCB2 PDNumTreeNumAdded (PDNumTree numTree,
ASInt32 key, CosObj value, void* clientData);
```

Description

An entry was added to a name tree.

Parameters

nameTree	The name tree to which an entry was added.
key	The key for the entry.
value	Cos object for the value associated with key .
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PDNumTreeNumRemoved](#)

Methods

[PDNumTreePut](#)

PDNumTreeNumRemoved

```
ACCB1 void ACCB2 PDNumTreeNumRemoved (PDNumTree numTree,  
ASInt32 key, void* clientData);
```

Description

An entry was removed from a name tree.

Parameters

nameTree	The name tree from which an entry was removed.
key	The key for the entry.
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PDNumTreeNumAdded](#)

Methods

[PDNumTreePut](#)

[PDNumTreeRemove](#)

PDOCContextDidChange

```
ACCB1 void ACCB2 PDDOCContextDidChange (PDOCContext ocContext,
PDOCContextChangeType whatHappened, void* objects,
void* clientData);
```

Description

An optional-content context changed in a way that could affect the visibility state of content.

Parameters

ocContext	The optional-content context whose visibility state changed.
whatHappened	The type of change that occurred.
objects	If whatHappened is kPDOCState , this is a NULL -terminated array of PDOC objects. Otherwise, it is NULL .
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PDOCContextWillChange](#)
[PDDocOCDidChange](#)
[PDDocOCWillChange](#)

Methods

[PDOCContextSetNonOCDrawing](#)
[PDOCContextSetOCDrawEnumType](#)
[PDOCContextSetIntent](#)
[PDOCContextInit](#)

PDOCContextWillChange

```
ACCB1 void ACCB2 PDDOCContextWillChange  
(PDOCContext ocContext, PDOCContextChangeType whatWillHappen,  
void* objects, void* clientData);
```

Description

An optional-content context is changing in a way that could affect the visibility state of content.

Parameters

ocContext	The optional-content context whose visibility state is changing.
whatWillHappen	The type of change that will occur.
objects	If whatHappened is kPDOCState , this is a NULL -terminated array of PDOC objects. Otherwise, it is NULL .
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PDOCContextDidChange](#)
[PDDocOCDidChange](#)
[PDDocOCWillChange](#)

Methods

[PDOCContextSetNonOCDrawing](#)
[PDOCContextSetOCDrawEnumType](#)
[PDOCContextSetIntent](#)
[PDOCContextInit](#)

PDPPageContentsDidChange

```
ACCB1 void ACCB2 PDPPageContentsDidChange (PDPPage page,  
void* clientData);
```

Description

The contents of a page have changed and the page will be redrawn.

Parameters

page	The page whose contents changed.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDPPageContentsDidChangeEx](#)

Methods

[PDPPageNotifyContentsDidChange](#)

[PDPPageNotifyContentsDidChangeEx](#)

PDPageContentsDidChangeEx

```
ACCB1 void ACCB2 PDPageContentsDidChangeEx (PDPage page,  
ASBool invalidateViews, void* clientData);
```

Description

The contents of a page changed. Unlike [PDPageContentsDidChange](#), this notification specifies whether the page is redrawn immediately.

Parameters

page	The page whose contents changed.
invalidateViews	If true , the page is redrawn immediately. If false , redrawing is suppressed.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDPageContentsDidChange](#)

Methods

[PDPageNotifyContentsDidChange](#)

[PDPageNotifyContentsDidChangeEx](#)

PDPageDidAddAnnot

```
ACCB1 void ACCB2 PDPageDidAddAnnot (PDPage page,  

PDAnnot annot, ASInt32 error, void* clientData);
```

Description

An annotation was added to a page.

Parameters

page	The page to which the annotation was added.
annot	The annotation that was added.
error	Error code. error is set to 0 if no errors occurred while adding the annotation. If an error occurred, error contains the error code.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDPageWillAddAnnot](#)

Methods

[PDPageAddAnnot](#)
[PDPageAddNewAnnot](#)

PDPageDidPrintAnnot

```
ACCB1 void ACCB2 PDPageDidPrintAnnot (PDAnnot annot,
ASInt32 page, ASInt32 status, void* clientData);
```

Description

Clients who register for **PDPageDidPrintAnnot** will be called after the annotation has printed. **status** indicates whether Acrobat tried to print any representation of this annotation.

Parameters

annot	The annotation that Acrobat tried to print.
page	The page on which the annotation occurs.
status	A status of 0 indicates that no representation of the annot was found. A status of 1 indicates the annot was printed.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDPageDidPrintAnnot](#)
[PDPageWillPrintAnnot](#)
[PDPageWillPrintAnnots](#)

Methods

Numerous

PDPageDidPrintAnnots

```
ACCB1 void ACCB2 PDPageDidPrintAnnots (PDDoc doc,  
ASInt32 page, ASStm stm, void* clientData);
```

Description

Annotations were printed.

Parameters

doc	The document containing the annotations.
page	The page containing the annotations.
stm	The PostScript print stream when printing to a PostScript printer, and NULL when printing to a non-PostScript printer. When printing to a PostScript printer, clients can write printing commands into stm (using ASStmWrite) to add marks to pages before any other marks have been made.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDPageDidPrintAnnot](#)
[PDPageWillPrintAnnot](#)
[PDPageWillPrintAnnots](#)

Methods

Numerous

PDPageDidRemoveAnnot

```
ACCB1 void ACCB2 PDPageDidRemoveAnnot (PDPage page,
ASInt32 annotIndex, ASInt32 error, void* clientData);
```

Description

An annotation has been removed from a page.

NOTE: Superseded by [PDPageDidRemoveAnnotEx](#) in Acrobat 6.0.

Parameters

page	The page from which an annotation was removed.
annotIndex	The index (in the page's annotation array) of the annotation that was removed. Because the annotation has already been removed from the array, it is not possible to access the annotation using annotIndex .
error	Error code. error is set to 0 if no errors occurred while removing the annotation. If an error occurred, error contains the error code.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDPageWillRemoveAnnot](#)

Methods

[PDPageRemoveAnnot](#)

PDPageDidRemoveAnnotEx

```
ACCB1 void ACCB2 PDPageDidRemoveAnnotEx (PDPage page,
PDAnnot removedAnnot, ASInt32 error, void* clientData);
```

Description

An annotation has been removed from a page.

NOTE: Supersedes [PDPageDidRemoveAnnot](#) in Acrobat 6.0.

Parameters

page	The page from which an annotation was removed.
removedAnnot	The annotation object that was removed.
error	Error code. error is set to 0 if no errors occurred while removing the annotation. If an error occurred, error contains the error code.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDPageWillRemoveAnnot](#)

Methods

[PDPageRemoveAnnot](#)

PDPageWillAddAnnot

```
ACCB1 void ACCB2 PDPageWillAddAnnot (PDPage page,
ASInt32 addAfter, PDAnnot annot, void* clientData);
```

Description

An annotation will be added to a page.

Parameters

page	The page to which the annotation will be added.
addAfter	The index in the page's annotation array after which the annotation will be added.
annot	The annotation that will be added.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDPageDidAddAnnot](#)

Methods

[PDPageAddAnnot](#)
[PDPageAddNewAnnot](#)

PDPageWillPrintAnnot

```
ACCB1 void ACCB2 PDPageWillPrintAnnot (PDAnnot annot,
ASInt32 pageNum, void* clientData);
```

Description

Clients who register for this notification will be notified just before an annotation is expected to print. This notification allows clients that manage annotations to prepare the appearance of the annotation for printing purposes. There is no requirement that the annot referred to in this parameter list actually have a print appearance.

Parameters

annot	The annotation to print.
pageNum	The page on which the annotation occurs.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDPageDidPrintAnnot](#)
[PDPageDidPrintAnnots](#)
[PDPageWillPrintAnnot](#)

Methods

Numerous

PDPageWillPrintAnnots

```
ACCB1 void ACCB2 PDPageWillPrintAnnots (PDDoc doc,
ASInt32 pageNum, ASStm stm, void* clientData);
```

Description

Clients who register for **PDPageWillPrintAnnots** will be called before the printed representations of any annotations have been emitted. If a page has no annotations, this will not be called. If a page has annotations, this will be called. There may not be any code emitted for the annotations on that page, however, since they may not have any appearance for printing. The **status** parameter passed in the [PDPageDidPrintAnnot](#) will indicate this.

Parameters

doc	The document containing the annotations.
pageNum	The page to be printed.
stm	The PostScript print stream when printing to a PostScript printer, and NULL when printing to a non-PostScript printer. When printing to a PostScript printer, clients can write printing commands into stm (using ASStmWrite) to add marks to pages before any other marks have been made.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDPageDidPrintAnnot](#)
[PDPageDidPrintAnnots](#)
[PDPageWillPrintAnnot](#)

Methods

Numerous

PDPageWillRemoveAnnot

```
ACCB1 void ACCB2 PDPageWillRemoveAnnot (PDPage page,  
ASInt32 annotIndex, void* clientData);
```

Description

An annotation will be removed from a page.

Parameters

page	The page from which an annotation will be removed.
annotIndex	The index (in the page's annotation array) of the annotation that will be removed. Use PDPageGetAnnot to obtain the annotation from its index.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDPageDidRemoveAnnot](#)

Methods

[PDPageRemoveAnnot](#)



PDThreadDidChange

```
ACCB1 void ACCB2 PDThreadDidChange (PDThread thread,  
void* clientData);
```

Description

A thread was changed.

Parameters

thread	The thread that changed.
clientData	Pointer to a block of user-supplied data that was passed when the client registered for this notification using AVAppRegisterNotification .

Related Notifications

[PDDocDidAddThread](#)

Methods

[PDThreadSetInfo](#)

PSPrintAfterBeginPageSetup

```
ACCB1 void ACCB2 PSPrintAfterBeginPageSetup (PDDoc doc,  
ASInt32 page, ASStm stm, void* clientData);
```

Description

This notification is broadcast after the beginning of the *Page Setup* (immediately after writing **%%BeginPageSetup**) during the printing of a page to a PostScript printer with the methods **PDFLPrintDoc** (only available with PDF Library SDK) or **PDDocPrintPages** (only available with PDF Library SDK). At this point it is possible to use **setpagedevice** and set the graphics state but marks cannot be made on the page.

Parameters

doc	The document from which a page is printed.
page	Page number of the page being printed.
stm	The PostScript print stream. PostScript commands can be added to the print stream using ASStmWrite . Some printer drivers (Windows PS4) may not allow additions to the PostScript stream at this point.
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PSPrintAfterPageTrailer](#)

Methods

PDDocPrintPages (Only available with PDF Library SDK)

PDFLPrintDoc (Only available with PDF Library SDK)

PSPrintAfterBeginProlog

```
ACCB1 void ACCB2 PSPrintAfterBeginProlog (PDDoc doc,
ASStm stm, void* clientData);
```

Description

This notification is broadcast after the beginning of the PostScript *Prolog* (immediately after writing **%%BeginPrologue**) during the printing of a document to a PostScript printer with the methods **PDFLPrintDoc** (only available with PDF Library SDK) or **PDDocPrintPages** (only available with PDF Library SDK). The Prolog is a set of application-specific procedure definitions that an application may emit in a PostScript stream.

At this point nothing should be added to the PostScript print stream that modifies the graphics state or puts marks on the page. Callers should only emit procset resources.

Parameters

doc	The document that is being printed.
stm	The PostScript print stream. PostScript commands can be added to the print stream using ASStmWrite .
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PSPrintAfterBeginSetup](#)

Methods

PDDocPrintPages (Only available with PDF Library SDK)
PDFLPrintDoc (Only available with PDF Library SDK)

PSPrintAfterBeginSetup

```
ACCB1 void ACCB2 PSPrintAfterBeginSetup (PDDoc doc, ASStm stm,
void* clientData);
```

Description

This notification is broadcast after the beginning of the *Document Setup* (immediately after writing **%%BeginSetup**) during the printing of a page to a PostScript printer with the methods **PDFLPrintDoc** (only available with PDF Library SDK) or **PDDocPrintPages** (only available with PDF Library SDK). During Document Setup, fonts may be downloaded, **setpagedevice** may be called, procsets may be initialized, the graphics state may be initialized, and so forth.

Parameters

doc	The document that is being printed.
stm	The PostScript print stream. PostScript commands can be added to the print stream using ASStmWrite .
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PSPrintBeforeEndSetup](#)

Methods

PDDocPrintPages (Only available with PDF Library SDK)

PDFLPrintDoc (Only available with PDF Library SDK)

PSPrintAfterEmitExtGState

```
ACCB1 void ACCB2 PSPrintAfterEmitExtGState (ASStm stm,  
void* clientData);
```

Description

This notification is broadcast after extended graphics state parameters are emitted while printing to a PostScript printer with the methods **PDFLPrintDoc** (only available with PDF Library SDK) or **PDDocPrintPages** (only available with PDF Library SDK).

These parameters are typically device-dependent. For information on extended graphics state, see Section 4.3.4 on extended graphics states in [PDF Reference](#).

Parameters

stm	The PostScript print stream. PostScript commands can be added to the print stream using ASStmWrite .
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

None

Methods

PDDocPrintPages (Only available with PDF Library SDK)
PDFLPrintDoc (Only available with PDF Library SDK)

PSPrintAfterPageTrailer

```
ACCB1 void ACCB2 PSPrintAfterPageTrailer (PDDoc doc, ASInt32
page, ASStm stm, void* clientData);
```

Description

This notification is broadcast after the page trailer is emitted (immediately after writing **%%PageTrailer**) during the printing of a page to a PostScript printer with the methods **PDFLPrintDoc** (only available with PDF Library SDK) or **PDDocPrintPages** (only available with PDF Library SDK). At this point it is possible to resolve comments (at end) and emit cleanup code.

Parameters

doc	The document from which a page is printed.
page	Page number of the page being printed.
stm	The PostScript print stream. PostScript commands can be added to the print stream using ASStmWrite .
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PSPrintAfterBeginPageSetup](#)

Methods

PDDocPrintPages (Only available with PDF Library SDK)

PDFLPrintDoc (Only available with PDF Library SDK)

PSPrintAfterTrailer

```
ACCB1 void ACCB2 PSPrintAfterTrailer (PDDoc doc, ASStm stm,
void* clientData);
```

Description

This notification is broadcast after the DSC trailer is emitted (immediately after writing **%%Trailer**) during the printing of a page to a PostScript printer with the methods **PDFLPrintDoc** (only available with PDF Library SDK) or **PDDocPrintPages** (only available with PDF Library SDK). At this point it is possible to resolve comments (at end) and emit cleanup code.

Parameters

doc	The document that is being printed.
stm	The PostScript print stream. PostScript commands can be added to the print stream using ASStmWrite .
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PSPrintBeforeEndSetup](#)

Methods

PDDocPrintPages (Only available with PDF Library SDK)
PDFLPrintDoc (Only available with PDF Library SDK)

PSPrintBeforeAcrobatProcsets

```
ACCB1 void ACCB2 PSPrintBeforeAcrobatProcsets (PDDoc doc,  
ASStm stm, void* clientData);
```

Description

Clients that register for this notification will be called back during PostScript printing—just prior to emission of the Acrobat procsets.

Parameters

doc	The document that is being printed.
stm	The PostScript print stream. PostScript commands can be added to the print stream using ASStmWrite .
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PSPrintBeforeEndSetup](#)

Methods

PDDocPrintPages (Only available with PDF Library SDK)

PDFLPrintDoc (Only available with PDF Library SDK)

PSPrintBeforeEndComments

```
ACCB1 void ACCB2 PSPrintBeforeEndComments (PDDoc doc,  
ASStm stm, void* clientData);
```

Description

This notification is broadcast after the DSC page-level comments that apply to all pages have been emitted (immediately before writing **%%EndComments**) during the printing of a page to a PostScript printer with the methods **PDFLPrintDoc** (only available with PDF Library SDK) or **PDDocPrintPages** (only available with PDF Library SDK).

Parameters

doc	The document that is being printed.
stm	The PostScript print stream. PostScript commands can be added to the print stream using ASStmWrite .
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PSPrintAfterBeginSetup](#)

[PSPrintBeforeEndSetup](#)

Methods

PDDocPrintPages (Only available with PDF Library SDK)

PDFLPrintDoc (Only available with PDF Library SDK)

PSPrintBeforeEndSetup

```
ACCB1 void ACCB2 PSPrintBeforeEndSetup (PDDoc doc, ASStm stm,
void* clientData);
```

Description

This notification is broadcast before the end of *Document Setup* (immediately before writing **%%EndSetup**) during the printing of a page to a PostScript printer with the methods **PDFLPrintDoc** (only available with PDF Library SDK) or **PDDocPrintPages** (only available with PDF Library SDK). At this point all of the job level resources and procsets have been added to the print stream.

Parameters

doc	The document that is being printed.
stm	The PostScript print stream. PostScript commands can be added to the print stream using ASStmWrite . Some printer drivers (Windows PS4) may not allow additions to the PostScript stream at this point.
clientData	Pointer to a block of user-supplied data that was passed in by the calling application when this notification was registered for using AVAppRegisterNotification .

Related Notifications

[PSPrintAfterBeginSetup](#)

Methods

PDDocPrintPages (Only available with PDF Library SDK)

PDFLPrintDoc (Only available with PDF Library SDK)

Errors

Error Systems

ErrSysAcroView	AcroView
ErrSysASFile	ASFile I/O
ErrSysCos	CosStore, filters
ErrSysCosSyntax	Cos syntax
ErrSysFontSvr	Font Server
ErrSysMDSystem	Platform-specific system
ErrSysMDApp	Platform-specific application
ErrSysNone	General error and out of memory
ErrSysPage	Page parsing and RIPping
ErrSysPDDoc	PDDoc and family, Page tree, bookmarks
ErrSysPDSEdit	PDFEdit
ErrSysPDMetadata	PDMetadata
ErrSysPDModel	Global PD
ErrSysPDPage	PDPage and family, thumbs, annots
ErrSysPDSEdit	PDSEdit
ErrSysRaster	Rasterizer
ErrSysXtnMgr	Extension manager.

Severities

ErrAlways	Always display error, even if others are suppressed.
ErrSilent	Never display a message.
ErrSuppressable	Display a message if the user has not suppressed errors.

ErrWarning	Display a warning.
ErrNoError	No error occurred.

ErrSysAcroView

System: **ErrSysAcroView**—AcroView (AV) level errors

Severity: **ErrAlways**

Platforms: All

avErrActionExternal	This action cannot be performed from within an external window.
avErrActionFullScreen	This action cannot be performed during full-screen mode.
avErrActionRestricted	This action can not be performed.
avErrBadActionCopy	No AVActionCopyProc is registered in an action handler and an action copy request occurred.
avErrBadAnnotationCopy	No AVAnnotHandlerCopyProc registered in an annotation handler and an annotation copy request occurred.
avErrCantOpenDialog	Acrobat can not open this file. There is a modal dialog open.
avErrCantOpenMoreThanTenDocs	No more than ten documents can be opened at a time.
avErrCantOpenPrinting	Acrobat can not open this file while printing another document.
avErrNoError	No error.
avErrNoText	There is no text.
avErrPrintJobTooBig	There are too many pages to print.
avErrTooManyChars	There is too much text to display. Cannot display more than 32,000 characters.

ErrSysASFile

Error System: **ErrSysASFile**—**ASFile** I/O errors

Severity: **ErrAlways**

Platforms: All

fileErrAlreadyOpen	This file is already open or in use by another application.
fileErrBusy	This file is busy and cannot be deleted.
fileErrBytesNotReady	Bytes requested by the viewer have not yet arrived. Occurs only over slow file systems.
fileErrDirFull	The directory is full.
fileErrDiskFull	The document's disk or the disk used for temporary files is full.
fileErrEOF	End of file was reached unexpectedly.
fileErrExists	Another file already exists under the same name.
fileErrFNF	This file cannot be found.
fileErrGeneral	A file error has occurred.
fileErrIO	A file I/O error has occurred.
fileErrIOTimeout	A file I/O error has occurred. The file connection timed out.
fileErrLocked	This file is locked.
fileErrNoErr	No error.
fileErrNSV	The disk containing this file is not available.
fileErrOpenFailed	File open failed.
fileErrPerm	You do not have access to this file.
fileErrRead	A file read error has occurred.
fileErrUserRequestedStop	User requested stop.
fileErrVLocked	This disk is locked and cannot be written to.
fileErrWrite	A file write error has occurred.
fileErrWrPerm	You do not have permission to write to this file.

ErrSysCos

Error System: **ErrSysCos**—CosStore, filter errors

Severities: **ErrSuppressable**, **ErrAlways**

Platforms: All

cosErrAfterSave	Implementation failure: this document is now invalid.
cosErrArrayBounds	Array out-of-bounds error.
cosErrBadFilterName	A stream specifies an unknown filter.
cosErrBadIndex	Bad master index.
cosErrBadSyntax	Syntax error.
cosErrCancelSave	A Save operation was canceled.
cosErrCantOpenTempFile	A temporary file could not be opened.
cosErrCCFError	Error in CCITT fax data filter.
cosErrDCTError	Error in JPEG data filter.
cosErrDictKeyNotName	Dictionary key must be a name object.
cosErrDocTableFull	Cos document table full.
cosErrEncryptionErr	Error in encryption filter. This error occurs if the: ● RC4 encryption code fails to initialize. ● RC4 encryption code fails to update its state when requested. ● Document's security handler has been changed since the document was last saved, and the length of the cryptData passed internally is too small to hold the new cryptData.
cosErrExpectedArray	Expected an array object.
cosErrExpectedBoolean	Expected an ASBool object.
cosErrExpectedDict	Expected a dictionary object.
cosErrExpectedDirect	Expected a direct object.
cosErrExpectedName	Expected a name object.

cosErrExpectedNull	Expected a NULL object.
cosErrExpectedNumber	Expected a number object.
cosErrExpectedStream	Expected a stream object.
cosErrExpectedString	Expected a string object.
cosErrInt16OutOfRange	A number is out of range.
cosErrInvalidAssignment	This direct object already has a container.
cosErrInvalidObj	Desired operation cannot be performed on this object.
cosErrListOverflow	Operation or data is too complex.
cosErrLZWError	Error in LZW data filter.
cosErrNeedFullSave	This file must be saved with a full save.
cosErrNeedRebuild	The file needs to be repaired.
cosErrNoError	No error.
cosErrOldLinFormat	Obsolete format: treating file as non-linearized.
cosErrReadError	Read error.
cosErrRebuildFailed	Could not repair file.
cosErrStreamTooShort	Stream source is shorter than specified length.
cosErrTempFileFull	The temporary file is full or nearly full. Close or save any modified documents.
cosErrTempTooShort	Temporary file unexpectedly short.
cosErrWriteError	Write error.

ErrSysCosSyntax

Error System: **ErrSysCosSyntax**—Cos syntax errors

Severity: **ErrSuppressable**

Platforms: All

cosSynErrBadArrayDict	Expected dictionary or array.
cosSynErrBadCharInHexString	Non-hex character in a hex string.
cosSynErrBadDict	Error reading dictionary.
cosSynErrBadFRef	Bad foreign object reference.
cosSynErrBadLinearized	Error reading linearized hint data.
cosSynErrBadObject	Error reading object.
cosSynErrBadObjectLabel	Object label badly formatted. This error occurs when the Acrobat viewer tries to read a Cos object in a file. It occurs if the object's: ● Object number is not an integer. ● Object number does not match the object number the viewer expected. ● Generation number is not an integer. ● Generation number does not match the generation number the viewer expected. ● Object/generation number line does not end with obj. One common cause of this error is incorrect offsets in the xref table. For example, if the offset to object 16 is one byte too high, Acrobat sees it as being object number 6—instead of 16—and raises this exception.
cosSynErrBadTrailerStart	Trailer dictionary start missing '<<'.
cosSynErrBadXref	Missing 'xref'.
cosSynErrBadXrefEntry	Error reading xref entry.
cosSynErrBadXrefHeader	Xref header should be two integers.
cosSynErrExtraEndStream	Unexpected endstream.
cosSynErrImageNeverEnded	End of image not found.

<code>cosSynErrNoEndStream</code>	Missing endstream.
<code>cosSynErrNoEOF</code>	Missing '%%EOF'.
<code>cosSynErrNoError</code>	No error.
<code>cosSynErrNoHeader</code>	File does not begin with '%PDF-'.
<code>cosSynErrNoStartAddress</code>	Value of startxref address not an integer.
<code>cosSynErrNoStartXRef</code>	Could not find startxref address.
<code>cosSynErrPStackUnderflow</code>	Parse stack underflow while reading object.
<code>cosSynErrStringTooLong</code>	String too long.
<code>cosSynErrTokenTooLong</code>	Token too long.
<code>cosSynErrUnexpectedArray</code>	Unexpected end of array.
<code>cosSynErrUnexpectedDict</code>	Unexpected end of dictionary.
<code>cosSynErrUnexpectedType</code>	Unexpected token type.
<code>cosSynErrUnknownName</code>	Unrecognized object name. This error occurs if the Acrobat viewer is reading a Cos object and encounters a token that is not a number, name, or string, and which is not one of the keywords: true, false, null, [,], <<, >>, stream, ID, endobj, startxref, endstream , and R . This error also occurs if the Acrobat viewer encounters one of these keywords in an unexpected place, for example a] without a preceding [.
<code>cosSynErrUnknownTokenType</code>	Unrecognized token type.
<code>cosSynErrUnterminatedString</code>	Un-terminated string.

ErrSysFontSvr

Error System: **ErrSysFontSvr**—Font server errors

Severity: **ErrAlways**

Platforms: All

fsErrBadParameter	Bad parameter passed to font server.
fsErrDownloadAborted	Font download aborted.
fsErrDownloadFailed	Font download failed.
fsErrInitFailed	Initialization of the font server module failed.
fsErrNeedNewATM	A new version of Adobe Type Manager is required.
fsErrNoATM	Adobe Type Manager was not found.
fsErrNoError	No error.
fsErrNoMMFonts	No multiple master fonts were found.
fsErrNoTlZapfDingbats	The Type 1 font 'Zapf Dingbats' must be installed.

ErrSysMDSysystem

MacOS

System: **ErrSysMDSysystem**—MacOS-specific system errors

Severity: **ErrAlways**

Platforms: MacOS

cfMacfBsyErr	This file is busy and cannot be deleted. (fBsyErr)
cfMacdirFulErr	The directory is full. (dirFulErr).
cfMacdskFulErr	The document's disk or the disk used for temporary files is full. (dskFulErr).
cfMacdsMemFullErr	Out of memory (-26). (dsMemFullErr)
cfMacdupFNErr	Another file already exists under the same name. (dupFNErr)
cfMaceofErr	End of file was reached unexpectedly. (eofErr)
cfMacfLckdErr	This file is locked. (fLckdErr)
cfMacGenPSErr	A Postscript error has occurred (-8133).
cfMaciIOAbort	An I/O error has occurred (-27). (iIOAbort)
cfMacioErr	A file I/O error has occurred. (ioErr)
cfMaciPrSavPFil	Error saving print file (-1).
cfMacmemFullErr	Out of memory (-108). (memFullErr)
cfMacNoErr	No error. (noErr).
cfMacnoMacDskErr	This disk is not a Macintosh disk. (noMacDskErr)
cfMacnsvErr	There is no such volume available. (nsvErr).
cfMacopWrErr	This file is already open or in use by another application. (opWrErr)
cfMacpermErr	You do not have permission to open this file. (permErr)

cfMacresNotFound	Tried to get a nonexistent resource (-192). (resNotFound)
cfMacServerLostConnection	This file's server connection has closed down. (-1070) (aspParamErr)
cfMacvLckdErr	This volume is locked and cannot be written to. (vLckdErr)
cfMacvolOffLinErr	This file's volume is not available. (volOffLinErr)
cfMacwrPermErr	You do not have permission to write to this file. (wrPermErr)

Windows

Error System: **ErrSysMDSystem**—Windows-specific system errors

Severity: **ErrAlways**

Platforms: Windows

WinAccessErr	Access denied.
WinBadDiskErr	Badly formatted disk.
WinBadDriveErr	Invalid drive.
WinBadFileErr	The file does not exist.
WinBadHdlErr	Bad file handle.
WinBadPathErr	The path does not exist.
WinDeviceErr	Device does not exist.
WinExistsErr	File already exists.
WinGeneralErr	General failure.
WinLockErr	Lock violation.
WinMemErr	Not enough memory.
WinNotDosErr	Not an MS-DOS disk.
WinShareErr	Sharing violation.
WinTooManyErr	Too many open files.
WinWrPermErr	You do not have write permission.



ErrSysMDApp

Error System: **ErrSysMDApp**—MacOS-specific application errors
Severity: **ErrAlways**
Platforms: MacOS

mdAppCantPrintToPDFWriter	Cannot print to Acrobat PDFWriter.
mdAppErrNoError	No error.
mdAppNoDAsWhilePrint	Close all Desk Accessories before printing.
mdAppNoPrinter	Printing is not possible until you have chosen a Printer using the Chooser.

ErrSysNone

Error System: **ErrSysNone**—General system and out-of-memory errors

Severity: **ErrAlways**

Platforms: All

genErrBadParm	Bad parameter.
genErrBadUnlock	Attempt to release an unlocked object.
genErrExceptionStackOverflow	Exception stack overflow.
genErrGeneral	An internal error occurred.
genErrListOverflow	Operation or data is too complex.
genErrMethodNotImplemented	Attempted to call a method that has not been implemented.
genErrNoError	No error.
genErrNoMemory	Out of memory.
genErrResourceLoadFailed	Failed to load an application resource (internal error).

ErrSysPage

Error System: **ErrSysPage**—MacOS-specific application errors

Severities: **ErrSuppressable**, **ErrSilent**

Platforms: All

pageErrArrayLenWrong	Array length is out of range.
pageErrBadColorSpace	Invalid colorspace.
pageErrBadContents	The page contents object has the wrong type.
pageErrBadDecodeArray	Bad decode array.
pageErrBadEGS	Invalid extended graphics state.
pageErrBadEPSColorSpace	An image uses a color space that will not separate correctly in some applications.
pageErrBadForm	Invalid form.
pageErrBadFunction	Invalid function resource.
pageErrBadPattern	Invalid pattern.
pageErrBadType3Font	Invalid Type 3 font.
pageErrBadTypeInXTextArray	Bad object type within a text operator array.
pageErrColorOutOfRange	A color value is out-of-range.
pageErrColorSpaceNotFound	Could not find the specified color space.
pageErrEGStateNotFound	Could not find the specified extended graphics state.
pageErrErrorReadingPage	There was an error reading page near the contents.
pageErrFontNotSet	Font has not been set.
pageErrFormNotFound	Could not find the form named '%s'.
pageErrErrorParsingImage	There was an error while trying to parse an image.
pageErrExpectedEndOfColor	Expected end of color space.

<code>pageErrExpectedHexOrASC85</code>	Expected ASCIIHexadecimal or ASCII85 string.
<code>pageErrImageExpectedEI</code>	Expected 'EI' while parsing an image.
<code>pageErrImageExpectedNumber</code>	Expected a number while parsing an image.
<code>pageErrFontNotInResDict</code>	Could not find a font in the Resources dictionary—using Helvetica instead.
<code>pageErrFontNotInResources</code>	A font is not in the Resources dictionary.
<code>pageErrFormTypeNotAvailable</code>	Specified form type is not supported.
<code>pageErrIllegalOpInPath</code>	Illegal operation inside a path.
<code>pageErrIllegalOpInTextObj</code>	Illegal operation inside a text object.
<code>pageErrIllegalOpInTextOutline</code>	Illegal operator inside a text outline object.
<code>pageErrIllegalTextOp</code>	Illegal operation outside text object.
<code>pageErrImageTooBig</code>	Image in form, Type 3 font, or pattern is too big.
<code>pageErrInvalidDash</code>	Dash arguments are invalid.
<code>pageErrInvalidGRestore</code>	Invalid restore.
<code>pageErrInvalidImageMaskDepth</code>	An image is specified as an image mask with more than 1 bit per pixel.
<code>pageErrMissingKey</code>	Dictionary is missing the key '%s'
<code>pageErrMissingResource</code>	A resource is missing.
<code>pageErrNoError</code>	No error.
<code>pageErrNumberOutOfRange</code>	A number value is out of range.
<code>pageErrParseContextError</code>	Error while parsing a form, Type 3 font, or pattern.
<code>pageErrReadLessImageData</code>	Read less image data than expected.
<code>pageErrSeveralParsingErrors</code>	There were several parsing errors on this page.
<code>pageErrTooFewPathOps</code>	Too few operands in path.
<code>pageErrOperandTooLarge</code>	An operand is too large.

<code>pageErrOpTooLarge</code>	Operand too large.
<code>pageErrPatternNotFound</code>	Could not find the pattern with the specified name.
<code>pageErrPatternTypeNotAvailable</code>	Pattern type is not supported.
<code>pageErrReadLessImageColor</code>	Read less image color data than expected.
<code>pageErrReadLessImageData</code>	Read less image data than expected.
<code>pageErrRecursiveMachine</code>	Internal error—machine called recursively.
<code>pageErrTokenTypeNotRec</code>	Token type not recognized.
<code>pageErrTooFewArgs</code>	There were too few arguments.
<code>pageErrTooFewOps</code>	Too few operands.
<code>pageErrTooManyArgs</code>	There were too many arguments.
<code>pageErrUnexpectedOpInDisplay</code>	Found an unexpected operator in the display list.
<code>pageErrUnknownColorSpace</code>	Unknown color space.
<code>pageErrUnknownFilterName</code>	Unknown filter name.
<code>pageErrUnknownXObjectType</code>	Unknown XObject type.
<code>pageErrUnrecognizedToken</code>	An unrecognized token '%s' was found.
<code>pageErrWrongArgsForSetColor</code>	Wrong number of arguments for a setcolor operator.
<code>pageErrWrongNumOpsInCurve</code>	A curve operator has the wrong number of operands.
<code>pageErrWrongOperand</code>	Wrong operand type—expected type '%s'.
<code>pageErrWrongOpType</code>	Wrong operand type.
<code>pageErrXObjectNotFound</code>	Could not find the XObject named '%s'.

ErrSysPDDoc

Error System: **ErrSysPDDoc**—**PDDoc** and family, page tree, bookmark errors

Severities: **ErrSuppressable**, **ErrAlways**

Platforms: All

pdErrAbortNotes	Creation of the notes file was canceled.
pdErrAfterSave	This document was successfully saved, but an error occurred after saving the document. Please close and reopen the document.
pdErrAlreadyOpen	This file is already open.
pdErrATMMemory	ATM is running out of memory. Text in font '%s' may not render properly.
pdErrBadAction	Invalid action object.
pdErrBadAnnotation	Invalid annotation object.
pdErrBadAnnotColor	Invalid annotation color (only RGB colors are allowed).
pdErrBadBaseObj	The base pages object is missing or invalid.
pdErrBadBead	Invalid article element.
pdErrBadBookmark	Invalid bookmark object.
pdErrBadCMap	The font contains a bad CMap /Encoding.
pdErrBadFileSpec	Invalid file specification object.
pdErrBadFont	Bad font object or font descriptor object.
pdErrBadFontBBox	The font '%s' contains a bad /BBox.
pdErrBadFontDescMetrics	The font '%s' contains bad /FontDescriptor metrics.
pdErrBadFontFlags	The font '%s' contains bad /Flags.
pdErrBadFontWidths	The font '%s' contains bad /Widths.
pdErrBadOutlineObj	The outlines object is missing or invalid.
pdErrBadPageObj	A page object is missing or invalid.

pdErrBadPageTree	The document's page tree contains an invalid node.
pdErrBadResMetrics	Invalid or corrupt font metrics in the resource file.
pdErrBadRootObj	The root object is missing or invalid.
pdErrBadThread	Invalid article object.
pdErrBookmarksError	There is an error in the bookmarks.
pdErrCancelSave	A Save operation was canceled.
pdErrCannotBeBlankPage	The page number cannot be left blank.
pdErrCannotDeleteAllPages	You cannot delete all pages. At least one page must remain.
pdErrCannotMergeWithSubsetFonts	Documents contain subset fonts that have the same name and cannot be merged.
pdErrCannotOpenMoreBkMark	Cannot open more bookmarks.
pdErrCannotOpenNotes	An error occurred while creating the document notes file.
pdErrCannotReopenDoc	This document was successfully saved, but an error occurred after saving the document. Close and reopen the document.
pdErrCantUseNewVersion	This file contains information not understood by the viewer. It cannot be used for this operation.
pdErrCopyPageFailed	The copy of a page failed.
pdErrEmbeddingFont	Error while trying to embed a font.
pdErrExceedEncryptionLength	Supported encryption key length exceeded.
pdErrExceedEncryptionVersion	Encryption version not supported by current header.
pdErrHostEncodingNotSet	The application has not set the host encoding.
pdErrInvalidPageNumber	"%s" is an invalid page number.

pdErrIsFileLocked	Unable to open file for writing. It may be locked or unavailable.
pdErrMissingGlyphs	Unable to find a substitution font with all the characters used by the font named "%s". Some characters may not be displayed or printed.
pdErrMissingSubsetFont	A font required for font substitution is missing.
pdErrNeedCryptHandler	Cannot execute this command on an unsecured document.
pdErrNeedJapanese	An error has occurred that may be fixed by installing the latest version of the Japanese Language Support package.
pdErrNeedKorean	An error has occurred that may be fixed by installing the latest version of the Korean Language Support package.
pdErrNeedPassword	This document requires a password.
pdErrNeedRebuild	This file is damaged.
pdErrNeedSimpChinese	An error has occurred that may be fixed by installing the latest version of the Traditional Chinese Language Support package.
pdErrNeedTradChinese	An error has occurred that may be fixed by installing the latest version of the Traditional Chinese Language Support package.
pdErrNoCryptHandler	The security client required by this command is unavailable (that is, the necessary security handler is not registered).
pdErrNoError	No error.
pdErrNoNotes	This document has no notes.
pdErrNoPDDocForCosDoc	There is no PDDoc associated with this CosDoc .
pdErrNotEnoughMemoryToOpenDoc	There is not enough memory available to open the document.

pdErrNotEnoughSpaceForTempFile	There is not enough temporary disk space for this operation.
pdErrNotValidPage	There is no page numbered “%s” in this document.
pdErrOldATMVersion	The font ‘%s’ cannot be displayed with the installed version of ATM.
pdErrOldEncryption	This viewer cannot decrypt this document.
pdErrOpNotPermitted	This operation is not permitted. Acrobat security does not allow content copying or extraction. These operations should be permitted when the document’s master password is used.
pdErrOptMemory	There is not enough memory to optimize this file.
pdErrPagesLockedNotDeleted	One or more pages are in use and could not be deleted.
pdErrRequireTrustedMode	Only Adobe-certified Acrobat clients are allowed while viewing this document.
pdErrStartLessThanEnd	The starting page number must be less than or the same as the ending page number.
pdErrTextStringTooShort	There are not enough bytes in text string for multibyte character code.
pdErrThreadProcessing	Error while processing articles.
pdErrThumbError	Error while processing thumbnail.
pdErrTooManyPagesForInsert	Inserting this file would result in a document with too many pages.
pdErrTooManyPagesForOpen	This file cannot be opened because it contains too many pages.
pdErrTrySaveAs	This file can only be saved using Save As.
pdErrUnableToCloseDueToRefs	Unable to close document due to outstanding references.
pdErrUnableToExtractFont	Unable to extract the embedded font ‘%s’. Some characters may not display or print correctly.

pdErrUnableToExtractFontErr	Unable to extract embedded font.
pdErrUnableToFindFont	Unable to find or create the font '%s'. Some characters may not display or print correctly.
pdErrUnableToOpenDoc	This file could not be opened.
pdErrUnableToRead	Unable to read file.
pdErrUnableToRecover	Unable to recover original file.
pdErrUnableToRenameTemp	Unable to rename temporary file to Save As name.
pdErrUnableToWrite	Unable to write file.
pdErrUnableToXlateText	Some text in the font and character '%s' could not be displayed or printed correctly. The font could not be re-encoded.
pdErrUnknownAction	Unknown action type.
pdErrUnknownFileType	This is not a valid Portable Document File (PDF) document. It cannot be opened.
pdErrUnknownProcsets	Information needed to print a page is unavailable.
pdErrWhileRecoverInsertPages	There was an error while inserting or extracting pages and another error while trying to recover.
pdErrZeroPageFile	This file cannot be opened because it has no pages.



ErrSysPDFEdit

Error System: **ErrSysPDFEdit**—PDFEdit errors

Severity: **ErrAlways**

Platforms: All

peErrBadBlockHeader	Bad block header for Type 1 embedded stream.
peErrCantCreateFontSubset	Unable to create embedded font subset.
peErrCantEmbedFont	This font is licensed and cannot be embedded.
peErrCantGetAttrs	Unable to get attributes for font.
peErrCantGetImageDict	Unable to get image dictionary.
peErrCantGetWidths	Unable to get widths for font.
peErrCantReadImage	Unable to read image data.
peErrFontToEmbedNotOnSys	Unable to find font to embed on this system.
peErrNoError	No error.
peErrPStackUnderflow	PDFEdit parse stack underflow while reading object.
peErrUnknownPDEColorSpace	Unknown PDEColorSpace value.
peErrUnknownResType	Unknown PDEObject resource type.
peErrWrongPDEObjectType	Incorrect PDEObject type.

ErrSysPDMetadata

Error System: **ErrSysPDMetadata**—PD metadata errors

Severity: **ErrAlways**

Platforms: All

pdMetadataErrBadXAP	The XMP metadata supplied was not syntactically correct RDF.
pdMetadataErrBadPDF	XMP metadata processing found a syntax error in PDF.
pdMetadataErrCouldntCreateMetaXAP	Could not create internal representation of XMP metadata.
pdMetadataErrInternalError	An internal error occurred while processing XMP metadata.



ErrSysPDModel

Error System: **ErrSysPDModel**—Global PD-level errors

Severity: **ErrAlways**

Platforms: All

pdModErrDuplicateCryptName	A security handler is already registered with this name.
pdModErrEncTablesFailed	Unable to initialize font encoding tables.
pdModErrNoError	No error.

ErrSysPDPage

Error System: **ErrSysPDDoc**—**PDPage** and family, thumbnail, annotation errors

Severities: **ErrSuppressable**, **ErrAlways**, **ErrSilent**

Platforms: All

pdPErrBadType3Font	Invalid Type 3 font.
pdPErrFormTooComplex	Form is too complex to print.
pdPErrNoError	No error.
pdPErrPageDimOutOfRange	The dimensions of this page are out-of-range.
pdPErrType3TooComplex	Type 3 font is too complex to print.
pdPErrUnableToCreateRasterPort	Creation of a raster port failed.

ErrSysPDSEdit

Error System: **ErrSysPDSEdit**—PDSEdit errors

Severity: **ErrAlways**

Platforms: All

pdsErrAlreadyExists	There is already a table entry with the same name.
pdsErrBadPDF	An incorrect structure was found in the PDF file.
pdsErrCantDo	Some software required to perform this operation is not present in this version of Adobe Acrobat.
pdsErrRequiredMissing	A required field was missing from a dictionary.
pdsErrWrongTypeEntry	Dictionary entry has wrong Cos type.
pdsErrWrongTypeParameter	Wrong type parameter supplied to a PDS procedure.

ErrSysRaster

Error System: **ErrSysRaster**—Rasterizer errors

Severity: **ErrAlways**

Platforms: All

rasErrCreatePort	Creation of rasterizer port failed.
rasErrDraw	A rasterizer error occurred.
rasErrInitFailed	Initialization of the rasterizer module failed.
rasErrNoError	No error.

ErrSysXtnMgr

Error System: **ErrSysXtnMgr**—Extension manager errors. Errors registered by clients (see [ASRegisterErrorString](#)) are automatically assigned to this error system.

Severity: **ErrAlways**

Platforms: All (some system-specific as noted)

xmErr68KOnly	Only the 68K Viewer can load this client (MacOS only).
xmErrCalledObsoleteProc	An obsolete function was called.
xmErrCannotReplaceSelector	Attempt to replace an selector that cannot be replaced.
xmErrDuplicatePluginName	Two clients are attempting to register with the same name.
xmErrInitializationFailed	The client failed to initialize.
xmErrNoError	No error.
xmErrNoPLUGResource	The client lacks a PLUG resource of ID 1 (MacOS only)
xmErrNotPrivileged	The Adobe Reader cannot load this client.
xmErrOutOfDateHFT	The client was compiled with an out-of-date HFT .
xmErrPluginIncompatible	The client is incompatible with this version of the Acrobat viewer.
xmErrPluginLoadFailed	The client failed to load.
xmErrPPCOnly	Only the PowerPC Viewer can load this client (MacOS only).



Macros

A4_GLOBALS

A4_GLOBALS

Description

(Macintosh only) Together with **A5_GLOBALS**, this specifies the address register off of which a client references its globals. They are used only for 68xxx Macintosh clients, not for PowerPC clients. Either **A4_GLOBALS** or **A5_GLOBALS** must be defined to be 1, and the other undefined.

Both **A4_GLOBALS** and **A5_GLOBALS** are set automatically by **PIPrefix.h**.

Header File

PIPrefix.h

Related Macros

[A5_GLOBALS](#)



A5_GLOBALS

A5_GLOBALS

Description

(Macintosh only) Together with **A4_GLOBALS**, **this** specifies the address register off of which a client references its globals. They are used only for 68xxx Macintosh clients, not for PowerPC clients. Either **A4_GLOBALS** or **A5_GLOBALS** must be defined to be 1, and the other undefined.

Both **A4_GLOBALS** and **A5_GLOBALS** are set automatically by **PIPrefix.h**.

Header File

PIPrefix.h

Related Macros

[A4_GLOBALS](#)



ACCB1

ACCB1

Description

Macro used when declaring callback functions. Its definition is platform-dependent. Use this macro in every callback function you declare.

Use **ACCB1** before the return value in a function declaration, as shown in the Example.

Header File

MacPlatform.h

WINPLTFM.H

Related Macros

[ACCB2](#)

[ACCBPROTO1](#)

[ACCBPROTO2](#)

Example

```
static ACCB1 ASAtom ACCB2 SnapZoomToolGetType(AVTool tool) {  
    ...  
}
```



ACCB2

ACCB2

Description

Macro used when declaring callback functions. Its definition is platform-dependent. Use this macro in every callback function you declare.

Use **ACCB2** after the return value in a function declaration, as shown in the Example.

Header File

MacPlatform.h

WINPLTFM.H

Related Macros

[ACCB1](#)

[ACCBPROTO1](#)

[ACCBPROTO2](#)

Example

```
static ACCB1 ASAtom ACCB2 SnapZoomToolGetType(AVTool tool) {  
    ...  
}
```




ACCBPROTO1

ACCBPROTO1

Description

Macro used when declaring function prototypes. Its definition is platform-dependent. Use this macro in every function prototype you declare.

Use **ACCBPROTO1** before the return value in a function prototype, as shown in the Example.

Header File

MacPlatform.h
WINPLTFM.H

Related Macros

[ACCBPROTO2](#)
[ACCB1](#)
[ACCB2](#)

Example

```
static ACCBPROTO1 void (ACCBPROTO2  
*DrawImageSelectionCallback) (AVPageView pageView, AVRect* updateRect,  
void *data);
```



ACCBPROTO2

ACCBPROTO2

Description

Macro used when declaring function prototypes. Its definition is platform-dependent. Use this macro in every function prototype you declare.

Use **ACCBPROTO2** after the return value in a function prototype, as shown in the Example.

Header File

MacPlatform.h
WINPLTFM.H

Related Macros

[ACCB1](#)
[ACCB2](#)
[ACCBPROTO1](#)

Example

```
static ACCBPROTO1 void (ACCBPROTO2  
*DrawImageSelectionCallback) (AVPageView pageView, AVRect* updateRect,  
void *data);
```



ACROASSERT

`ACROASSERT`

Description

A platform-independent version of the ANSI assert function.

Header File

`acroassert.h`

ASFileSysCreatePathFromCString

ASFileSysCreatePathFromCString(asfs, cPath)

Description

Helper macro for the [ASFileSysCreatePathName](#) method. See this method for more information.

NOTE: This macro uses a local variable named **scratchFourBytes** — (**void* scratchFourBytes**). PDF Library users need to provide this variable in order to utilize the macro; the variable must be local to the client application, not to the library. Any client can use this macro provided he also has code similar to the following in the same source file that uses the macro:

```
static void* scratchFourBytes;
```

Parameters

asfs	(May be NULL) The file system through which the ASPathName is obtained.
cPath	A C string containing the path for which the ASPathName is obtained.

Header File

ASExpT.h

Related Macros

[ASFileSysCreatePathFromDIPath](#)
[ASFileSysCreatePathFromFSSpec](#)
[ASFileSysCreatePathWithFolderName](#)

ASFileSysCreatePathFromDIPath

```
ASFileSysCreatePathFromDIPath(asfs, cDIPath, aspRelativeTo)
```

Description

Helper macro for the [ASFileSysCreatePathName](#) method. See this method for more information.

Parameters

asfs	(May be NULL) The file system through which the ASPathName is obtained.
cDIPath	A C string containing the device-independent path for which the ASPathName is obtained.
aspRelativeTo	(May be NULL) An ASPathName that cDIPath will be evaluated against if it contains a relative path.

Header File

```
ASExpT.h
```

Related Macros

[ASFileSysCreatePathFromCString](#)
[ASFileSysCreatePathFromFSSpec](#)
[ASFileSysCreatePathWithFolderName](#)

ASFileSysCreatePathFromFSSpec

ASFileSysCreatePathFromFSSpec(asfs, cPath)

Description

Helper macro for the [ASFileSysCreatePathName](#) method. See this method for more information.

Parameters

asfs	(May be NULL) The file system through which the ASPathName is obtained.
cPath	The FSSpec for which the ASPathName is obtained.

Header File

ASExpT.h

Related Macros

[ASFileSysCreatePathFromCString](#)
[ASFileSysCreatePathFromDIPath](#)
[ASFileSysCreatePathWithFolderName](#)

ASFileSysCreatePathWithFolderName

ASFileSysCreatePathWithFolderName(asfs, aspFolder, cFileName)

Description

Helper macro for the [ASFileSysCreatePathName](#) method. See this method for more information.

Parameters

asfs	(May be NULL) The file system through which the ASPathName is obtained.
aspFolder	ASPathName contained the path of the folder.
cFileName	A C string containing the name of the file. The returned ASPathName contains the result of appending cFileName to aspFolder .

Header File

ASExpT.h

Related Macros

[ASFileSysCreatePathFromCString](#)
[ASFileSysCreatePathFromDIPath](#)
[ASFileSysCreatePathFromFSSpec](#)



ASFixedRoundToInt16

ASFixedRoundToInt16 (f)

Description

Converts a fixed point number to an [ASInt16](#), rounding the result.

Parameters

f	The fixed point number to convert.

Header File

ASExpT.h

Related Macros

[Fixed Numbers](#)
[ASFixedRoundToInt32](#)
[ASFixedToFloat](#)
[ASFixedTruncToInt16](#)
[ASFixedTruncToInt32](#)
[ASFloatToFixed](#)
[ASInt16ToFixed](#)
[ASInt32ToFixed](#)

ASFixedRoundToInt32

ASFixedRoundToInt32 (f)

Description

Converts a fixed point number to an [ASInt32](#), rounding the result.

Parameters

f	The fixed point number to convert.

Header File

ASExpT.h

Related Macros

- Fixed Numbers
- [ASFixedRoundToInt16](#)
- [ASFixedToFloat](#)
- [ASFixedTruncToInt16](#)
- [ASFixedTruncToInt32](#)
- [ASFloatToFixed](#)
- [ASInt16ToFixed](#)
- [ASInt32ToFixed](#)



ASFixedToFloat

ASFixedToFloat (f)

Description

Converts a fixed point number to a floating point number of type **double**.

NOTE: The result of this macro is not a **float** type, as the name suggests. You can cast the result to a **float**.

Parameters

f	The fixed point number to convert.

Header File

ASExpT.h

Related Macros

[Fixed Numbers](#)

[ASFixedRoundToInt16](#)

[ASFixedRoundToInt32](#)

[ASFixedTruncToInt16](#)

[ASFixedTruncToInt32](#)

[ASFloatToFixed](#)

[ASInt16ToFixed](#)

[ASInt32ToFixed](#)



ASFixedTruncToInt16

ASFixedTruncToInt16 (f)

Description

Converts a fixed point number to an [ASInt16](#), truncating the result.

Parameters

f	The fixed point number to convert.

Header File

ASExpT.h

Related Macros

[Fixed Numbers](#)
[ASFixedRoundToInt16](#)
[ASFixedRoundToInt32](#)
[ASFixedToFloat](#)
[ASFixedTruncToInt32](#)
[ASFloatToFixed](#)
[ASInt16ToFixed](#)
[ASInt32ToFixed](#)



ASFixedTruncToInt32

ASFixedTruncToInt32 (f)

Description

Converts a fixed point number to an [ASInt32](#), truncating the result.

Parameters

f	The fixed point number to convert.

Header File

ASExpT.h

Related Macros

[Fixed Numbers](#)
[ASFixedRoundToInt16](#)
[ASFixedRoundToInt32](#)
[ASFixedToFloat](#)
[ASFixedTruncToInt16](#)
[ASFloatToFixed](#)
[ASInt16ToFixed](#)
[ASInt32ToFixed](#)



ASFloatToFixed

`ASFloatToFixed(f)`

Description

Converts a floating point number to a fixed point number.

Parameters

<code>f</code>	The floating point number to convert.

Header File

`ASExpT.h`

Related Macros

[Fixed Numbers](#)

[ASFixedRoundToInt16](#)

[ASFixedRoundToInt32](#)

[ASFixedToFloat](#)

[ASFixedTruncToInt16](#)

[ASFixedTruncToInt32](#)

[ASInt16ToFixed](#)

[ASInt32ToFixed](#)



ASInt16ToFixed

ASInt16ToFixed(i)

Description

Converts an [ASInt16](#) to a fixed point number.

Parameters

i	The ASInt16 to convert.

Header File

ASExpT.h

Related Macros

[Fixed Numbers](#)

[ASFixedRoundToInt16](#)

[ASFixedRoundToInt32](#)

[ASFixedToFloat](#)

[ASFixedTruncToInt16](#)

[ASFixedTruncToInt32](#)

[ASFloatToFixed](#)

[ASInt32ToFixed](#)

ASInt32ToFixed

ASInt32ToFixed(i)

Description

Converts an [ASInt32](#) to a fixed point number.

Parameters

i	The ASInt32 to convert.

Header File

ASExpT.h

Related Macros

- [Fixed Numbers](#)
- [ASFixedRoundToInt16](#)
- [ASFixedRoundToInt32](#)
- [ASFixedToFloat](#)
- [ASFixedTruncToInt16](#)
- [ASFixedTruncToInt32](#)
- [ASFloatToFixed](#)
- [ASFixedToFloat](#)
- [ASInt16ToFixed](#)

CALL_REPLACED_PROC

```
CALL_REPLACED_PROC(hft, sel, replacer) (args...)
```

Description

Calls the previous implementation of a replaced method (that is, the code that would have been executed before the method was replaced using [REPLACE](#)).

Parameters

hft	The HFT containing the replaced method to execute, for example, gAcroViewHFT . See HFT Values for a list of the Acrobat viewer's built-in HFTs .
sel	The selector for the replaced method to execute. The name must have the characters SEL appended, for example, AVAlertSEL .
replacer	The callback whose previous implementation is called. Recall that a method may be replaced more than once, and all replacements for a particular method are kept in a linked list. proc must be an element in that linked list, and the entry before proc is the one that is called.
args...	The argument list to pass to the procedure being called.

Header File

```
ASCalls.h
```

Related Macros

[REPLACE](#)

Related Methods

[HFTGetReplacedEntry](#)

Example

```
CALL_REPLACED_PROC(gAcroViewHFT, AVAlertSEL, myAlertCallback) (iconType,  
gsm, button1, button2, button3, beep);
```




CastToPDAnnot

CastToPDAnnot (a)

Description

Casts a link annotation or a text annotation to a generic annotation.

Parameters

a	The link annotation or text annotation to cast.

Header File

PDExpT.h

Related Macros

[CastToPDLinkAnnot](#)

[CastToPDTextAnnot](#)



CastToPDLinkAnnot

CastToPDLinkAnnot (a)

Description

Casts a generic annotation or a text annotation to a link annotation.

Parameters

a	The generic annotation or text annotation to cast.

Header File

PDExpT.h

Related Macros

[CastToPDAnnot](#)

[CastToPDTextAnnot](#)



CastToPDTextAnnot

CastToPDTextAnnot (a)

Description

Casts a link annotation or a generic annotation to a text annotation.

Parameters

a	The link annotation or generic annotation to cast.

Header File

PDExpT.h

Related Macros

[CastToPDAnnot](#)

[CastToPDLinkAnnot](#)

DEBUG

DEBUG

Description

Enables and disables compile-time type-checking in various declarations.

Define DEBUG as 1 to enable type-checking (when developing and testing clients) and as 0 to disable type-checking (before shipping your client).

Header File

MacPlatform.h

WINPLTFM.H

Related Macros

[ASCallbackCreateNotification](#)

[ASCallbackCreateProto](#)

[ASCallbackCreateReplacement](#)

Example

```
#define DEBUG 1
```



DURING

DURING

Description

Begins the section of code where Acrobat APIs may throw an exception. After calling an Acrobat API method, execution may jump into the **HANDLER** clause. This macro is similar to the **TRY** clause in the C++ language.

Header File

CorCalls.h

Related Macros

[END_HANDLER](#)

[E_RETURN](#)

[E_RTRN_VOID](#)

[HANDLER](#)



END_HANDLER

END_HANDLER

Description

Ends a DURING/HANDLER/END_HANDLER clause.

Header File

CorCalls.h

Related Macros

[DURING](#)

[E_RETURN](#)

[E_RTRN_VOID](#)

[HANDLER](#)



E_RETURN

`E_RETURN`

Description

Returns a value from an enclosing function when nested inside a DURING/HANDLER clause.

Header File

`CorCalls.h`

Related Macros

[DURING](#)

[END_HANDLER](#)

[E_RTRN_VOID](#)

[HANDLER](#)

ERRORCODE

ERRORCODE

Description

Macro defined to call [ASGetExceptionErrorCode](#). Returns an [ASInt32](#) containing exception error code. See [Errors](#) for a list of predefined exceptions.

Header File

CorCalls.h

Related Macros

[DURING](#)
[END_HANDLER](#)
[E_RETURN](#)
[HANDLER](#)

E_RTRN_VOID

E_RTRN_VOID

Description

Returns from an enclosing function when nested inside a DURING/HANDLER clause, without returning a value.

Header File

CorCalls.h

Related Macros

[ErrGetCode](#)

[ErrGetSeverity](#)

[ErrGetSignedCode](#)

[ErrGetSystem](#)

ErrBuildCode

```
ErrBuildCode(xseverity, xsys, xerror)
```

Description

Builds an error code for the specified severity, system, and error number. Error codes are used in exception handling. The [ASRaise](#) method takes an error code for its argument; [ASRegisterErrorString](#) returns an error code.

An error code has three components:

- Severity: none, warning, severe; 4 bits
- System: Cos, PDDoc, and so on; 8 bits
- Error: FileOpen, Syntax, and so on; 16 bits

Parameters

xseverity	Severity of the error. One of Severities .
xsys	Error system. Must be one of Error Systems .
xerror	Error number in the error system, xsys .

Header File

```
AcroErr.h
```

Related Macros

[ErrGetCode](#)
[ErrGetSeverity](#)
[ErrGetSignedCode](#)
[ErrGetSystem](#)

Related Methods

[ASRaise](#)
[ASRegisterErrorString](#)

Example

```
myErrorCode = ErrBuildCode(ErrAlways, ErrSysAcroView,  
    avErrActionRestricted);
```



ErrGetCode

ErrGetCode (xcode)

Description

Gets the error number from an error code.

Parameters

xcode	Error code.

Header File

AcroErr.h

Related Macros

[ErrBuildCode](#)
[ErrGetSeverity](#)
[ErrGetSignedCode](#)
[ErrGetSystem](#)

Related Methods

[ASGetErrorString](#)
[ASGetExceptionErrorCode](#)
[ASRegisterErrorString](#)

Example

```
errorNumber = ErrGetCode (errorCode) ;
```



ErrGetSeverity

`ErrGetSeverity(xcode)`

Description

Gets the error severity from an error code. Returns one of [Severities](#).

Parameters

xcode	Error code.

Header File

`AcroErr.h`

Related Macros

[ErrBuildCode](#)
[ErrGetCode](#)
[ErrGetSignedCode](#)
[ErrGetSystem](#)

Related Methods

[ASGetErrorString](#)
[ASGetExceptionErrorCode](#)
[ASRegisterErrorString](#)

Example

```
errorSeverity = ErrGetSeverity(errorCode);
```



ErrGetSignedCode

ErrGetSignedCode (xcode)

Description

Gets a signed error number from an error code.

Parameters

xcode	Error code.

Header File

AcroErr.h

Related Macros

[ErrBuildCode](#)
[ErrGetCode](#)
[ErrGetSeverity](#)
[ErrGetSystem](#)

Related Methods

[ASGetErrorString](#)
[ASGetExceptionErrorCode](#)
[ASRegisterErrorString](#)

Example

```
errorSignedNumber = ErrGetSignedCode (errorCode) ;
```



ErrGetSystem

ErrGetSystem(xcode)

Description

Gets the error system from an error code. Returns one of [Error Systems](#).

Parameters

xcode	Error code.

Header File

AcroErr.h

Related Macros

[ErrBuildCode](#)
[ErrGetCode](#)
[ErrGetSeverity](#)
[ErrGetSignedCode](#)

Related Methods

[ASGetErrorString](#)
[ASGetExceptionErrorCode](#)
[ASRegisterErrorString](#)

Example

```
errorSystem = ErrGetSystem(errorCode);
```

Fixed Numbers

Description

A variety of predefined fixed-point constants.

<code>fixedZero</code>	<code>fixedFive</code>
<code>fixedHundredth</code>	<code>fixedSix</code>
<code>fixedSixteenth</code>	<code>fixedSeven</code>
<code>fixedTenth</code>	<code>fixedEight</code>
<code>fixedEighth</code>	<code>fixedNine</code>
<code>fixedQuarter</code>	<code>fixedTen</code>
<code>fixedHalf</code>	<code>fixedEleven</code>
<code>fixedThreeQuarters</code>	<code>fixedTwelve</code>
<code>fixedPi4</code>	<code>fixedSixteen</code>
<code>fixedSevenEighths</code>	<code>fixedThirtyTwo</code>
<code>fixedOne1</code>	<code>fixedFifty</code>
<code>fixedOne</code>	<code>fixedSeventyTwo</code>
<code>fixedOneAndQuarter</code>	<code>fixedNinety</code>
<code>fixedFourThirds</code>	<code>fixedHundred</code>
<code>fixedSqrtTwo</code>	<code>fixedHundredFifty</code>
<code>fixedThreeHalves</code>	<code>fixedOneEighty</code>
<code>fixedOneAnd3Qtr</code>	<code>fixedTwoSeventy</code>
<code>fixedPi2</code>	<code>fixedFiveHundred</code>
<code>fixedGolden</code>	<code>fixedThousand</code>
<code>fixedTwo</code>	<code>fixedTenThousand</code>
<code>fixedThree</code>	<code>fixedNegativeInfinity</code>
<code>fixedFour</code>	<code>fixedPositiveInfinity</code>

Header File

`ASExpT.h`

**Related Macros**

[ASFixedTruncToInt16](#)
[ASFixedTruncToInt32](#)
[ASFixedRoundToInt16](#)
[ASFixedRoundToInt32](#)
[ASFixedToFloat](#)
[ASFloatToFixed](#)
[ASInt16ToFixed](#)
[ASInt32ToFixed](#)



HANDLER

HANDLER

Description

Follows a DURING macro. Code inserted between HANDLER and END_HANDLER macros will be executed only if an Acrobat function or other function THROWS. This macro is similar to the CATCH clause in the C++ language.

Header File

CorCalls.h

Related Macros

[DURING](#)

[END_HANDLER](#)

[E_RETURN](#)

[E_RTRN_VOID](#)

MAC_PLATFORM

MAC_PLATFORM

Description

(Macintosh only, previously known as MAC_ENV) Defined if the client is being compiled for a Macintosh, undefined otherwise. [MAC_PLATFORM](#), [WIN_PLATFORM](#) and [UNIX_PLATFORM](#) should be used by client developers to conditionally compile platform-dependent code.

MAC_PLATFORM is automatically set by the header files.

Header File

Environ.h (based on a value set in MacPlatform.h)

Related Macros

[UNIX_PLATFORM](#)

[WIN_PLATFORM](#)



MAC68K

MAC68K

Description

(Macintosh only) Together with [POWER_PC](#), specifies which processor architecture a client is targeted for. Either **POWER_PC** or **MAC68K** must be defined as 1, the other as 0.

POWER_PC and **MAC68K** are automatically set by **PIPrefix.h**.

Header File

PIPrefix.h

Related Macros

[POWER_PC](#)

PI_ACROSUPPORT_VERSION

PI_ACROSUPPORT_VERSION

Description

Specifies the version of the Acrobat support level **HFT**.

This is automatically set by **PIRequir.h**.

Header File

PIRequir.h

Errors

If the **HFT** version is higher than the viewer loading the client supports, it displays an alert box with the message "There was an error while loading the client '<plug-in name>'. The client is incompatible with this version of the Viewer."

Related Macros

[PI_ACROVIEW_VERSION](#)

[PI_CORE_VERSION](#)

[PI_COS_VERSION](#)

[PI_MACINTOSH_VERSION](#)

[PI_PDMODEL_VERSION](#)

[PI_UNIX_VERSION](#)

[PI_WIN_VERSION](#)

PI_ACROVIEW_VERSION

PI_ACROVIEW_VERSION

Description

Specifies the version of the Acrobat viewer level **HFT**.

This is automatically set by **PIRequir.h**.

Header File

PIRequir.h

Errors

If the **HFT** version is higher than the viewer loading the client supports, it displays an alert box with the message "There was an error while loading the client '<plug-in name>'. The client is incompatible with this version of the Viewer."

Related Macros

[PI_ACROSUPPORT_VERSION](#)

[PI_CORE_VERSION](#)

[PI_COS_VERSION](#)

[PI_MACINTOSH_VERSION](#)

[PI_PDMODEL_VERSION](#)

[PI_UNIX_VERSION](#)

[PI_WIN_VERSION](#)

PI_CORE_VERSION

PI_CORE_VERSION

Description

Specifies the version of the **HFT**.

This is automatically set by **PIRequir.h**.

Header File

PIRequir.h

Errors

If the **HFT** version is higher than the viewer loading the client supports, it displays an alert box with the message "There was an error while loading the client '<plug-in name>'. The client is incompatible with this version of the viewer."

Related Macros

[PI_ACROSUPPORT_VERSION](#)

[PI_ACROVIEW_VERSION](#)

[PI_COS_VERSION](#)

[PI_MACINTOSH_VERSION](#)

[PI_PDMODEL_VERSION](#)

[PI_UNIX_VERSION](#)

[PI_WIN_VERSION](#)

PI_COS_VERSION

PI_COS_VERSION

Description

Specifies the version of the Cos-level **HFT**.

This is automatically set by **PIRequir.h**.

Header File

PIRequir.h

Errors

If the **HFT** version is higher than the viewer loading the client supports, it displays an alert box with the message "There was an error while loading the client '<plug-in name>'. The client is incompatible with this version of the Viewer."

Related Macros

[PI_ACROSUPPORT_VERSION](#)

[PI_ACROVIEW_VERSION](#)

[PI_CORE_VERSION](#)

[PI_MACINTOSH_VERSION](#)

[PI_PDMODEL_VERSION](#)

[PI_UNIX_VERSION](#)

[PI_WIN_VERSION](#)

PI_MACINTOSH_VERSION

PI_MACINTOSH_VERSION

Description

Specifies the version of the Macintosh-only methods **HFT**.

This is automatically set by **PIRequir.h**.

Header File

PIRequir.h

Errors

If the **HFT** version is higher than the viewer loading the client supports, it displays an alert box with the message "There was an error while loading the client '<plug-in name>'. The client is incompatible with this version of the Viewer."

Related Macros

[PI_ACROSUPPORT_VERSION](#)

[PI_ACROVIEW_VERSION](#)

[PI_CORE_VERSION](#)

[PI_COS_VERSION](#)

[PI_PDMODEL_VERSION](#)

[PI_UNIX_VERSION](#)

[PI_WIN_VERSION](#)

PI_PDMODEL_VERSION

PI_PDMODEL_VERSION

Description

Specifies the version of the PD level **HFT**.

This is automatically set by **PIRequir.h**.

Header File

PIRequir.h

Errors

If the **HFT** version is higher than the viewer loading the client supports, it displays an alert box with the message "There was an error while loading the client '<plug-in name>'. The client is incompatible with this version of the Viewer."

Related Macros

[PI_ACROSUPPORT_VERSION](#)

[PI_ACROVIEW_VERSION](#)

[PI_CORE_VERSION](#)

[PI_COS_VERSION](#)

[PI_MACINTOSH_VERSION](#)

[PI_UNIX_VERSION](#)

[PI_WIN_VERSION](#)

PI_UNIX_VERSION

PI_UNIX_VERSION

Description

Specifies the version of the UNIX-only methods **HFT**.

This is automatically set by **PIRequir.h**.

Header File

PIRequir.h

Errors

If the **HFT** version is higher than the viewer loading the client supports, it displays an alert box with the message "There was an error while loading the client '<plug-in name>'. The client is incompatible with this version of the Viewer."

Related Macros

[PI_ACROSUPPORT_VERSION](#)

[PI_ACROVIEW_VERSION](#)

[PI_CORE_VERSION](#)

[PI_COS_VERSION](#)

[PI_MACINTOSH_VERSION](#)

[PI_PDMODEL_VERSION](#)

[PI_WIN_VERSION](#)

PI_WIN_VERSION

PI_WIN_VERSION

Description

Specifies the version of the Windows-only methods **HFT**.

This is automatically set by **PIRequir.h**.

Header File

PIRequir.h

Errors

If the **HFT** version is higher than the viewer loading the client supports, it displays an alert box with the message "There was an error while loading the client '<plug-in name>'. The client is incompatible with this version of the Viewer."

Related Macros

[PI_ACROSUPPORT_VERSION](#)

[PI_ACROVIEW_VERSION](#)

[PI_CORE_VERSION](#)

[PI_COS_VERSION](#)

[PI_MACINTOSH_VERSION](#)

[PI_PDMODEL_VERSION](#)

[PI_UNIX_VERSION](#)



PLATFORM

PLATFORM

Description

Defines the platform-specific header file. Must be "**MacPlatform.h**" in Mac OS, "**WINPLTFM.H**" in Windows.

PLATFORM is automatically set by the header file.

Header File

PIPrefix.h (*Macintosh*)
ENVIRON.H (*Windows*)



POWER_PC

POWER_PC

Description

(*Macintosh only*) Together with [MAC68K](#), specifies which processor architecture a client is targeted for. Either **POWER_PC** or **MAC68K** must be defined as 1, the other as 0.

POWER_PC and **MAC68K** are automatically set by **PIPrefix.h**.

Header File

PIPrefix.h

Related Macros

[MAC68K](#)



PRODUCT

PRODUCT

Description

Defines the platform-specific header file. Must be “**Plugin.h**” in Mac OS and Windows.

PRODUCT is automatically set by the header file.

Header File

PIPrefix.h (**Macintosh**)

ENVIRON.H (**Windows**)



REGISTER_NOTIFICATION

```
REGISTER_NOTIFICATION(nselName, proc, rock)
```

Description

Macro to register a notification. Uses [AVAppRegisterNotification](#).

Header File

PICommon.h

REPLACE

```
REPLACE(hft, sel, proc)
```

Description

Replaces one of the Acrobat viewer's built-in methods. The method being replaced must be one of the [Replaceable Methods](#). The method's **HFTEntryReplaceable** flag is automatically set, allowing it to be subsequently replaced.

All clients, and the Acrobat viewer itself, share a single copy of each **HFT**. As a result, when a client replaces the implementation of a method, all other clients and the Acrobat viewer also use the new implementation of that method. In addition, once a method's implementation has been replaced, there is no way to remove the new implementation without restarting the Acrobat viewer.

NOTE: The [CALL_REPLACED_PROC](#) macro is available to call the previous HFT entry function that was replaced.

Parameters

hft	The HFT containing the method to replace, for example, gAcroViewHFT . See HFT Values for a list of the Acrobat viewer's built-in HFTs .
sel	The selector for the method to replace. The name must have the characters SEL appended, for example, AVAlertSEL .
proc	A callback containing the replacement method. The callback must have been created using ASCallbackCreateReplacement .

Header File

```
PICommon.h
```

Related Macros

[ASCallbackCreateReplacement](#)
[CALL_REPLACED_PROC](#)

Related Methods

[HFTReplaceEntry](#)

Examples

```
myAlertCallback = ASCallbackCreateReplacement(AVAlertSEL, myAlert);  
REPLACE(gAcroViewHFT, AVAlertSEL, myAlertCallback);
```




RERAISE

RERAISE

Description

Re-raises the most recently raised exception and passes it to the next exception handler in the stack.

Parameters

None

Header File

CorCalls.h

Related Macros

[ErrGetCode](#)

[ErrGetSeverity](#)

[ErrGetSignedCode](#)

[ErrGetSystem](#)

Related Methods

[ASRaise](#)

UNIX_PLATFORM

UNIX_PLATFORM

Description

(UNIX only) Defined if the client is being compiled for a UNIX platform, undefined otherwise. [MAC_PLATFORM](#), [WIN_PLATFORM](#), and [UNIX_PLATFORM](#) should be used by client developers to conditionally compile platform-dependent code.

UNIX_PLATFORM must be defined in the arguments to the C compiler. The make files for the sample clients in the Acrobat SDK do this automatically.

Header File

Environ.h (based on a value set in UNIXPlatform.h)

Related Macros

[MAC_PLATFORM](#)

[WIN_PLATFORM](#)

WIN_PLATFORM

WIN_PLATFORM

Description

(Windows only, previously known as WIN_ENV) Defined if the client is being compiled for a Windows machine, undefined otherwise. [MAC_PLATFORM](#), [WIN_PLATFORM](#), and [UNIX_PLATFORM](#) should be used by client developers to conditionally compile platform-dependent code.

WIN_PLATFORM must be defined in the arguments to the C compiler. The make files for the sample clients in the Acrobat SDK do this automatically.

Header File

Environ.h (based on a value set in WinPltfrm.h)

Related Macros

[MAC_PLATFORM](#)

[UNIX_PLATFORM](#)



API Changes

This chapter summarizes changes to the API in the following versions:

- [Version 2.1](#)
- [Version 3.0](#)
- [Version 3.01](#)
- [Adobe PDF Library, Version 1.0](#)
- [Version 4](#)
- [Version 5](#)
- [Version 6](#)

Version 2.1

Methods

The following methods were added in version 2.1.

Available only if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to 0x00020001 or higher.

[AVDocGetClientName](#)

[AVDocGetPageText](#)

[AVDocSetClientName](#)

Available only if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to 0x00020001 or higher.

[PDDocClearFlags](#)

[PDDrawCosObjToWindow](#)

[PDPageNotifyContentsDidChangeEx](#)

Errors

The following errors were added in version 2.1.

[xmErrPluginLoadFailed](#)

[xmErrNotPrivileged](#)

[xmErr68KOnly](#)

[xmErrPPCOnly](#)

[avErrNoText](#)

Notifications

The following notifications were added in version 2.1.

[PDPageContentsDidChangeEx](#)

Version 3.0

Objects

The following object was added in version 3.0.

[PDTrans](#)

Methods

The following methods were added in version 3.0.

Available only if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to 0x00020002 or higher.

[ASFileFromMDFile](#)
[ASFileGetFileSysByName](#)
[ASFileGetMDFile](#)
[ASFilePushData](#)
[ASFileRegisterFileSys](#)
[ASFileSetMode](#)
[ASFileSysAcquireFileSysPath](#)
[ASFileSysCreatePathName](#)
[ASFileUnregisterFileSys](#)

Available only if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to 0x00020002 or higher.

[AVAppEnumTransHandlers](#)
[AVAppGetTransHandlerByType](#)
[AVAppHandlePlatformEvent](#)
[AVAppRegisterTransHandler](#)
[AVAuthOpen](#)
[AVDocDoActionPropsDialog](#)
[AVDocDoSaveAs](#)
[AVDocGetViewDef](#)
[AVDocIsExternal](#)
[AVDocOpenFromASFileWithParamString](#)
[AVDocPrintPagesWithParams](#)
[AVDocSendAuxData](#)
[AVDocSetDead](#)
[AVDocSetViewDef](#)
[AVHasAuxDataHandler](#)
[AVPageViewDrawCosObj](#)
[AVPageViewGetFirstVisiblePageNum](#)

AVPageViewGetLastVisiblePageNum
 AVPageViewGetLayoutMode
 AVPageViewGetNextView
 AVPageViewGetSelectedAnnotPageNum
 AVPageViewHighlightText
 AVPageViewInsetRect
 AVPageViewInvalidateText
 AVPageViewPageNumIsVisible
 AVPageViewPointInText
 AVPageViewSetLayoutMode
 AVPageViewTrackText
 AVPageViewSetPageNum
 AVPageViewUseThisDestination
 AVRegisterAuxDataHandler
 AVToolButtonSetExternal
 AVToolButtonSetHelpText
 AVUnregisterAuxDataHandler
 AVWindowGetCursorAtPoint
 AVWindowHandlePlatformEvent

Available only if `PI_MACINTOSH_VERSION` (in `PIRequir.h`) is set to 0x00020002 or higher.

AVAppEnumSystemFonts

Available only if `PI_COS_VERSION` (in `PIRequir.h`) is set to 0x00020002 or higher.

CosDocClose
 CosDocCreate
 CosDocOpenWithParams
 CosDocSaveToFile
 CosDocSetDirty

Available only if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to 0x00020002 or higher.

PDDocGetCryptHandlerClientData
 PDDocGetFullScreen
 PDDocOpenEx
 PDDocOpenFromASFile
 PDDocOpenFromASFileEx
 PDDocReadAhead
 PDDocSaveWithParams
 PDDocSetFullScreen
 PDFileSpecGetDoc
 PDFileSpecGetFileSysName
 PDNameTreeLookup
 PDPageGetDuration
 PDPageGetTransition
 PDPageHasTransition

PDPageSetDuration
 PDPageSetTransition
 PDPageStmGetInlineImage
 PDPageStmGetToken
 PDRegisterCryptHandlerEx
 PDRegisterFileSpecHandlerByName
 PDTransEqual
 PDTransFromCosObj
 PDTransGetCosObj
 PDTransGetDuration
 PDTransGetSubtype
 PDTransNew
 PDTransNewFromCosDoc
 PDTransNull
 PDTransIsValid
 PDViewDestResolve

Available only if `PI_WIN_VERSION` (in `PIRequir.h`) is set to 0x00020000 or higher.

WinAppEnableIdleTimer
 WinAppGetModalParent
 WinAppGetPalette
 WinAppRegisterModelessDialog
 WinAppUnRegisterModelessDialog

The following methods were added to the Acrobat Toolkit.

ASGetErrorString
 ASfree
 ASmalloc
 ASrealloc
 CosDictPut
 CosDictRemove
 CosNewString
 PDDocAuthorize
 PDDocCreate
 PDDocInsertPages
 PDDocSaveWithParams
 PDFontAcquireXlateTable
 PDFontXlateTableRelease
 PDFontXlateString
 PDFontXlateWidths
 PDPageGetDoc
 PDRegisterCryptHandler
 PDXlateToPDFDocEnc

The following methods are now replaceable.

AVDocDoSaveAs
 AVPageViewGetNextView

PDDocSave
PDDocSaveWithParams

Callbacks

The following callbacks were added in version 3.0.

ASFileCompletionProc
ASFileSysAcquireFileSysPathProc
ASFileSysAsyncAbortProc
ASFileSysAsyncReadProc
ASFileSysAsyncWriteProc
ASFileSysClearOutstandingMReadsProc
ASFileSysCreatePathNameProc
ASFileSysGetFileFlags
ASFileSysGetStatusProc
ASFileSysMReadRequestProc
ASFileSysYieldProc
ASIODoneProc
ASMemFreeProc
AVAnnotHandlerCursorEnterProc
AVAnnotHandlerCursorExitProc
AVAuxDataPerformProc
AVTransHandlerCompleteTransDictProc
AVTransHandlerDoPropertiesProc
AVTransHandlerEnumProc
AVTransHandlerExecuteProc
AVTransHandlerGetButtonTextProc
AVTransHandlerGetInstructionsProc
AVTransHandlerGetStringOneTextProc
AVTransHandlerGetStringTwoTextProc
AVTransHandlerGetTypeProc
AVTransHandlerGetItemUINameProc
AVTransHandlerGetUINameProc
AVTransHandlerInitTransDictProc
PDAuthProcEx
PDCryptFreeAuthDataProc
PDCryptFreeSecurityDataProc
PDLaunchActionProc
PDPageStmImageDataProc
PDPageStmStringOverflowProc

Data

The following data structures were added in version 3.0.

ASFile Flags
ASFileMode (ASFILE_SERIAL and ASFILE_LOCAL)
ASFileStatus Flags
ASFileSysRec (significantly expanded)

[ASIORequest](#)
[ASPlatformPrinterSpec](#)
[AVAnnotHandler](#) (new callbacks)
[AVAuxDataHandler](#)
[AVDocOpenParams](#) (significantly expanded)
[AVDocPrintParams](#)
[AVDocViewDef](#)
[AVPrefsType](#) (significantly expanded)
[AVSystemFont](#)
[AVSystemFont Flags](#)
[AVTransHandler](#)
[AVTransitionPort](#)
[CosDocOpenParams](#)
[CosDocSaveFlags](#)
[CosDocSaveParams](#)
[Emit Font Options](#)
[Emit Flags](#)
[Page Specification](#)
[PDCryptHandler](#) (new callbacks)
[PDDocReadAhead Flags](#)
[PDLayoutMode](#)
[PDPageStmToken](#)
[Predefined Cursors](#) (cursors added)
[Tool Button Flags](#)
[Transition Duration](#)

Errors

The following error system was added in version 3.0.

[ErrSysXtnMgr](#)

The following errors were added in version 3.0.

[cfMacGenPSErr](#)
[cfMaciPrSavPFil](#)
[cfMacServerLostConnection](#)
[cosErrBadIndex](#)
[cosErrCancelSave](#)
[cosErrOldLinFormat](#)
[cosErrTempTooShort](#)
[cosSynErrBadLinearized](#)
[fileErrBytesNotReady](#)
[fileErrUserRequestedStop](#)
[fileErrIOTimeout](#)
[fsErrBadParameter](#)
[fsErrDownloadAborted](#)
[fsErrDownloadFailed](#)
[pageErrBadColorSpace](#)
[pageErrBadEGS](#)

[pageErrBadPattern](#)
[pageErrEGStateNotFound](#)
[pageErrMissingKey](#)
[pageErrMissingResource](#)
[pageErrPatternNotFound](#)
[pageErrPatternTypeNotAvailable](#)
[pdErrATMMemory](#)
[pdErrBadCMap](#)
[pdErrCancelSave](#)
[pdErrCannotReopenDoc](#)
[pdErrOldATMVersion](#)
[pdErrOptMemory](#)
[pdErrTextStringTooShort](#)
[pdErrZeroPageFile](#)

Notifications

The following notifications were added in version 3.0.

Available only if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to 0x00020002 or higher.

[AVDocDidAddToSelection](#)
[AVDocWantsToDie](#)
[AVDocWillOpenFromPDDoc](#)
[PDDocWillPrintDoc](#)
[PDDocWillSaveEx](#)

Miscellaneous

Values of some types in methods, callbacks, and data have been changed for improved cross-platform portability.

Previous name	New name
<code>boolean</code>	ASBool
<code>Int8</code>	ASInt8
<code>Int16</code>	ASInt16
<code>Int32</code>	ASInt32
<code>Uns8</code>	ASUns8
<code>Uns16</code>	ASUns16
<code>Uns32</code>	ASUns32



Version 3.0J

Methods

The following methods were added in version 3.0J.

Available only if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to 0x00020003 or higher.

`PDGetHostEncoding`
`PDHostMBLen`
`PDXlateToHostEx`
`PDXlateToPDFDocEncEx`
`PDDocCreateWordFinderUCS`
`PDFontGetCIDSystemInfo`
`PDFontGetCIDSystemSupplement`
`PDFontGetDescendant`
`PDFontGetEncodingName`
`PDFontXlateToHost`
`PDFontXlateToUCS`

The following methods have new behavior in version 3.0J.

`PDWordFilterString`
`PDWordGetString`
`PDWordSplitString`



Version 3.01

Methods

The following methods were added in version 3.01.

Available only if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to 0x00020003 or higher.

[AVDestInfoDestroy](#)
[AVDocCopyAction](#)
[AVDocCopyActionCommon](#)
[AVDocCopyAdditionalActions](#)
[AVDocCopyAnnot](#)
[AVDocCopyAnnotCommon](#)
[AVDocDoCopyAs](#)
[AVPageViewDrawCosObjEx](#)
[AVPageViewToDestInfo](#)
[AVPageViewUseDestInfo](#)

Callbacks

The following callbacks were added in version 3.01.

[AVActionCopyProc](#)
[AVAnnotHandlerCopyProc](#)
[CrossDocLinkWithDestProc](#)

Data

The following data structures were added or changed in version 3.01.

[AVAnnotHandler](#) (new flag)
[AVDestInfo](#)
[Emit Font Options](#) (obsoleted and new flags)
[ExternalDocServerCreationData](#) (new callback and data)

Errors

The following errors were added in version 3.01.

[avErrBadActionCopy](#)
[avErrBadAnnotationCopy](#)

Adobe PDF Library, Version 1.0

All API elements listed here are available only if the Adobe PDF Library Version is 1.0 or later. This version adds the PDFEdit methods.

Objects

The following objects were added for PDFEdit.

[PDEClip](#)
[PDEColorSpace](#)
[PDEContainer](#)
[PDEContent](#)
[PDEElement](#)
[PDEExtGState](#)
[PDEFont](#)
[PDEForm](#)
[PDEImage](#)
[PDEObject](#)
[PDEPath](#)
[PDEPattern](#)
[PDEPlace](#)
[PDEText](#)
[PDEXObject](#)
[PDSysFont](#)

Methods

The following methods were added for the Adobe PDF Library.

[PDDocPrintPages](#)
[PDFLInit](#)
[PDFLPrintDoc](#)
[PDFLTerm](#)
[AVExtensionMgrRegisterNotification](#)
[AVExtensionMgrUnregisterNotification](#)
[PDSetHostEncoding](#)

The following methods were added for PDFEdit.

[PDELogDump](#)
[PDEObjectDump](#)
[PDEAttrEnumTable](#)
[PDEDefaultGState](#)
[PDEEnumElements](#)
[PDEMergeResourcesDict](#)
[PDEClipAddElem](#)
[PDEClipCreate](#)
[PDEClipGetElem](#)
[PDEClipGetNumElems](#)

PDEClipRemoveElems
PDEColorSpaceCreate
PDEColorSpaceCreateFromName
PDEColorSpaceGetBase
PDEColorSpaceGetBaseNumComps
PDEColorSpaceGetCosObj
PDEColorSpaceGetCTable
PDEColorSpaceGetHiVal
PDEColorSpaceGetName
PDEColorSpaceGetNumComps
PDEContainerCreate
PDEContainerGetContent
PDEContainerGetDict
PDEContainerGetMCTag
PDEContainerSetContent
PDEContainerSetDict
PDEContainerSetMCTag
PDEContentAddElem
PDEContentCreate
PDEContentCreateFromCosObj
PDEContentGetAttrs
PDEContentGetElem
PDEContentGetNumElems
PDEContentGetResources
PDEContentRemoveElem
PDEContentToCosObj
PDEElementCopy
PDEElementGetBBox
PDEElementGetClip
PDEElementGetGState
PDEElementGetMatrix
PDEElementSetGState
PDEElementSetMatrix
PDEExtGStateCreate
PDEExtGStateGetCosObj
PDEFontCreate
PDEFontCreateFromCosObj
PDEFontCreateFromSysFont
PDEFontCreateWithParams
PDEFontGetCosObj
PDEFontGetNumCodeBytes
PDEFontSubsetNow
PDFindSysFontForPDEFont
PDEShadingCreateFromCosObj
PDEFormGetContent
PDEFormGetCosObj
PDEImageCreate

PDEImageCreateFromCosObj
PDEImageDataIsEncoded
PDEImageGetAttrs
PDEImageGetColorSpace
PDEImageGetCosObj
PDEImageGetData
PDEImageGetDataLen
PDEImageGetDataStm
PDEImageGetFilterArray
PDEImageIsCosObj
PDEImageSetData
PDEImageSetDataStm
PDEAcquire
PDEAddTag
PDEGetTag
PDEObjectGetType
PDERelease
PDERemoveTag
PDEPathAddSegment
PDEPathCreate
PDEPathGetData
PDEPathGetPaintOp
PDEPathSetData
PDEPathSetPaintOp
PDEPatternCreate
PDEPatternGetCosObj
PDEPlaceCreate
PDEPlaceGetDict
PDEPlaceGetMCTag
PDEPlaceSetDict
PDEPlaceSetMCTag
PDETextAdd
PDETextCreate
PDETextGetAdvanceWidth
PDETextGetBBox
PDETextGetFont
PDETextGetGState
PDETextGetNumBytes
PDETextGetNumRuns
PDETextGetQuad
PDETextGetRunForChar
PDETextGetStrokeMatrix
PDETextGetText
PDETextGetTextMatrix
PDETextGetTextState
PDETextIsAtPoint
PDETextReplaceChars

PDTextRunGetCharOffset
 PDTextRunGetNumChars
 PDTextRunSetFont
 PDTextRunSetGState
 PDTextRunSetStrokeMatrix
 PDTextRunSetTextMatrix
 PDTextRunSetTextState
 PDTextSplitRunAt
 PDXObjectCreate
 PDXObjectGetCosObj
 PDEnumSysFonts
 PDFindSysFont
 PDPageAcquirePDEContent
 PDPageGetPDEContentFilters
 PDPageGetPDEContentFlags
 PDPagePDEContentWasChanged
 PDPageRegisterForPDEContentChanged
 PDPageRegisterForPDEContentNotCached
 PDPageReleasePDEContent
 PDPageSetPDEContent
 PDPageSetPDEContentFilters
 PDPageSetPDEContentFlags
 PDPageSuspendPDEContentChanged
 PDPageUnRegisterForPDEContentNotCached
 PDSysFontAcquirePlatformData
 PDSysFontGetEncoding
 PDSysFontGetInfo
 PDSysFontGetName
 PDSysFontGetType0Widths

Callbacks

The following callbacks were added for PDFEdit.

PDEAttrEnumProc
 PDEElementEnumProc
 PDEObjectDumpProc
 PDSysFontEnumProc

Data

The following data structures were added, mostly for PDFEdit.

PDFLData (for Adobe PDF Library)
 PDEColorSpec
 PDEColorValue
 PDEContentAttrs
 PDEContentFlags
 PDEContentToCosObjFlags
 PDEDash

PDEElementCopyFlags
PDEEnumElementsFlags
PDEFilterArray
PDEFilterSpec
PDEFontAttrs
PDEFontCreateFlags
PDEFontInfoP
PDEGraphicStateP
PDEGraphicStateWasSetFlags
PDEImageAttrFlags
PDEImageAttrs
PDEImageDataFlags
PDEPathElementType
PDEPathOpFlags
PDETextFlags
PDETextRenderMode
PDETextState
PDETextStateWasSetFlags
PDEType
PDSysFontMatchFlags

Notifications

The following notifications were added for PDFEdit.

PagePDEContentDidChange
PagePDEContentNotCached



Version 4

Objects

The following objects were added in version 4.0.

[ASExtension](#)
[PDEDeviceNColors](#)
[PDEGroup](#)
[PDEShading](#)
[PDEUnknown](#)
[PDNameTree](#)
[PDNumTree](#)
[PDPageLabel](#)
[PDSAttrObj](#)
[PDSCClassMap](#)
[PDSElement](#)
[PDSMC](#)
[PDSRoleMap](#)
[PDSTreeRoot](#)

Methods

The following methods were obsoleted in version 4.0.

[PDPageEnumContents](#)
[PDPageEnumResources](#)

The following methods were added in version 4.0.

Available only if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to 0x00040000 or higher.

[ASEnumExtensions](#)
[ASExtensionGetFileName](#)
[ASExtensionGetRegisteredName](#)
[ASFileStmWrOpen](#)
[ASProcStmWrOpen](#)
[HFTIsValid](#)

Available only if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to 0x00040000 or higher.

[AVAnnotHandlerDeleteInfo](#)
[AVAnnotHandlerGetInfo](#)
[AVAppHandlePlatformEvent](#)
[AVAppOpenHelpFile](#)
[AVDocAlert](#)
[AVDocAlertConfirm](#)
[AVDocAlertNote](#)
[AVDocAlertYesNo](#)
[AVDocDoCopyAs](#)

AVDocDoPrint
AVDocDoSaveAsWithParams
AVDocIsReadOnly
AVDocSelectionEnumPageRanges
AVDocSetReadOnly
AVMenuIsHiddenOnMenubar
AVMenubarAddHiddenMenu
AVPageViewAppearanceGetAVMatrix
AVPageViewDeviceRectToPageRZ
AVPageViewDoPopupMenu
AVPageViewDrawAnnotSequence
AVPageViewGetGrayRect
AVPageViewGetVisibleAnnotPage
AVPageViewInvertQuad
AVPageViewShowControl
AVPageViewTransformRectRZ
AVSysAllocTimeStringFromTimeRec
AVToolBarNewFlyout
AVToolButtonGetFlyout
AVToolButtonGetIcon
AVToolButtonGetMenu
AVToolButtonSetFlyout
AVToolButtonSetIcon
AVToolButtonSetMenu
AVWindowHandlePlatformEvent

Available only if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to 0x00040005 or higher.

AVAppRegisterForPageViewKeyDown
AVAppUnregisterForPageViewKeyDown

Available only if **PI_COS_VERSION** (in `PIRequir.h`) is set to 0x00040000 or higher.

CosArrayRemoveNth
CosDocEnumEOFs
CosDocEnumIndirect
CosDocGetObjByID
CosDocSaveWithParams
CosObjCopy
CosObjGetGeneration
CosObjGetID
CosObjHash
CosStringGetHexFlag
CosStringSetHexFlag

Available only if **PI_COS_VERSION** (in `PIRequir.h`) is set to 0x00040005 or higher.

[CosCryptGetVersion](#)
[CosDecryptGetMaxKeyBytes](#)
[CosEncryptGetMaxKeyBytes](#)

Available only if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to 0x00040000 or higher.

[PDDocCreateNameTree](#)
[PDDocCreateStructTreeRoot](#)
[PDDocEnumOCGs](#)
[PDDocExportNotes](#)
[PDDocFindPageNumForLabel](#)
[PDDocFromCosDoc](#)
[PDDocGetLabelForPageNum](#)
[PDDocGetNameTree](#)
[PDDocGetPageLabel](#)
[PDDocGetStructTreeRoot](#)
[PDDocImportCosDocNotes](#)
[PDDocImportNotes](#)
[PDDocOpenWithParams](#)
[PDDocReadAheadPages](#)
[PDDocRemoveNameTree](#)
[PDDocRemovePageLabel](#)
[PDDocRemoveStructTreeRoot](#)
[PDDocSetPageLabel](#)
[PDFontFromCosObj](#)
[PDGetAnnotHandlerByName](#)
[PDImageSelectAlternate](#)
[PDImageSelAdjustMatrix](#)
[PDImageSelGetDeviceAttr](#)
[PDImageSelGetGeoAttr](#)
[PDNameTreeEnum](#)
[PDNameTreeEqual](#)
[PDNameTreeFromCosObj](#)
[PDNameTreeGet](#)
[PDNameTreeGetCosObj](#)
[PDNameTreeIsValid](#)
[PDNameTreeNew](#)
[PDNameTreePut](#)
[PDNameTreeRemove](#)
[PDPageGetAnnotSequence](#)
[PDPageLabelEqual](#)
[PDPageLabelFromCosObj](#)
[PDPageLabelGetCosObj](#)
[PDPageLabelGetPrefix](#)
[PDPageLabelGetStart](#)
[PDPageLabelGetStyle](#)
[PDPageLabelIsValid](#)

[PDPageLabelNew](#) [PDRegisterAnnotHandler](#)

The following methods were added for the Adobe PDF Library.

ASAtomGetCount
ASPurgeMemory
PDFLPrintDoc

The following methods were added for Placed PDF, which is part of the Adobe PDF Library.

AGMCleanup
AGMClip
AGMClosePath
AGMConcat
AGMDeletePort
AGMDeleteRasterDev
AGMFill
AGMInit
AGMLineTo
AGMMoveTo
AGMNewBitmapPort
AGMNewPath
AGMNewRasterDev
AGMNewRasterPort
AGMNewWindowPort
AGMSetAntiAliasPolicy
AGMSetRGBColor
CCSetSystemCalibration
PDDocPrintPages
PDFontDownloadContextCreate
PDFontDownloadContextDestroy
PDFontStreamPS
PDFontWasExtracted
PDFontWasFauxed
PDFFreeMemory
PDPageDrawContents
PDPageDrawContentsPlaced
PDPageDrawContentsPlacedToWindow
PDPageEmitPSOrient

The following methods were added for PDFEdit.

Available only if **PI_PDFEDIT_READ_VERSION** (in **PIRequir.h**) is set to 0x00040000 or higher.

[PDEClipFlattenedEnumElems](#)
[PDEContentGetDefaultColorSpace](#)
[PDEDeviceNColorsGetColorValue](#)

PDEElementIsAtPoint
 PDEElementIsAtRect
 PDEFontGetNumCodeBytes
 PDEFontGetOneByteEncoding
 PDEFontIsMultiByte
 PDEFontSumWidths
 PDEGroupGetContent
 PDEImageGetDecodeArray
 PDETextGetNumBytes
 PDETextIsAtPoint
 PDETextIsAtRect

Available only if **PI_PDFEDIT_WRITE_VERSION** (in `PIRequir.h`) is set to 0x00040000 or higher.

PDEClipAddElem
 PDEClipCopy
 PDEClipCreate
 PDEColorSpaceCreate
 PDEDeviceNColorsCreate
 PDEFontCreateFromSysFontEx
 PDEFontCreateWithParams
 PDEGroupCreate
 PDEGroupSetContent
 PDEImageSetDecodeArray
 PDEPurgeCache
 PDEShadingCreateFromCosObj
 PDEUnknownGetOpName

Available only if **PI_PDSYSFONT_VERSION** (in `PIRequir.h`) is set to 0x00040000 or higher.

PDEmbedSysFontForPDEFont
 PDFindSysFontEx
 PDSysFontAcquirePlatformData
 PDSysFontGetCIDSystemInfo
 PDSysFontGetType0Widths
 PDSysFontGetWidthsEx
 PDSysFontReleasePlatformData

The following methods were added for PDS (Structure level).

Available only if **PI_PDS_READ_VERSION** (in `PIRequir.h`) is set to 0x00040000 or higher.

PDSAttrObjGetOwner
 PDSClassMapGetAttrObj
 PDSClassMapGetNumAttrObjs
 PDSElementGetAlt
 PDSElementGetAttrObj
 PDSElementGetClass

PDSElementGetFirstPage
PDSElementGetID
PDSElementGetKid
PDSElementGetNumAttrObjs
PDSElementGetNumClasses
PDSElementGetNumKids
PDSElementGetParent
PDSElementGetRevision
PDSElementGetStructTreeRoot
PDSElementGetTitle
PDSElementGetType
PDSMCGetInfo
PDSOBJGetParent
PDSRoleMapDoesMap
PDSRoleMapGetDirectMap
PDSTreeRootGetClassMap
PDSTreeRootGetElementFromID
PDSTreeRootGetKid
PDSTreeRootGetNumKids
PDSTreeRootGetRoleMap

Available only if `PI_PDS_WRITE_VERSION` (in `PIRequir.h`) is set to 0x00040000 or higher.

PDSAttrObjCreate
PDSAttrObjCreateFromStream
PDSClassMapAddAttrObj
PDSClassMapRemoveAttrObj
PDSClassMapRemoveClass
PDSElementAddAttrObj
PDSElementAddClass
PDSElementClearID
PDSElementCreate
PDSElementIncrementRevision
PDSElementInsertKid
PDSElementInsertMCAsKid
PDSElementInsertOBJAsKid
PDSElementRemoveAttrObj
PDSElementRemoveAllAttrObjs
PDSElementRemoveAllClasses
PDSElementRemoveClass
PDSElementRemoveKid
PDSElementRemoveKidMC
PDSElementReplaceKid
PDSElementReplaceKidMC
PDSElementReplaceKidOBJ
PDSElementSetAlt
PDSElementSetID

[PDSElementSetTitle](#)
[PDSElementSetType](#)
[PDSRoleMapCopy](#)
[PDSRoleMapMap](#)
[PDSRoleMapUnMapDst](#)
[PDSRoleMapUnMapSrc](#)
[PDSTreeRootCreateClassMap](#)
[PDSTreeRootCreateRoleMap](#)
[PDSTreeRootInsertKid](#)
[PDSTreeRootRemoveClassMap](#)
[PDSTreeRootRemoveKid](#)
[PDSTreeRootRemoveRoleMap](#)
[PDSTreeRootReplaceKid](#)

Available only if **PI_WIN_VERSION** (in `PIRequir.h`) is set to 0x00040000 or higher.

[WinAppGetPrinterHDC](#)

The following methods are now replaceable.

[AVDocDoPrint](#)
[AVDocDoSaveAsWithParams](#)
[AVDocPrintPages](#)
[AVDocPrintPagesWithParams](#)
[PDImageSelectAlternate](#)

Callbacks

The following callbacks were added in version 4.0.

[ASExtensionEnumProc](#)
[ASProcStmDestroyProc](#)
[ASStmProc](#)
[AVAnnotHandlerDeleteInfoProc](#)
[AVAnnotHandlerGetInfoProc](#)
[AVDocSelectionEnumPageRangesProc](#)
[AVDocSelectionGetSelectionTypeProc](#)
[AVPageViewKeyDownProc](#)
[AVSelectionPageRangeEnumProc](#)
[AVWindowDestroyPlatformThingProc](#)
[CosDocEnumEOFsProc](#)
[CosObjOffsetProc](#)
[CosObjSetCallbackFlagProc](#)
[DoClickProcType](#)
[DoLeaveProcType](#)
[GetSelectionServerProcType](#)
[PDAnnotHandlerDeleteAnnotInfoProc](#)
[PDAnnotHandlerGetAnnotInfoFlagsProc](#)
[PDAnnotHandlerGetAnnotInfoProc](#)
[PDAnnotHandlerGetTypeProc](#)

PDAnnotWillPrintProc
PDCryptNewCryptDataExProc
PDDocPreSaveProc
PDDocWillExportAnnotCallback
PDDocWillExportAnnotProc
PDDocWillImportAnnotCallback
PDDocWillImportAnnotProc
PDEClipEnumProc
PIExportHFTsProcType
PIHandshake
PIImportReplaceAndRegisterProcType
PIInitProcType
PIUnloadProcType

The following callbacks were added for the Adobe PDF Library.

ASMemAllocProc
ASMemAvailProc
ASMemFreeProc
ASMemReallocProc
PDFLPrintCancelProc
TKResourceAcquireProc
TKResourceReleaseProc

The following callbacks were added for Placed PDF, which is part of the Adobe PDF Library.

AGMMemAllocator
AGMMemDeleter
AGMPortDestructProcPtr
DoCancel
DocBegin
DocEnd
DocSetup
EmitPageContents
EmitPrologString
EmitPSFontBegin
EmitPSFontEncodingBegin
EmitPSFontEncodingEnd
EmitPSFontEnd
EmitPSResourceBegin
EmitPSResourceEnd
EndSetup
FlushString
GetFontVMUsage
NotifyNewPage
PageBegin
PageEnd

PageSetup
 PDPrintCanEmitFontProc
 PDPrintEmitFontProc
 PDPrintEmitPrologResourceProc
 PDPrintGetFontEncodingMethodProc

Data

The following data structures were added in version 4.0.

AGMBlackPointFlt
 AGMColorRangeFlt
 AGMGrayCalFlt
 AGMLabCalFlt
 AGMRGBCalFlt
 AGMWhitePointFlt
 AGMXYZColorFlt
 AVAnnotHandler
 AVAnnotHandlerInfo
 AVDocSaveParams
 AVIcon
 AVPageViewControlID
 PDAnnotHandler
 PDAnnotInfo
 PDDocCopyParams
 PDDocOpenParams
 PDDocPreSaveInfo
 PDEColorSpaceStruct
 PDEDeviceNColorData
 PDEFontCreateParams
 PDEGrayCalFlt
 PDEICCBasedColorData
 PDEIndexedColorData
 PDELabCalFlt
 PDEPatternColorSpace
 PDERGBCalFlt
 PDESeparationColorData
 PDFLPrintUserParamsRec (for Adobe PDF Library)
 PDFontStyles
 PDPageDrawFlags
 PIHandshakeData_V0200
 StdPassword
 StdSecurityData
 TKAllocatorProcs (for Adobe PDF Library)
 TKResourceProcs (for Adobe PDF Library)

The following data structures were added for Placed PDF, which is part of the Adobe PDF Library.

AGMCMYKColorRec
AGMColorTab
AGMFixedMatrix
AGMFixedPoint
AGMImageAlphaRecord
AGMInt16Rect
AGMLABColorRec
AGMMemObj
AGMRGBColorRec
PDPageDrawFlags
PDPrintClient

Errors

The following errors were added in version 4.0.

avErrActionExternal
avErrActionFullScreen
avErrActionRestricted
avErrCantOpenDialog
avErrCantOpenPrinting
pageErrBadEPSColorSpace
pageErrBadFunction
pageErrInvalidImageMaskDepth
pdErrCannotMergeWithSubsetFonts
pdErrHostEncodingNotSet
pdErrNoPDDocForCosDoc
ErrSysRaster
peErrCantCreateFontSubset
peErrCantEmbedFont
peErrCantGetAttrs
peErrCantGetWidths
peErrCantReadImage
peErrFontToEmbedNotOnSys
peErrNoError
peErrPStackUnderflow
peErrUnknownPDEColorSpace
peErrUnknownResType
peErrWrongPDEObjectType

Notifications

The following notifications were added in version 4.0.

AVAppWillCloseAllInternalDocs
AVDocDidClickName
AVDocDidPrint
AVDocWillPrint
PDBookmarkDidUnlink
PDDocDidClose



PDDocDidExportAnnots
PDDocDidImportAnnots
PDDocDidInsertPagesEx
PDDocPageLabelDidChange
PDDocWillClose
PDDocWillExportAnnots
PDDocWillImportAnnots
PDDocWillInsertPagesEx
PDNameTreeNameAdded
PDNameTreeNameRemoved
PSPrintAfterBeginPageSetup
PSPrintAfterBeginProlog
PSPrintAfterBeginSetup
PSPrintAfterEmitExtGState
PSPrintAfterPageTrailer
PSPrintAfterTrailer
PSPrintBeforeEndComments
PSPrintBeforeEndSetup



Version 5

Objects

The following objects were added in version 5.0.

[ASCab](#)
[ASText](#)
[AVCommand](#)
[AVSweetPea](#)
[PDEBeginContainer](#)
[PDEBeginGroup](#)
[PDEEndContainer](#)
[PDEEndGroup](#)
[PDEPS](#)
[PDESoftMask](#)
[PDEXGroup](#)
[PDSysEncoding](#)

Modified Objects

The following objects were modified in version 5.0.

[PDWord](#)

Methods

The following methods were deprecated in version 5.0.

[ASPathFromPlatformPath](#)
[AVToolBarGetFrame](#)

The following method was deleted in version 5.0.

[AVAppWindowHandlePlatformEvent](#)

The following method UNIX-specific method was added in version 5.0.

[UnixMachinePortGetOffscreenPixmap](#)

The following methods were added or modified in version 5.0.

Available only if **PI_ASEXTRA_VERSION** (in **PIRequir.h**) is set to **0x00050000** or higher.

[ASCabCopy](#)
[ASCabDestroy](#)
[ASCabDestroyEmpties](#)
[ASCabDetachBinary](#)
[ASCabDetachCab](#)
[ASCabDetachPathName](#)
[ASCabDetachPointer](#)
[ASCabDetachString](#)
[ASCabDetachText](#)
[ASCabDup](#)

ASCabEnum
ASCabEqual
ASCabFromEntryList
ASCabGetAtom
ASCabGetBinary
ASCabGetBinaryCopy
ASCabGetBool
ASCabGetCab
ASCabGetDouble
ASCabGetInt
ASCabGetPathNameCopy
ASCabGetPointer
ASCabGetPointerDestroyProc
ASCabGetPointerType
ASCabGetString
ASCabGetStringCopy
ASCabGetText
ASCabGetType
ASCabKnown
ASCabMakeEmpty
ASCabMunge
ASCabNew
ASCabNumEntries
ASCabPutAtom
ASCabPutBinary
ASCabPutBool
ASCabPutCab
ASCabPutDouble
ASCabPutInt
ASCabPutNull
ASCabPutPathName
ASCabPutPointer
ASCabPutString
ASCabPutText
ASCabReadFromStream
ASCabRemove
ASCabRename
ASCabValueEqual
ASCabWriteToStream
ASScriptFromHostEncoding
ASScriptToHostEncoding
ASTextCat
ASTextCatMany
ASTextCmp
ASTextCopy
ASTextDestroy
ASTextDup

ASTextFromEncoded
ASTextFromInt32
ASTextFromPDText
ASTextFromScriptText
ASTextFromSizedEncoded
ASTextFromSizedPDText
ASTextFromSizedScriptText
ASTextFromSizedUnicode
ASTextFromUnicode
ASTextFromUns32
ASTextGetBestEncoding
ASTextGetBestScript
ASTextGetCountry
ASTextGetEncoded
ASTextGetEncodedCopy
ASTextGetLanguage
ASTextGetPDTextCopy
ASTextGetScriptText
ASTextGetScriptTextCopy
ASTextGetUnicode
ASTextGetUnicodeCopy
ASTextIsEmpty
ASTextMakeEmpty
ASTextNew
ASTextNormalizeEndOfLine
ASTextReplace
ASTextReplaceASCII
ASTextSetCountry
ASTextSetEncoded
ASTextSetLanguage
ASTextSetPDText
ASTextSetScriptText
ASTextSetSizedEncoded
ASTextSetSizedPDText
ASTextSetSizedScriptText
ASTextSetSizedUnicode
ASTextSetUnicode

Available only if `PI_ACROSUPPORT_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

ASFileGetOpenMode
ASFileGetURL
ASFileHardFlush
ASFileIsSame
ASFileSysAcquireParent
ASFileSysCreateFolder
ASFileSysDestroyFolderIterator

ASFileSysDisplayStringFromPath
ASFileSysFirstFolderItem
ASFileSysFlushVolume
ASFileSysGetItemProps
ASFileSysGetNameFromPath
ASFileSysGetStorageFreeSpace
ASFileSysGetTempPathName
ASFileSysGetTypeAndCreator
ASFileSysNextFolderItem
ASFileSysRemoveFolder
ASFileSysSetTypeAndCreator
ASFileSysURLFromPath
ASGetSecs
ASHostMBLen
ASPathFromPlatformPath
HFTReplaceEntryEx
HFTUnreplaceEntry

Available only if `PI_ACROVIEW_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

AVAcquireSpecialFilePathName
AVAcquireSpecialFolderPathName
AVAlertGetPref
AVAlertResetPrefs
AVAlertSetPref
AVAlertWithParams
AVAppChooseFolderDialog
AVAppDidOrWillSwitchForDialog
AVAppFindCommandHandlerByName
AVAppFindGlobalCommandByName
AVAppGetLanguageEncoding
AVAppGetReportProc
AVAppGetToolBarByName
AVAppOpenDialog
AVAppRegisterCommandHandler
AVAppRegisterFromPDFHandler
AVAppRegisterGlobalCommand
AVAppRegisterToolBarPosition
AVAppRegisterToPDFHandler
AVAppSaveDialog
AVAppUnregisterGlobalCommand
AVAppYieldToOtherApps
AVCommandCancel
AVCommandDestroy
AVCommandExecute
AVCommandGetAVDoc
AVCommandGetCab

AVCommandGetCancelProc
AVCommandGetConfig
AVCommandGetInputs
AVCommandGetName
AVCommandGetParams
AVCommandGetPDDoc
AVCommandGetProgressMonitor
AVCommandGetProps
AVCommandGetReportProc
AVCommandGetStatus
AVCommandGetUIPolicy
AVCommandNew
AVCommandPutCab
AVCommandReset
AVCommandSetConfig
AVCommandSetInputs
AVCommandSetParams
AVCommandShowDialog
AVCommandWork
AVConversionConvertFromPDFWithHandler
AVConversionConvertToPDF
AVConversionConvertToPDFWithHandler
AVConversionEnumFromPDFConverters
AVConversionEnumToPDFConverters
AVDocFromPDDoc
AVDocIsDead
AVDocPerformActionEx
AVDocPermRequest
AVExtensionAcquireInfo
AVExtensionGetNumPlugIns
AVExtensionReleaseInfo
AVIdentityGetText
AVIdentitySetText
AVPageViewClearFocusAnnot
AVPageViewDeviceToInfo
AVPageViewDragOutNewRectSnapped
AVPageViewDragRectSnapped
AVPageViewDrawRectOutlineWithHandles
AVPageViewFilterKeyDownForFocusAnnot
AVPageViewFocusAnnotPerformOp
AVPageViewGetFocusAnnot
AVPageViewGetMousePositionSnapped
AVPageViewGhostRectOutline
AVPageViewInfoToDevice
AVPageViewInfoToPoint
AVPageViewIsAnnotOfTypeAtPoint
AVPageViewIsFocusAnnot

AVPageViewPointToInfo
 AVPageViewResumeOffscreenDrawing
 AVPageViewScrollToAnnot
 AVPageViewSetFocusAnnot
 AVPageViewSnapPoint
 AVPageViewSnapRect
 AVPageViewSuspendOffscreenDrawing
 AVPageViewUpdateInfoPanel
 AVRectHandleHitTest
 AVSweetPeaGetBasicSuiteP
 AVSweetPeaGetPluginRef
 AVSweetPeaGetResourceAccess
 AVSweetPeaIsADMAvailable
 AVSweetPeaProcessADMEvent
 AVSweetPeaSetResourceAccess
 AVSysSetWaitCursor
 AVToolBarNew
 AVUtilGetBaseNameAndExtensionByPathName
 AVUtilGetBaseNameAndExtensionByString
 AVWindowCenter
 AVWindowEnsureInBounds

Available only if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00050001** or higher.

AVMenuClone

Available only if **PI_PDMETADATA_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

CosDictGetXAPMetadata
 CosDictSetXAPMetadata
 PDDocCalculateImplicitMetadata
 PDDocGetXAPMetadata
 PDDocSetXAPMetadata
 PDEContainerGetXAPMetadata
 PDEContainerSetXAPMetadata

Available only if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

CosCopyStringValue
 CosDocGetID
 CosObjCmp
 CosStringValueSafe

Available only if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00050000** or higher.

PDBookmarkGetColor
 PDBookmarkGetFlags
 PDBookmarkRemoveAction
 PDBookmarkSetColor

PDBookmarkSetFlags
PDDocExportSomeNotes
PDDocGetPageObjByNum
PDDocPermRequest
PDDocRemoveOpenAction
PDLinkAnnotRemoveAction
PDNameTreeNotifyNameAdded
PDNameTreeNotifyNameRemoved
PDPageDrawContentsToWindowEx
PDPageGetBox
PDPageGetPalette
PDPageHasTransparency
PDPageResumePDEContentChanged
PDPageSetBox
PDTextSelectCreatePageHiliteEx
PDTextSelectCreateRangesEx
PDTextSelectCreateWordHiliteEx
PDTextSelectEnumTextUCS
PDPageSuspendPDEContentChanged

Available only if `PI_PDFEDIT_READ_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

PDEBeginContainerGetDict
PDEBeginContainerGetMCTag
PDEElementHasGState
PDEExtGStateAcquireSoftMask
PDEExtGStateGetAIS
PDEExtGStateGetBlendMode
PDEExtGStateGetOpacityFill
PDEExtGStateGetOpacityStroke
PDEExtGStateGetOPFill
PDEExtGStateGetOPM
PDEExtGStateGetOPStroke
PDEExtGStateGetSA
PDEExtGStateGetTK
PDEExtGStateHasSoftMask
PDEFormAcquireXGroup
PDEFormHasXGroup
PDEImageGetMatteArray
PDEImageGetSMask
PDEImageHasSMask
PDESoftMaskAcquireForm
PDESoftMaskGetBackdropColor
PDESoftMaskGetCosObj
PDESoftMaskGetName
PDESoftMaskGetTransferFunction
PDTextGetMatrix

PDTextGetState
 PDEXGroupAcquireColorSpace
 PDEXGroupGetCosObj
 PDEXGroupGetIsolated
 PDEXGroupGetKnockout
 PDSysEncodingGetWMode
 PDSysEncodingIsIdentity
 PDSysEncodingIsMultiByte

Available only if `PI_PDFEDIT_WRITE_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

PDEBeginContainerCreate
 PDEBeginContainerSetDict
 PDEBeginContainerSetMCTag
 PDEBeginGroupCreate
 PDEEndContainerCreate
 PDEEndGroupCreate
 PDEExtGStateCreateNew
 PDEExtGStateSetAIS
 PDEExtGStateSetBlendMode
 PDEExtGStateSetOpacityFill
 PDEExtGStateSetOpacityStroke
 PDEExtGStateSetOPFill
 PDEExtGStateSetOPM
 PDEExtGStateSetOPStroke
 PDEExtGStateSetSA
 PDEExtGStateSetSoftMask
 PDEExtGStateSetTK
 PDEFontCreateFromSysFontAndEncoding
 PDEFontCreateFromSysFontWithParams
 PDEFontCreateToUnicodeNow
 PDEFontCreateWidthsNow
 PDEFontEmbedNow
 PDEFontEmbedNowDontSubset
 PDEFontGetCreateNeedFlags
 PDEFontGetWidthsNow
 PDEFontTranslateGlyphIdsToUnicode
 PDEFormSetXGroup
 PDEImageSetColorSpace
 PDEImageSetMatteArray
 PDEImageSetSMask
 PDESoftMaskCreate
 PDESoftMaskCreateFromCosObj
 PDESoftMaskCreateFromName
 PDESoftMaskSetBackdropColor
 PDESoftMaskSetTransferFunction
 PDESoftMaskSetXGroup

[PDTextRunSetMatrix](#)
[PDTextRunSetState](#)
[PDEXGroupCreate](#)
[PDEXGroupCreateFromCosObj](#)
[PDEXGroupSetColorSpace](#)
[PDEXGroupSetIsolated](#)
[PDEXGroupSetKnockout](#)
[PDSysEncodingCreateFromBaseName](#)
[PDSysEncodingCreateFromCMapName](#)
[PDSysFontGetCreateFlags](#)

Available only if `PI_PDSYSFONT_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

[PDFindSysFontForPDEFont](#)

Available only if `PI_PDS_READ_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

[PDSElementGetActualText](#)
[PDSElementGetKidEx](#)
[PDSElementGetLanguage](#)
[PDSElementHasActualText](#)
[PDSElementHasAlt](#)
[PDSElementHasLanguage](#)

Available only if `PI_PDS_WRITE_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

[PDSElementRemoveKidOBJ](#)
[PDSElementSetActualText](#)
[PDSElementSetLanguage](#)

Available only if `PI_WIN_VERSION` (in `PIRequir.h`) is set to `0x00050000` or higher.

[WinAppRegisterInterface](#)

Callbacks (new or modified)

The following callbacks were added in version 5.0.

[ASCabEnumProc](#)
[ASCabPointerDestroyProc](#)
[ASCancelProc](#)
[ASFileSysCreateFolderProc](#)
[ASFileSysDestroyFolderIteratorProc](#)
[ASFileSysDisplayStringFromPathProc](#)
[ASFileSysFirstFolderItemProc](#)
[ASFileSysGetItemPropsProc](#)
[ASFileSysGetParentProc](#)
[ASFileSysGetTypeAndCreatorProc](#)
[ASFileSysNextFolderItemProc](#)
[ASFileSysRemoveFolderProc](#)
[ASFileSysSetTypeAndCreatorProc](#)

ASFileSysURLFromPathProc
ASReportProc
AVActionPerformExProc
AVAnnotHandlerCanPerformOpProc
AVAnnotHandlerDoKeyDownExProc
AVAnnotHandlerDrawExProc
AVAnnotHandlerPerformOpProc
AVCmdHandlerInitProc
AVCmdHandlerTermProc
AVCommandCancelProc
AVCommandCreatedProc
AVCommandDestroyProc
AVCommandGetProc
AVCommandPostflightFileProc
AVCommandPostflightSequenceProc
AVCommandPreflightFileProc
AVCommandPreflightSequenceProc
AVCommandRegisterCommandsProc
AVCommandResetProc
AVCommandSetProc
AVCommandShowDialogProc
AVCommandWorkProc
AVConversionConvertFromPDFProc
AVConversionConvertToPDFProc
AVConversionDefaultSettingsProc
AVConversionFromPDFEnumProc
AVConversionParamDescProc
AVConversionSettingsDialogProc
AVConversionToPDFEnumProc
AVDocPermReqProc
AVDocSelectionGetAVRectProc
AVDocSelectionShowMenuProc
AVOpenSaveDialogSettingsComputeEnabledProc
AVOpenSaveDialogSettingsExecuteProc
AVSetFocusProc
PDCryptAuthorizeExProc
PDCryptBatchAuthorizeProc
PDCryptBatchFreeAuthDataProc
PDCryptBatchNewAuthDataProc
PDCryptBatchParamDescProc
PDCryptBatchPostSequenceProc
PDCryptBatchPreSequenceProc
PDCryptBatchShowDialogProc
PDCryptBatchUpdateSecurityDataProc
PDCryptDisplaySecurityDataProc
PDCryptGetAuthDataExProc
PDCryptReservedProc

PDImplicitMetadataProc

PMSetTextProc

Data (New or modified)

The following data structures were added or changed in version 5.0.

ASCabEntryRec

ASCabMungeAction

ASCabValueType

ASFile Flags

ASFileSysCreatePathFromCString (macro)

ASFileSysCreatePathFromDIPath (macro)

ASFileSysCreatePathFromFSSpec (macro)

ASFileSysCreatePathWithFolderName (macro)

ASFileSysRec

ASFileSysItemProps

ASFileSysItemType

ASFolderIterator

ASFixedRoundToInt16 (macro)

ASHostEncoding

ASMDFile (replaced MDFile)

ASPortRef

ASProgressMonitor

ASReportType

ASScript

ASUnicodeFormat

ASWindowRef

AVAccessColorPolicy

AVActionContext

AVAlertButtonInfo

AVAlertCheckBoxInfo

AVAlert Icons

AVAlertParams

AVAnnotHandler

AVAnnotOp

AVAnnotOpData

AVBatchContext

AVCommandHandler

AVCommandStatus

AVCommandUIPolicy

AVConversionClientData

AVConversionEnumProcData

AVConversionFlags

AVConversionFromPDFHandler

AVConversionStatus
AVConversionToPDFHandler
AVDocOpenParams
AVDocPrintParams
AVDocSaveParams
AVDocSelectionServer
AVExtensionInfo
AVFileDescRec
AVFileFilterRec
AVFullScreenMonitor
AVIconBundle
AVIdentity
AVInfoPanelUpdateType
AVOpenSaveDialogFlags
AVOpenSaveDialogParams
AVPageViewControlID
AVPrefsType
AVRectHandleType
AVSpecialCategory
AVSpecialError
AVSpecialFolder
AVStatusMonitorProcs
AVToolBarDockPosition
AVToolBarLayout
AVToolBarPosition
AVTransitionPort
AVWindow Flags
AVWindowLayer
AVZoomType
Command Keys
Emit Flags
ExternalDocServerCreationData
Modifier Keys
PDAnnotArray
PDAnnotInfo
PDCryptBatchHandler
PDCryptHandler
PDEFontAttrs
PDEFontCreateFromSysFontParams
PDEFontCreateFlags
PDEFontCreateNeedFlags
PDEGrayCalFlt
PDELabCalFlt

PDERGBCalFlt
PDESoftMaskCreateFlags
PDTextState
PDType
PDEXGroupCreateFlags
PDPermReqObj
PDPermReqOpr
PDPermReqStatus
PDTile
Predefined Cursors
StdSecurityData
Type/Creator Codes

Errors

The following errors were added in version 5.0.

pdErrCannotBeBlankPage
pdErrCannotDeleteAllPages
pdErrExceedEncryptionLength
pdErrExceedEncryptionVersion
pdErrInvalidPageNumber
pdErrMissingGlyphs
pdErrMissingSubsetFont
pdErrNeedJapanese
pdErrNeedKorean
pdErrNeedSimpChinese
pdErrNeedTradChinese
pdErrNotValidPage
pdErrRequireTrustedMode
pdErrStartLessThanEnd
ErrSysPDModel
pdMetadataErrBadPDF
pdMetadataErrCouldntCreateMetaXAP
pdMetadataErrInternalError

Notifications

The following notifications were added in version 5.0.

AVAppOldPrefDidChange
AVAppPrefDidChange
AVAppUsingPDDocForBatch
AVDocDidDeleteSelection
AVPageViewAnnotDidPerformOp
AVPageViewWillDraw
PDDocDidChangePageAreas
PDDocCalculateMetadata
PDDocDidOpen

[PDDocDidPrintTiledPage](#)
[PDDocPrintingTiledPage](#)
[PDDocWillPrintTiledPage](#)
[PDPageDidPrintAnnot](#)
[PDPageDidPrintAnnots](#)
[PDPageWillPrintAnnot](#)
[PDPageWillPrintAnnots](#)
[PDNameTreeNameAdded](#)
[PDNameTreeNameRemoved](#)
[PSPrintBeforeAcrobatProcsets](#)

Deleted PDFLibrary-specific Methods, Callbacks, and Data:

The following methods, callbacks, and data were deleted because they are specific to the PDF Library, so are not available to Acrobat clients. The methods, callbacks, and data that remain available in the PDF Library are now documented in the *PDF Library Supplement to the Core API Reference*. See that document for more information.

NOTE: Many of the following methods, callbacks, and data are no longer available in any Adobe product. They are mentioned here merely for API record-keeping purposes.

ASAtomGetCount (method)
ASPurgeMemory (method)
AVExtensionMgrRegisterNotification (method)
AVExtensionMgrUnregisterNotification (method)
PDFLGetVersion (method)
PDFLInit (method)
PDFLPrintDoc (method)
PDFLPrintPDF (method)
PDFLTerm (method)
AGMCleanup (method)
AGMClip (method)
AGMClosePath (method)
AGMConcat (method)
AGMDeletePort (method)
AGMDeleteRasterDev (method)
AGMFill (method)
AGMInit (method)
AGMLineTo (method)
AGMMoveTo (method)
AGMNewBitmapPort (method)
AGMNewPath (method)
AGMNewRasterDev (method)
AGMNewRasterPort (method)
AGMNewWindowPort (method)
AGMSetAntiAliasPolicy (method)
AGMSetRGBColor (method)
CCSetSystemCalibration (method)
PDDocPrintPages (method)

PDFontDownloadContextCreate (method)
PDFontDownloadContextDestroy (method)
PDFontStreamPS (method)
PDFontWasExtracted (method)
PDFontWasFauxed (method)
PDFreeMemory (method)
PDPageDrawContents (method)
PDPageDrawContentsPlaced (method)
PDPageDrawContentsPlacedToWindow (method)
PDPageEmitPSOrient (method)
PDSetHostEncoding (method)
AGMMemAllocator (callback)
AGMMemDeleter (callback)
AGMPortDestructProcPtr (callback)
ASMemAllocProc (callback)
ASMemAvailProc (callback)
ASMemFreeProc (callback)
ASMemReallocProc (callback)
DoCancel (callback)
DocBegin (callback)
DocEnd (callback)
DocSetup (callback)
EmitPageContents (callback)
EmitPrologString (callback)
EmitPSFontBegin (callback)
EmitPSFontEncodingBegin (callback)
EmitPSFontEncodingEnd (callback)
EmitPSFontEnd (callback)
EmitPSResourceBegin (callback)
EmitPSResourceEnd (callback)
EndSetup (callback)
FlushString (callback)
GetFontVMUsage (callback)
NotifyNewPage (callback)
PageBegin (callback)
PageEnd (callback)
PageSetup (callback)
PDFLPrintCancelProc (callback)
PDPrintCanEmitFontProc (callback)
PDPrintEmitFontProc (callback)
PDPrintEmitPrologResourceProc (callback)
PDPrintGetFontEncodingMethodProc (callback)
TKResourceAcquireProc (callback)
TKResourceReleaseProc (callback)
AGMBlackPointFlt (data)
AGMCMYKColorRec (data)
AGMColorRangeFlt (data)

AGMColorTab (data)
AGMFixedMatrix (data)
AGMFixedPoint (data)
AGMGrayCalFlt (data)
AGMImageAlphaRecord (data)
AGMInt16Rec (data)
AGMLabCalFlt (data)
AGMLABColorRec (data)
AGMMemObj (data)
AGMRGBCalFlt (data)
AGMRGBColorRec (data)
AGMWhitePointFlt (data)
AGMXYZColorFlt (data)
PDFLData (data)
PDFLPrintUserParamsRec (data)
PDInclusion (data)
PDOutputType (data)
PDPrintClient (data)
PDPrintParams (data)
TKAllocatorProcs (data)
TKResourceProcs (data)

Version 6

- [New Objects](#): introduced in this release
- [Modified Objects](#): with new or modified methods in this release
- [Deprecated Methods and Callbacks](#)
- [New and Modified Methods](#): introduced or significantly modified in this release
- [Methods with Cosmetic Type Changes](#)
- [New and Modified Callbacks](#): introduced or significantly modified in this release
- [Callbacks with Cosmetic Type Changes](#)
- [New and Modified Data](#): introduced or significantly modified in this release
- [Data with Cosmetic Type Changes](#)
- [New Notifications](#): introduced in this release

NOTE: Cosmetic Changes: Acrobat 6.0 introduced named cover types that are used in place of generic numeric types in methods and data structures (generally to support a future change from 16-bit to 32-bit values). Methods and data that use these named cover types are listed separately.

In this release, methods and data can be compiled without error if they still use the generic types, but future releases may conditionally compile the named types depending on the release version. It is recommended that you update your code to use the new named types.

New Objects

The following objects were added in version 6.0.

Available only if **PI_ASEXTRA_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

[ASDate](#)

[ASPlatformPath](#)

[ASTimeSpan](#)

Available only if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

[AVUndo](#)

Available only if **PI_COS_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

[CosObjCollection](#)

Available only if **PI_PDMODEL_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

[PDOCConfig](#)

[PDOCContext](#)

[PDOCG](#)

[PDOCMD](#)

Available if `PI_PDFEDIT_READ_VERSION` and `PI_PDFEDIT_WRITE_VERSION` (in `PIRequir.h`) are set to `0x00060000` or higher.

[PDETextItem](#)

Modified Objects

The following objects were modified in version 6.0.

[ASCab](#)

[ASFile](#)

[ASFileSys](#)

[ASText](#)

[AVAnnotHandler](#)

[AVApp](#)

[AVConversion](#)

[AVDoc](#)

[AVMenu](#)

[AVMenuItem](#)

[AVPageView](#)

[AVToolButton](#)

[AVWindow](#)

[HFT](#)

[CosDoc](#)

[CosObj](#)

[PDAnnot](#)

[PDDoc](#)

[PDPage](#)

[PDWord](#)

[PDWordFinder](#)

[PDEContent](#)

[PDEElement](#)

[PDEText](#)

Deprecated Methods and Callbacks

The following methods and callbacks are deprecated in version 6.0. They should not be used, and will be removed from public header files in future releases.

[AVAppOpenHelpFile](#)

[AVAnnotHandlerAdjustCursorProc](#)

[AVAnnotHandlerDoClickProc](#)

[AVAnnotHandlerDoKeyDownProc](#)

AVAnnotHandlerDoPropertiesProc
AVAnnotHandlerDrawProc
AVAnnotHandlerGetLayerProc
PDDocGetPermissions

New and Modified Methods

The following methods were added or modified in version 6.0. All methods are new unless otherwise marked.

Available only if `PI_ASEXTRA_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

ASCabGetUns
ASCabPutUns
ASConstCabEnum
ASCalendarTimeSpanAddWithBase
ASCalendarTimeSpanCompare
ASCalendarTimeSpanDiff
ASDateAddCalendarTimeSpan
ASDateAddTimeSpan
ASDateCalendarDiff
ASDateClear
ASDateCompare
ASDateCopy
ASDateDestroy
ASDateDup
ASDateExactDiff
ASDateGetLocalTime
ASDateGetTimeString
ASDateGetUTCTime
ASDateNew
ASDateSetLocalTimeOffset
ASDateSetTimeFromRec
ASDateSetTimeFromString
ASDateSetToCurrentLocalTime
ASDateSetToCurrentUTCTime
ASDateSubtractCalendarTimeSpan
ASDateSubtractTimeSpan
ASFileCanSetEOF
ASFileHasOutstandingMReads
ASFileSysAcquirePlatformPath
ASFileSysCanPerformOpOnItem
ASFileSysConvertCabToItemProps
ASFileSysConvertItemPropsToCab

ASFileSysDisplayASTextFromPath
ASFileSysDIPathFromPathEx
ASFileSysGetItemPropsAsCab
ASFileSysGetNameFromPathAsASText
ASFileSysPathFromDIPathEx
ASFileSysPerformOpOnItem
ASFileSysReleasePlatformPath
ASIsValidUTF8
ASPlatformPathGetCFURLRefRecPtr
ASPlatformPathGetCstringPtr
ASPlatformPathGetFSRefPtr
ASPlatformPathGetFSRefWithCFStringRefRecPtr
ASPlatformPathGetFSSpecPtr
ASPlatformPathGetPOSIXPathPtr
ASProcStmRdOpenEx
ASStmFlush
ASTextCaseSensitiveCmp
ASTextEval
ASTextFilter
ASTextReplaceBadChars
ASTimeSpanAdd
ASTimeSpanCompare
ASTimeSpanCopy
ASTimeSpanDestroy
ASTimeSpanDiff
ASTimeSpanDup
ASTimeSpanGetASInt32
ASTimeSpanNegate
ASTimeSpanNew
ASTimeSpanSet
ASTimeSpanSetFromASInt32
ASTimeSpanSetFromString
ASUUIDFromCString
ASUUIDGenFromHash
ASUUIDGenFromName
ASUUIDGenUnique
ASUUIDToCString
HFTGetVersion
HFTNewEx

Available only if **PI_ACROVIEW_VERSION** (in `PIRequir.h`) is set to **0x00060000** or higher.

AVAcquireSpecialFilePathNameWithASText

AVAppAutoShowHowToPanel
AVAppCreateIconBundle6
AVAppGetHowToPanelAutoShow
AVAppGetLanguageWithParams
AVAppGetUUID
AVAppHelpSearch
AVAppHelpShowContents
AVAppHelpShowIndex
AVAppOpenHelpFileWithParams
AVAppRegisterAnnotHandler
AVAppRegisterForContextMenuAddition
AVAppRegisterForPageViewAdjustCursor (modified)
AVAppRegisterForPageViewClicks (modified)
AVAppRegisterForPageViewDrawing (modified)
AVAppRegisterForPageViewRightClicks
AVAppRegisterHowToPanel
AVAppRegisterToolBarPosition (modified)
AVAppSetHowToPanelAutoShow
AVAppSetHowToPanelAutoShowText
AVAppSetHowToPanelComputeVisibleProc
AVAppUnregisterForPageViewDrawingEx
AVAppUnregisterForPageViewRightClicks

AVConversionConvertStreamFromPDFWithHandler
AVConversionConvertStreamFromStructNodeWithHandler
AVConversionConvertStreamToPDF
AVConversionConvertStreamToPDFWithHandler

AVDocBeginUndoOperation
AVDocClearUndos
AVDocEndUndoOperation
AVDocGetActiveTool
AVDocGetNthPageView
AVDocGetNumPageViews
AVDocGetPageText
AVDocGetSelectionServerByType
AVDocGetServerType
AVDocGetTopUndo
AVDocGetTopUndoAndRedo
AVDocGetViewDef (modified)
AVDocGetViewDefEx
AVDocIsSlow
AVDocOpenFromASFileWithParams (modified)
AVDocOpenFromASFileWithParamString
AVDocOpenFromFile (modified)
AVDocOpenFromFileWithParamString
AVDocOpenFromPDDoc (modified)
AVDocOpenFromPDDocWithParams (modified)

AVDocOpenFromPDDocWithParamString
AVDocPrintPagesWithParams (modified)
AVDocPrintSeparations
AVDocRegisterSelectionServer
AVDocSetActiveTool
AVDocSetViewDef (modified)
AVDocSetViewDefEx

AVMenuDoPopUp
AVMenuGetTitleAsASText
AVMenuNewWithASText

AVMenuItemClone
AVMenuItemGetTitleAsASText
AVMenuItemIsVisible
AVMenuItemNewWithASText
AVMenuItemSetComputeVisibleProc
AVMenuItemSetTitleWithASText

AVPageViewDevicePointToPage (modified)
AVPageViewDeviceRectToPage (modified)
AVPageViewDeviceRectToPageRZ (modified)
AVPageViewDoPopupMenu (modified)
AVPageViewDragOutNewRect (modified)
AVPageViewDragOutNewRectSnapped (modified)
AVPageViewDragRect (modified)
AVPageViewDragRectSnapped (modified)
AVPageViewDragRectSnappedEx
AVPageViewDrawAnnotSequence (modified)
AVPageViewDrawCosObj (modified)
AVPageViewDrawCosObjEx (modified)
AVPageViewDrawNowWithTransition
AVPageViewDrawPolygon
AVPageViewDrawPolygonOutline
AVPageViewDrawRect (modified)
AVPageViewDrawRectOutline (modified)
AVPageViewDrawRectOutlineWithHandles (modified)
AVPageViewGetAnnotRect (modified)
AVPageViewGetAperture (modified)
AVPageViewGetGrayRect (modified)
AVPageViewGetMousePosition (modified)
AVPageViewGetMousePositionSnapped (modified)
AVPageViewGetNumVisibleInks
AVPageViewGetPageToDevScaling
AVPageViewGetPixelInformationAtPoint
AVPageViewGetVisibleInks
AVPageViewGhostRectOutline (modified)
AVPageViewInsetRect (modified)
AVPageViewInvalidateRect (modified)

[AVPageViewInvertQuad](#) (modified)
[AVPageViewInvertRect](#) (modified)
[AVPageViewIsAnnotAtPoint](#) (modified)
[AVPageViewIsAnnotOfTypeAtPoint](#) (modified)
[AVPageViewIsBeadAtPoint](#) (modified)
[AVPageViewPointInText](#) (modified)
[AVPageViewPointToDevice](#) (modified)
[AVPageViewRectToDevice](#) (modified)
[AVPageViewSetAnnotLocation](#) (modified)
[AVPageViewSetInkPreview](#)
[AVPageViewSetVisibleInks](#)
[AVPageViewScrollTo](#) (modified)
[AVPageViewScrollToRect](#) (modified)
[AVPageViewSnapPoint](#) (modified)
[AVPageViewSnapPointEx](#)
[AVPageViewSnapRect](#) (modified)
[AVPageViewTrackText](#) (modified)
[AVRectHandleHitTest](#) (modified)
[AVSysGetIconFromFilename](#)
[AVSysGetIconFromMimeType](#)
[AVSysGetIconFromTypeAndCreator](#)
[AVSysGetUsePenForInput](#)
[AVToolButtonGetLabelText](#)
[AVToolButtonIsVisible](#)
[AVToolButtonSetComputeVisibleProc](#)
[AVToolButtonSetLabelText](#)
[AVToolButtonSetNotifyTooltipProc](#)
[AVUndoGetAVDoc](#)
[AVUndoGetData](#)
[AVUndoGetType](#)
[AVUndoNew](#)
[AVUndoSetData](#)
[AVUtilGetBaseNameAndExtensionEx](#)
[AVWindowDoModal](#)
[AVWindowEndModal](#)
[AVWindowGetBorderWidths](#)
[AVWindowGetDesktopBounds](#)
[AVWindowGetFrame](#) (modified)
[AVWindowGetInterior](#) (modified)
[AVWindowGetMinMaxSize](#) (modified)
[AVWindowGetTitle](#) (modified)
[AVWindowInvalidateRect](#) (modified)
[AVWindowNew](#) (modified)
[AVWindowNewFromPlatformThing](#) (modified)

[AVWindowSetFrame](#) (modified)
[AVWindowSetMinMaxSize](#) (modified)
[AVWindowSetTitle](#) (modified)

Available if `PI_COS_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

[CosDocHasFullCompression](#)
[CosDocHasPartialCompression](#)
[CosDocObjIsWithinRange](#)
[CosDocSaveToFile](#)
[CosDocSaveWithParams](#)
[CosNewObjCollection](#)
[CosObjAddToCollection](#)
[CosObjCollectionEnum](#)
[CosObjCollectionEqual](#)
[CosObjCollectionIsNull](#)
[CosObjCollectionSize](#)
[CosObjGetCollection](#)
[CosObjGetCompressibility](#)
[CosObjIsCompressed](#)
[CosObjRefreshAfterLinearizedSave](#)
[CosObjRemoveFromCollection](#)
[CosObjSetCompressibility](#)
[CosSetMaxDocStorages](#)

Available only if `PI_PDMODEL_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

[PDActionCanCopy](#)
[PDActionCanPaste](#)
[PDActionCopy](#)
[PDActionDestroyClipboardData](#)
[PDActionPaste](#)

[PDAnnotCanCopy](#)
[PDAnnotCanPaste](#)
[PDAnnotCopy](#)
[PDAnnotDestroyClipboardData](#)
[PDAnnotGetOCMD](#)
[PDAnnotIsCurrentlyVisible](#)
[PDAnnotPaste](#)
[PDAnnotRemoveOCMD](#)
[PDAnnotSetOCMD](#)

[PDCharProcEnumWithParams](#)
[PDCryptAuthorizeFilterAccess](#)

[PDDocCreateWordFinderEx](#)
[PDDocEnumOCConfigs](#)

PDDocEnumOCGs
PDDocFindPageNumForLabelEx
PDDocFlattenOC
PDDocGetLabelForPageNumEx
PDDocGetNumOCGs
PDDocGetOCConfig
PDDocGetOCContext
PDDocGetOCGs
PDDocGetTrapped
PDDocGetXAPMetadataProperty
PDDocHasOC
PDDocReadAheadEmbeddedFile
PDDocReplaceOCG
PDDocRequestEntireFile
PDDocRequestPages
PDDocSetNewCryptFilterData
PDDocSetNewCryptFilterMethod
PDDocSetNewCryptHandlerEx
PDDocSetNewDefaultFilters
PDDocSetTrapped
PDDocSetXAPMetadataProperty
PDDrawCosObjWithParams
PDFontGetASTextName
PDFormEnumPaintProcWithParams
PDOCConfigCreate
PDOCConfigDestroy
PDOCConfigGetAllRadioButtonGroups
PDOCConfigGetCosObj
PDOCConfigGetCreator
PDOCConfigGetInitState
PDOCConfigGetIntent
PDOCConfigGetName
PDOCConfigGetOCGOrder
PDOCConfigGetPDDoc
PDOCConfigGetRadioButtonGroupForOCG
PDOCConfigMakeRadioButtonGroup
PDOCConfigSetCreator
PDOCConfigSetInitState
PDOCConfigSetIntent
PDOCConfigSetName
PDOCConfigSetOCGOrder
PDOCContextApplyAutoStateChanges
PDOCContextClearAllUserOverrides
PDOCContextContentIsVisible
PDOCContextFindAutoStateChanges
PDOCContextFree

PDODContextGetIntent
PDODContextGetNonOCDrawing
PDODContextGetOCDrawEnumType
PDODContextGetOCGStates
PDODContextGetPDDoc
PDODContextInit
PDODContextMakeCopy
PDODContextMakeCopyWithAutoStateChanges
PDODContextNew
PDODContextNewWithInitialState
PDODContextNewWithOCDisabled
PDODContextPopOCMD
PDODContextPushOCMD
PDODContextResetOCMDStack
PDODContextSetIntent
PDODContextSetNonOCDrawing
PDODContextSetOCGStates
PDODContextXObjectIsVisible
PDODCGCreate
PDODCGCreateFromCosObj
PDODCGDestroy
PDODCGGetCosObj
PDODCGGetCurrentState
PDODCGGetFromCosObj
PDODCGGetInitialState
PDODCGGetIntent
PDODCGGetName
PDODCGGetPDDoc
PDODCGGetUsageEntry
PDODCGGetUserOverride
PDODCGHasUsageInfo
PDODCGRemoveInitialState
PDODCGSetCurrentState
PDODCGSetInitialState
PDODCGSetIntent
PDODCGSetName
PDODCGSetUsageDictEntry
PDODCGUsedInOCConfig
PDODCGUsedInOCContext
PDODCMDCreate
PDODCMDFindOrCreate
PDODCMDGetCosObj
PDODCMDGetFromCosObj
PDODCMDGetOCGs
PDODCMDGetPDDoc
PDODCMDGetVisPolicy
PDODCMDIsCurrentlyVisible

PDOCMDsAreCurrentlyVisible
PDOCMDsMakeContentVisible

PDPageDrawContentsPlacedWithParams
PDPageDrawContentsWithParams
PDPageEnumContents
PDPageEnumInks
PDPageEnumOCGs
PDPageFlattenOC
PDPageGetOCGs
PDPageGetVisibleBBox
PDPageMakeSeparations
PDPageSetPDEContentCanRaise

PDRegisterActionHandler

PDWordCreateTextSelect
PDWordGetASText
PDWordGetAttrEx
PDWordGetByteIdxFromHiliteChar
PDWordGetCharEncFlags
PDWordGetCharOffsetEx
PDWordGetCharQuad
PDWordGetNumHiliteChar
PDWordIsCurrentlyVisible
PDWordMakeVisible

PDWordFinderAcquireVisibleWordList
PDWordFinderEnumVisibleWords
PDWordFinderEnumWordsStr

Available only if `PI_PDFEDIT_READ_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

PDEElementGetAllVisibilities
PDEElementGetOCMD
PDEElementIsCurrentlyVisible

PDEFontGetSysEncoding
PDEFontGetSysFont
PDEFontIsEmbedded

PDETextGetAdvance
PDETextGetItem

PDETextItemGetFont
PDETextItemGetGState
PDETextItemGetTextLen
PDETextItemGetTextMatrix
PDETextItemGetTextState

Available only if `PI_PDFEDIT_WRITE_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

`PDEContentFlattenOC`

`PDEElementMakeVisible`

`PDEElementRemoveOCMD`

`PDEElementSetOCMD`

`PDEFontSetSysEncoding`

`PDEFontSetSysFont`

`PDEFormCreateClone`

`PDEFormSetContent`

`PDETextAddItem`

`PDETextRemoveItems`

`PDETextItemCopyText`

`PDETextItemCreate`

`PDETextItemRemoveChars`

`PDETextItemReplaceChars`

`PDETextItemReplaceText`

`PDETextItemSetFont`

`PDETextItemSetGState`

`PDETextItemSetTextMatrix`

`PDETextItemSetTextState`

`PDSysEncodingCreateFromCodePage`

Available only if `PI_PDS_READ_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

`PDSAttrObjGetCosObj`

`PDSElementGetCosObj`

`PDSElementGetKidWithMCInfo`

`PDSMCGetInfo`

`PDSMCGetPDEContainer`

`PDSMCIDGetParent`

`PDSTreeRootReplaceStreamRef`

Available only if `PI_PDS_WRITE_VERSION` (in `PIRequir.h`) is set to `0x00060000` or higher.

`PDSElementInsertStmMCAsKid`

Methods with Cosmetic Type Changes

Acrobat 6.0 introduced named cover types that are used in place of generic numeric types in methods and data structures. Such methods can be compiled without error if they still use the generic types, but in future the new types may be conditionally compiled

(depending on the Acrobat version) to 32-bit values. It is recommended that you update your code to use the new named types.

NOTE: New coordinate numeric types such as [AVDevCoord](#) are conditionally compiled as [ASInt16](#) or [ASInt32](#) depending on the Acrobat version level. These types are used in other structures, such as [AVRect](#). Methods that use these types are listed in [New and Modified Methods](#).

The following methods use these named cover types.

[ASCabDetachBinary](#)
[ASCabGetBinary](#)
[ASCabGetBinaryCopy](#)
[ASCabNumEntries](#)
[ASCabPutBinary](#)
[ASExtensionGetFileName](#)
[ASGetErrorString](#)
[ASGetExceptionErrorCode](#)
[ASFileClose](#)
[ASFileGetEOF](#)
[ASFileGetPos](#)
[ASFileHardFlush](#)
[ASFilePushData](#)
[ASFileRead](#)
[ASFileReopen](#)
[ASFileSetPos](#)
[ASFileWrite](#)
[ASFileSysCreateFolder](#)
[ASFileSysFlushVolume](#)
[ASFileSysGetItemProps](#)
[ASFileSysGetNameFromPath](#)
[ASFileSysOpenFile](#)
[ASFileSysRemoveFile](#)
[ASFileSysRemoveFolder](#)
[ASStmRead](#)
[ASStmWrite](#)
[ASTextGetUnicode](#)
[ASTextGetUnicodeCopy](#)
[ASTextFromSizedEncoded](#)
[ASTextFromSizedPDText](#)
[ASTextFromSizedScriptText](#)
[ASTextFromSizedUnicode](#)
[ASTextFromUnicode](#)
[ASTextSetSizedEncoded](#)
[ASTextSetSizedPDText](#)
[ASTextSetSizedScriptText](#)

ASTextSetSizedUnicode
ASTextSetUnicode
ASRaise
ASRegisterErrorString

AVAcquireSpecialFolderPathName
AVAppChooseFolderDialog
AVAppGetNumDocs
AVAppGetVersion
AVAppOpenDialog
AVAppRegisterForPageViewKeyDown
AVAppRegisterTool
AVAppSaveDialog
AVExtensionAcquireInfo
AVExtensionGetNumPlugIns
AVDocGetClientName
AVMenubarGetNumMenus
AVMenuGetNumMenuItems
AVMenuGetTitle
AVMenuItemGetTitle
AVMenuItemNew
AVPageViewAppearanceGetAVMatrix
AVPageViewDeviceToInfo
AVPageViewDrawPolygon
AVPageViewDrawPolygonOutline
AVPageViewFilterKeyDownForFocusAnnot
AVPageViewGetFirstVisiblePageNum
AVPageViewGetLastVisiblePageNum
AVPageViewGetNextView
AVPageViewGetPageNum
AVPageViewGetThreadIndex
AVPageViewGetSelectedAnnotPageNum
AVPageViewGetVisibleAnnotPage
AVPageViewGoTo
AVPageViewInfoToDevice
AVPageViewInvertRect
AVPageViewInvertRectOutline
AVPageViewPageNumIsVisible
AVPageViewSetPageNum
AVToolBarGetNumButtons
AVUtilGetBaseNameAndExtensionByPathName
AVUtilGetBaseNameAndExtensionByString

[CosArrayGet](#)
[CosArrayInsert](#)
[CosArrayLength](#)
[CosArrayPut](#)
[CosArrayRemoveNth](#)
[CosCopyStringValue](#)
[CosDecryptData](#)
[CosDocCreate](#)
[CosDocGetID](#)
[CosDocSaveToFile](#)
[CosEncryptData](#)
[CosNewArray](#)
[CosNewDict](#)
[CosNewString](#)
[CosStreamLength](#)
[CosStreamPos](#)
[CosStringValue](#)
[CosStringValueSafe](#)

New and Modified Callbacks

The following callbacks were added or changed in version 6.0.

[ASConstCabEnumProc](#)
[ASCryptStmFCloseProc](#)
[ASCryptStmFFlushProc](#)
[ASCryptStmFilBufProc](#)
[ASCryptStmFlsBufProc](#)
[ASCryptStmFPutEOFProc](#)
[ASCryptStmFResetProc](#)
[ASCryptStmUnGetcProc](#)
[ASFileSysAcquirePlatformPathProc](#)
[ASFileSysAsyncReadProc](#)
[ASFileSysAsyncWriteProc](#)
[ASFileSysCanPerformOpOnItemProc](#)
[ASFileSysCanSetEofProc](#)
[ASFileSysCloseProc](#)
[ASFileSysCreateFolderProc](#)
[ASFileSysDIPathFromPathExProc](#)
[ASFileSysDisplayASTextFromPathProc](#)
[ASFileSysFlushProc](#)
[ASFileSysFlushVolumeProc](#)
[ASFileSysGetEofProc](#)
[ASFileSysGetItemPropsProc](#)

ASFileSysGetItemPropsAsCabProc
ASFileSysGetNameAsASTextProc
ASFileSysGetNameProc
ASFileSysGetPosProc
ASFileSysOpenProc
ASFileSysPathFromDIPathProc
ASFileSysPerformOpOnItemProc
ASFileSysRangeArrivedProc
ASFileSysReadProc
ASFileSysReleasePlatformPathProc
ASFileSysRemoveFolderProc
ASFileSysRemoveProc
ASFileSysRenameProc
ASFileSysReopenProc
ASFileSysSetEofProc
ASFileSysSetPosProc
ASFileSysWriteProc
ASFileSysYieldProc

AVActionDoPropertiesExProc
AVActionGetDetailsProc
AVActionPerformExProc
AVAnnotHandlerAdjustCursorExProc
AVAnnotHandlerAppearanceDrawingProc
AVAnnotHandlerDoClickExProc
AVAnnotHandlerDoClickProc
AVAnnotHandlerDoKeyDownExProc
AVAnnotHandlerDoKeyDownProc
AVAnnotHandlerDoPropertiesExProc
AVAnnotHandlerGetAppearanceProc
AVAnnotHandlerGetFlagsProc
AVAnnotHandlerGetLayerExProc
AVAnnotHandlerPtInAnnotViewBBoxProc
AVComputeVisibleProc
AVContextMenuAdditionProc
AVConversionConvertStreamFromStructNodeProc
AVConversionConvertStreamToPDFProc
AVDocSelectionAcquireQuadsProc
AVDocSelectionHighlightSelectionExProc
AVDocSelectionKeyDownProc
AVDocSelectionShowMenuProc
AVIconHandlerGetFlagsProc
AVIconHandlerMeasureProc
AVIconHandlerOpenStmProc

AVIconHandlerReleaseProc
AVNotifyTooltipProc
AVPageViewClickProc
AVPageViewCursorProc
AVPageViewKeyDownProc
AVToolDestroyProc
AVToolGetLabelProc
AVToolGetLabelIconProc
AVUndoBeginEndProc
AVUndoGetTitleProc
AVUndoExecuteProc
AVUndoReleaseProc
AVUndoVerifyProc
AVWindowAdjustCursorProc
AVWindowMouseDownProc
DoClickProcType
DoKeyDownProcType
HFTServerProvideHFTPProc

PDActionHandlerCanCopyProc
PDActionHandlerCanPasteProc
PDActionHandlerCopyProc
PDActionHandlerDestroyProc
PDActionHandlerDestroyDataProc
PDActionHandlerGetTypeProc
PDActionHandlerPasteProc
PDAnnotHandlerCanCopyProc
PDAnnotHandlerCanPasteProc
PDAnnotHandlerCopyProc
PDAnnotHandlerDestroyProc
PDAnnotHandlerDestroyDataProc
PDAnnotHandlerGetHeelPointProc
PDAnnotHandlerGetPrintAppearanceProc
PDAnnotHandlerPasteProc
PDCryptFilterAuthorizeProc
PDCryptFilterGetDataProc
PDCryptFilterStreamProc
PDCryptFilterStringProc
PDCryptGetDocPermsProc
PDDocPreWriteProc
PDDocRequestEntireFileProc
PDDocRequestPagesProc
PDOCConfigEnumProc
PDOCGEnumProc

Callbacks with Cosmetic Type Changes

Acrobat 6.0 introduced named cover types that are used in place of generic numeric types in methods and data structures. Such methods can be compiled without error if they still use the generic types. The following callback methods use these named cover types.

`CosDocEnumEOFsProc`
`CosObjOffsetProc`

`ASFileCompletionProc`
`ASFileSysGetNameProc`
`ASFileSysMReadRequestProc`
`ASStmProc`

`AVActionGetButtonTextProc`
`AVActionGetInstructionsProc`
`AVActionGetStringOneTextProc`
`AVActionGetStringTwoTextProc`
`AVDocSelectionGetAVRectProc`
`AVSelectionPageRangeEnumProc`
`AVSelectionPageRangeEnumProc`

`PDAnnotHandlerGetAnnotInfoFlagsProc`

New and Modified Data

The following data structures were added or changed in version 6.0.

`ASCoord`
`ASCabValueType`
`ASCalendarTimeSpan`
`ASConstCab`
`ASConstText`
`ASCryptStm`
`ASCryptStmProcs`
`ASDateTimeFormat`
`ASFileSys`
`ASTextFilterType`
`ASTimeRec`
`ASTVersion`
`ASUUID`
`ASUUIDMaxStringLen`

`AVActionHandlerProcs`
`AVAdjustCursorParams`
`AVAlertParams`
`AVAlertType`
`AVAnnotHandler`
`AVAnnotOpData`

AVAppLanguageFormat
AVAppLanguageParams
AVAppUUIDType
AVClickParams
AVColorForcing
AVConversionFromPDFHandler
AVConversionMimeTypeString
AVConversionToPDFHandler
AVDevCoord
AVDevRect
AVDevSize
AVDocOpenParams
AVDocPrintOverrideData
AVDocPrintParams
AVDocPrintRasterizeData
AVDocPrintSepsParams
AVDocPrintTileData
AVDocServerType
AVDocViewDef
AVFlagBits16
AVIconBundle6
AVIconColorFormat
AVIconData
AVIconDataFormat
AVIconHandler
AVInkValue
AVlCoord
AVLine
AVOpenSaveDialogParams
AVPageSize
AVResourcePolicy
AVRasterizeFlags
AVRect
AVScreenCoord
AVScreenRect
AVSpecialFolder
AVStructNode
AVSysIconType
AVTileMark
AVTool
AVUndoHandler
AVUndoHandlerData
AVUseValue

AVWindowCoord
AVWindowRect
Code Page Values
CosByte
CosByteMax
CosDocOpenParams
CosDocSaveFlags
CString_Ptr
FSRef_Ptr
FSRefWithCFStringRefRec_Ptr
FSSpec_Ptr
HFTData
PDAActionClipboardData
PDAActionHandler
PDAActionHandlerData
PDAnnotClipboardData
PDAnnotHandler
PDAnnotPrintOp
PDCryptHandler
PDCryptFilterHandler
PDDocOCChangeType
PDDocRequestReason
PDDocSaveParams
PDDrawParams
PDGraphicEnumParams
PDHostSepsPlate
PDHostSepsSpec
PDOCConfigBaseState
PDOCContextChangeType
PDOCContextChangeType
PDOCDrawEnumType
PD CMDVisPolicy
PDPageMarkFlags
PDPrintWhat
PDSaveFlags2
PDSMCInfo
PDWordFinderConfig
POSIXPath_Ptr
Quad
Word Attributes
WordFinder Character Encoding Flags
WordFinder Sort Order Flags

Data with Cosmetic Type Changes

Acrobat 6.0 introduced named cover types that are used in place of generic numeric types in methods and data structures. Such methods can be compiled without error if they still use the generic types. The following are the named cover types, as well as the structures that use them but have not changed in any other way.

ASCount
ASErrorCode
ASIORequest
ASTArraySize
ASTCount
ASTFilePos
ASUnicodeChar
ASUTF16Val

AVArraySize
AVDestInfo
AVFilterIndex
AVFlagBits32
AVMenuIndex
AVPageIndex
AVPixelOffset
AVTArraySize
AVTBufferSize
AVTCount
AVTFlagBits
AVTFlagBits16
AVTSmallArraySize
AVTVersionNumPart

CosDocOpenParams
CosDocOpenParams
CosDocSaveParams

PDAnnotInfo
PDCount
PDDocInsertPagesParams
PDFontAngle
PDFontMetrics
PDFontOffset
PDGraphicState
PDiFontMetric
PDImageAttrs
PDImageScalar
PDPageNumber
PDSmallFlagBits



New Notifications

The following notifications were added or changed in version 6.0.

AVAppDidInitExtensions
AVAppSystemLogonSessionSwitched
AVAppToolDidChange
AVAppWillTerminateExtensions
AVDocActivePageViewDidChange
AVDocDidClearSelection
AVPageViewDocDidChange
AVPageViewWasCreated
AVPageViewWillDestroy

PDBookmarkDidChangeOpenState
PDDocOCDidChange
PDDocOCWillChange
PDDocPermsReady
PDDocWillPrintDocInMode

PDOCContextDidChange
PDOCContextWillChange

PDPageDidRemoveAnnotEx

Index of Methods

A

ASAtomExistsForString 158
ASAtomFromString 159
ASAtomGetString 160
ASCabCopy 161
ASCabDestroy 162
ASCabDestroyEmpties 163
ASCabDetachBinary 164
ASCabDetachCab 165
ASCabDetachPathName 166
ASCabDetachPointer 167
ASCabDetachString 169
ASCabDetachText 170
ASCabDup 171
ASCabEnum 172
ASCabEqual 173
ASCabFromEntryList 174
ASCabGetAtom 175
ASCabGetBinary 176
ASCabGetBinaryCopy 177
ASCabGetBool 178
ASCabGetCab 179
ASCabGetDouble 180
ASCabGetInt 181
ASCabGetPathNameCopy 182
ASCabGetPointer 183
ASCabGetPointerDestroyProc 184
ASCabGetPointerType 185
ASCabGetString 186
ASCabGetStringCopy 187
ASCabGetText 188
ASCabGetType 189
ASCabGetUns 190
ASCabKnown 191
ASCabMakeEmpty 192
ASCabMunge 193
ASCabNew 194
ASCabNumEntries 195
ASCabPutAtom 196
ASCabPutBinary 197
ASCabPutBool 198
ASCabPutCab 199
ASCabPutDouble 200
ASCabPutInt 201
ASCabPutNull 202
ASCabPutPathName 203
ASCabPutPointer 204
ASCabPutString 205
ASCabPutText 206
ASCabPutUns 207
ASCabReadFromStream 208
ASCabRemove 210
ASCabRename 211
ASCabValueEqual 212
ASCabWriteToStream 213
ASCalendarTimeSpanAddWithBase 384
ASCalendarTimeSpanCompare 385
ASCalendarTimeSpanDiff 386
ASCallbackCreate 216
ASCallbackCreateNotification 218
ASCallbackCreateProto 2176
ASCallbackCreateReplacement 219
ASCallbackDestroy 220
ASCancelProc 2410
ASConstCabEnum 215
ASCStringToFixed 410
ASDateAddCalendarTimeSpan 221
ASDateAddTimeSpan 222
ASDateCalendarDiff 223
ASDateClear 224
ASDateCompare 225
ASDateCopy 226
ASDateDestroy 227
ASDateDup 228
ASDateExactDiff 229
ASDateGetLocalTime 230
ASDateGetTimeString 231
ASDateGetUTCTime 232
ASDateNew 233



ASDataSetLocalTimeOffset 234
 ASDataSetTimeFromRec 235
 ASDataSetTimeFromString 236
 ASDataSetToCurrentLocalTime 237
 ASDataSetToCurrentUTCTime 238
 ASDateSubtractCalendarTimeSpan 239
 ASDateSubtractTimeSpan 240
 ASEnumExtensions 241
 ASExtensionGetFileName 242
 ASExtensionGetRegisteredName 243
 ASExtensionMgrGetHFT 421
 ASFileAcquirePathName 244
 ASFileCanSetEOF 245
 ASFileClose 246
 ASFileFlush 247
 ASFileFromMDFile 248
 ASFileGetEOF 249
 ASFileGetFileSys 250
 ASFileGetFileSysByName 266
 ASFileGetMDFile 251
 ASFileGetOpenMode 252
 ASFileGetPos 253
 ASFileGetURL 254
 ASFileHardFlush 255
 ASFileHasOutstandingMReads 256
 ASFileIsSame 257
 ASFilePushData 258
 ASFileRead 259
 ASFileRegisterFileSys 267
 ASFileReopen 260
 ASFileSetEOF 261
 ASFileSetMode 262
 ASFileSetPos 264
 ASFileStmRdOpen 323
 ASFileStmWrOpen 325
 ASFileSysAcquireFileSysPath 268
 ASFileSysAcquireParent 270
 ASFileSysAcquirePlatformPath 271
 ASFileSysCanPerformOpOnItem 273
 ASFileSysConvertItemPropsToCab 274, 275
 ASFileSysCopyPath 277
 ASFileSysCreateFolder 278
 ASFileSysCreatePathName 279
 ASFileSysDestroyFolderIterator 282
 ASFileSysDIPathFromPath 285
 ASFileSysDIPathFromPathEx 287
 ASFileSysDisplayASTextFromPath 283
 ASFileSysDisplayStringFromPath 284
 ASFileSysFirstFolderItem 288
 ASFileSysFlushVolume 290
 ASFileSysGetItemProps 291
 ASFileSysGetItemPropsAsCab 292
 ASFileSysGetNameFromPath 294
 ASFileSysGetNameFromPathAsASText 295
 ASFileSysGetStorageFreeSpace 296
 ASFileSysGetTempPathName 297
 ASFileSysGetTypeAndCreator 298
 ASFileSysNextFolderItem 299
 ASFileSysOpenFile 301
 ASFileSysPathFromDIPath 303, 305
 ASFileSysPerformOpOnItem 307
 ASFileSysReleasePath 308
 ASFileSysReleasePlatformPath 309
 ASFileSysRemoveFile 310
 ASFileSysRemoveFolder 311
 ASFileSysSetTypeAndCreator 312
 ASFileSysURLFromPath 313
 ASFileUnregisterFileSys 314
 ASFileWrite 265
 ASFixedDiv 411
 ASFixedMatrixConcat 412
 ASFixedMatrixInvert 414
 ASFixedMatrixTransform 415
 ASFixedMatrixTransformRect 417
 ASFixedMul 419
 ASFixedToCString 420
 ASfree 436
 ASGetConfiguration 155
 ASGetDefaultFileSys 315
 ASGetErrorString 404
 ASGetExceptionErrorCode 405
 ASGetSecs 157
 ASHostMBLen 439
 ASIsValidUTF8 441
 ASmalloc 437
 ASMemStmRdOpen 327
 ASPathFromPlatformPath 316
 ASPlatformPathGetCFURLRefRecPtr 317



ASPlatformPathGetCStringPtr 318, 322
 ASPlatformPathGetFSRefPtr 319
 ASPlatformPathGetFSRefWithCStringRefRecPtr 320
 ASPlatformPathGetFSSpecPtr 321
 ASPopExceptionFrame 406
 ASProcStmRdOpen 328
 ASProcStmRdOpenEx 329
 ASProcStmWrOpen 330
 ASPushExceptionFrame 407
 ASRaise 408
 ASrealloc 438
 ASRegisterErrorString 409
 ASScriptFromHostEncoding 442
 ASScriptToHostEncoding 443
 ASStmClose 332
 ASStmFlush 333
 ASStmRead 334
 ASStmWrite 335
 ASTextCaseSensitiveCmp 336
 ASTextCat 337
 ASTextCatMany 338
 ASTextCmp 339
 ASTextCopy 340
 ASTextDestroy 341
 ASTextDup 342
 ASTextEval 343
 ASTextFilter 344
 ASTextFromEncoded 345
 ASTextFromInt32 346
 ASTextFromPDText 347
 ASTextFromScriptText 348
 ASTextFromSizedEncoded 349
 ASTextFromSizedPDText 350
 ASTextFromSizedScriptText 351
 ASTextFromSizedUnicode 352
 ASTextFromUnicode 353
 ASTextFromUns32 354
 ASTextGetBestEncoding 355
 ASTextGetBestScript 357
 ASTextGetCountry 358
 ASTextGetEncoded 359
 ASTextGetEncodedCopy 360
 ASTextGetLanguage 361
 ASTextGetPDTextCopy 362
 ASTextGetScriptText 363
 ASTextGetScriptTextCopy 364
 ASTextGetUnicode 365
 ASTextGetUnicodeCopy 366
 ASTextIsEmpty 367
 ASTextMakeEmpty 368
 ASTextNew 369
 ASTextNormalizeEndOfLine 370
 ASTextReplace 371
 ASTextReplaceASCII 372
 ASTextReplaceBadChars 373
 ASTextSetCountry 374
 ASTextSetEncoded 375
 ASTextSetLanguage 376
 ASTextSetPDText 377
 ASTextSetScriptText 378
 ASTextSetSizedEncoded 379
 ASTextSetSizedPDText 380
 ASTextSetSizedScriptText 381
 ASTextSetSizedUnicode 382
 ASTextSetUnicode 383
 ASTimeSpanAdd 387
 ASTimeSpanCompare 388
 ASTimeSpanCopy 389
 ASTimeSpanDestroy 390
 ASTimeSpanDiff 391
 ASTimeSpanDup 392
 ASTimeSpanGetASInt32 393
 ASTimeSpanNegate 394
 ASTimeSpanNew 395
 ASTimeSpanSet 396
 ASTimeSpanSetFromASInt32 397
 ASTimeSpanSetFromString 398
 ASUUIDFromCString 399
 ASUUIDGenFromHash 400
 ASUUIDGenFromName 401
 ASUUIDGenUnique 402
 ASUUIDToCString 403
 AVAcquireSpecialFilePathName 445
 AVAcquireSpecialFilePathNameWithASText 447
 AVAcquireSpecialFolderPathName 449
 AVActionHandlerGetProcs 460
 AVActionHandlerGetType 462
 AVActionHandlerGetUIName 463

AVAlert 464
 AVAlertConfirm 466
 AVAlertGetPref 467
 AVAlertNote 468
 AVAlertResetPrefs 469
 AVAlertSetPref 470
 AVAlertWithParams 471
 AVAnnotHandlerDeleteInfo 472
 AVAnnotHandlerGetInfo 473
 AVAppAutoShowHowToPanel 474
 AVAppBeginFullScreen 475
 AVAppBeginModal 476
 AVAppCanQuit 478
 AVAppChooseFolderDialog 479
 AVAppCreateIconBundle6 480
 AVAppDidOrWillSwitchForDialog 2127
 AVAppDoingFullScreen 481
 AVAppEndFullScreen 482
 AVAppEndModal 483
 AVAppEnumActionHandlers 484
 AVAppEnumAnnotHandlers 486
 AVAppEnumDocs 487
 AVAppEnumSystemFonts 2128
 AVAppEnumTools 488
 AVAppEnumTransHandlers 489
 AVAppFindCommandHandlerByName 490
 AVAppFindGlobalCommandByName 491
 AVAppGetActionHandlerByType 492
 AVAppGetActiveDoc 493
 AVAppGetActiveTool 495
 AVAppGetAnnotHandlerByName 496
 AVAppGetCancelProc 497
 AVAppGetDefaultTool 498
 AVAppGetDocProgressMonitor 499
 AVAppGetHowToPanelAutoShow 501
 AVAppGetLanguage 502
 AVAppGetLanguageEncoding 503
 AVAppGetLanguageWithParams 504
 AVAppGetLastActiveTool 505
 AVAppGetMenubar 506
 AVAppGetName 507
 AVAppGetNumDocs 508
 AVAppGetPreference 509
 AVAppGetReportProc 510
 AVAppGetToolBar 511
 AVAppGetToolBarByName 512
 AVAppGetToolByName 513
 AVAppGetTransHandlerByType 514
 AVAppGetUUID 515
 AVAppGetVersion 516
 AVAppHandleAppleEvent 2129
 AVAppHandlePlatformEvent 517
 AVAppHelpSearch 518
 AVAppHelpShowContents 519
 AVAppHelpShowIndex 520
 AVAppIsIdle 521
 AVAppModalWindowsOpen 522
 AVAppOpenDialog 523
 AVAppOpenHelpFile 525, 526
 AVAppRegisterActionHandler 527
 AVAppRegisterAnnotHandler 529
 AVAppRegisterCommandHandler 531
 AVAppRegisterForContextMenuAddition 532
 AVAppRegisterForPageViewAdjustCursor 533
 AVAppRegisterForPageViewClicks 534
 AVAppRegisterForPageViewDrawing 536
 AVAppRegisterForPageViewKeyDown 538
 AVAppRegisterForPageViewRightClicks 539
 AVAppRegisterFromPDFHandler 540
 AVAppRegisterGlobalCommand 541
 AVAppRegisterHowToPanel 542
 AVAppRegisterIdleProc 544
 AVAppRegisterNotification 546
 AVAppRegisterTool 548
 AVAppRegisterToolBarPosition 550
 AVAppRegisterToPDFHandler 551
 AVAppRegisterTransHandler 552
 AVAppSaveDialog 553
 AVAppSetActiveTool 555
 AVAppSetHowToPanelAutoShow 557
 AVAppSetHowToPanelAutoShowText 558
 AVAppSetHowToPanelComputeVisibleProc 559
 AVAppSetPreference 560
 AVAppUnregisterForPageViewAdjustCursor 561
 AVAppUnregisterForPageViewClicks 562
 AVAppUnregisterForPageViewDrawing 563
 AVAppUnregisterForPageViewDrawingEx 564
 AVAppUnregisterForPageViewKeyDown 565



AVAppUnregisterForPageViewRightClicks 566
 AVAppUnregisterGlobalCommand 567
 AVAppUnregisterIdleProc 568
 AVAppUnregisterNotification 569
 AVAppYieldToOtherApps 570
 AVAuthOpen 617
 AVCommandCancel 574
 AVCommandDestroy 575
 AVCommandExecute 576
 AVCommandGetAVDoc 578
 AVCommandGetCab 579
 AVCommandGetCancelProc 580
 AVCommandGetConfig 581
 AVCommandGetInputs 583
 AVCommandGetName 585
 AVCommandGetParams 586
 AVCommandGetPDDoc 587
 AVCommandGetProgressMonitor 588
 AVCommandGetProps 589
 AVCommandGetReportProc 591
 AVCommandGetStatus 592
 AVCommandGetUIPolicy 593
 AVCommandNew 594
 AVCommandPutCab 595
 AVCommandReset 596
 AVCommandSetConfig 597
 AVCommandSetInputs 598
 AVCommandSetParams 601
 AVCommandShowDialog 604
 AVCommandWork 605
 AVConversionConvertFromPDFWithHandler 606
 AVConversionConvertStreamFromPDFWithHandler 608
 AVConversionConvertStreamFromStructNodeWithHandler 609
 AVConversionConvertStreamToPDF 610
 AVConversionConvertStreamToPDFWithHandler 612
 AVConversionConvertToPDF 613
 AVConversionConvertToPDFWithHandler 614
 AVConversionEnumFromPDFConverters 615
 AVConversionEnumToPDFConverters 616
 AVCryptDoStdSecurity 618
 AVCryptGetPassword 619
 AVDestInfoDestroy 450
 AVDocAlert 620
 AVDocAlertConfirm 622
 AVDocAlertNote 623
 AVDocAlertYesNo 624
 AVDocBeginUndoOperation 625, 645
 AVDocClearSelection 626
 AVDocClearUndos 628
 AVDocClose 629
 AVDocCopyAction 630
 AVDocCopyActionCommon 631
 AVDocCopyAdditionalActions 632
 AVDocCopyAnnot 633
 AVDocCopyAnnotCommon 634
 AVDocCopySelection 635
 AVDocDeleteSelection 637
 AVDocDoActionPropsDialog 638
 AVDocDoCopyAs 639
 AVDocDoPrint 640
 AVDocDoSave 641
 AVDocDoSaveAs 642
 AVDocDoSaveAsWithParams 643
 AVDocDoSelectionProperties 644
 AVDocEnumSelection 646
 AVDocFromPDDoc 648
 AVDocGetActiveTool 649
 AVDocGetAVWindow 650
 AVDocGetClientName 651
 AVDocGetNthPageView 652
 AVDocGetNumPageViews 653
 AVDocGetPageText 654
 AVDocGetPageView 656
 AVDocGetPDDoc 657
 AVDocGetSelection 658
 AVDocGetSelectionServerByType 659
 AVDocGetSelectionType 660
 AVDocGetServerType 661
 AVDocGetSplitterPosition 662
 AVDocGetTopUndo 663
 AVDocGetTopUndoAndRedo 664
 AVDocGetViewDef 665
 AVDocGetViewDefEx 666
 AVDocGetViewMode 667
 AVDocIsDead 668
 AVDocIsExternal 669



AVDocIsReadOnly 670
 AVDocIsSlow 671
 AVDocOpenFromASFileWithParams 672
 AVDocOpenFromASFileWithParamString 674
 AVDocOpenFromFile 676
 AVDocOpenFromFileWithParams 678
 AVDocOpenFromFileWithParamString 680
 AVDocOpenFromPDDoc 682
 AVDocOpenFromPDDocWithParams 684
 AVDocOpenFromPDDocWithParamString 686
 AVDocPerformAction 688
 AVDocPerformActionEx 690
 AVDocPermRequest 691
 AVDocPrintPages 692
 AVDocPrintPagesWithParams 694
 AVDocPrintSeparations 700
 AVDocRegisterSelectionServer 701
 AVDocSelectionEnumPageRanges 703
 AVDocSendAuxData 705
 AVDocSetActiveTool 706
 AVDocSetClientName 707
 AVDocSetDead 708
 AVDocSetReadOnly 709
 AVDocSetSelection 710
 AVDocSetSplitterPosition 714
 AVDocSetViewDef 715
 AVDocSetViewDefEx 716
 AVDocSetViewMode 717
 AVDocShowSelection 718
 AVExtensionAcquireInfo 451
 AVExtensionGetNumPlugIns 452
 AVExtensionReleaseInfo 453
 AVGrafSelectCreate 720
 AVGrafSelectDestroy 721
 AVGrafSelectGetBoundingRect 722
 AVHasAuxDataHandler 571
 AVIdentityGetText 454
 AVIdentitySetText 455
 AVMenuAcquire 723
 AVMenuAcquireMenuItemByIndex 725
 AVMenuAddMenuItem 727
 AVMenubarAcquireMenuByIndex 745
 AVMenubarAcquireMenuByName 747
 AVMenubarAcquireMenuByPredicate 749
 AVMenubarAcquireMenuItemByName 750
 AVMenubarAcquireMenuItemByPredicate 752
 AVMenubarAddHiddenMenu 753
 AVMenubarAddMenu 754
 AVMenubarGetMenuIndex 755
 AVMenubarGetNumMenus 756
 AVMenubarHide 757
 AVMenubarShow 758
 AVMenuClone 728
 AVMenuDoPopUp 729
 AVMenuGetMenuItemIndex 730
 AVMenuGetName 731
 AVMenuGetNumMenuItems 732
 AVMenuGetParentMenubar 733
 AVMenuGetParentMenuItem 734
 AVMenuGetTitle 735, 737, 771
 AVMenuIsHiddenOnMenubar 738
 AVMenuItemAcquire 759
 AVMenuItemAcquireSubmenu 761
 AVMenuItemClone 763
 AVMenuItemExecute 764
 AVMenuItemGetLongOnly 765
 AVMenuItemGetName 766
 AVMenuItemGetParentMenu 767
 AVMenuItemGetShortcut 768
 AVMenuItemGetTitle 769
 AVMenuItemIsEnabled 772
 AVMenuItemIsMarked 773
 AVMenuItemIsVisible 774
 AVMenuItemNew 775
 AVMenuItemNewWithASText 777
 AVMenuItemRelease 779
 AVMenuItemRemove 780
 AVMenuItemSetComputeEnabledProc 781
 AVMenuItemSetComputeMarkedProc 783
 AVMenuItemSetComputeVisibleProc 785
 AVMenuItemSetExecuteProc 786
 AVMenuItemSetTitle 788
 AVMenuItemSetTitleWithASText 790
 AVMenuNew 739
 AVMenuNewWithASText 741
 AVMenuRelease 743
 AVMenuRemove 744

AVPageViewAcquireMachinePort 791
 AVPageViewAppearanceGetAVMatrix 792
 AVPageViewBeginOperation 794
 AVPageViewClearFocusAnnot 795
 AVPageViewDevicePointToPage 796
 AVPageViewDeviceRectToPage 797
 AVPageViewDeviceRectToPageRZ 798
 AVPageViewDeviceToInfo 800
 AVPageViewDoPopupMenu 801
 AVPageViewDragOutNewRect 802
 AVPageViewDragOutNewRectSnapped 803
 AVPageViewDragRect 805
 AVPageViewDragRectSnapped 807
 AVPageViewDrawAnnotSequence 810
 AVPageViewDrawCosObj 811
 AVPageViewDrawCosObjEx 812
 AVPageViewDrawNow 814
 AVPageViewDrawNowWithTransition 815
 AVPageViewDrawPolygon 816
 AVPageViewDrawPolygonOutline 817
 AVPageViewDrawRect 818
 AVPageViewDrawRectOutline 820
 AVPageViewDrawRectOutlineWithHandles 821
 AVPageViewEndOperation 822
 AVPageViewFilterKeyDownForFocusAnnot 823
 AVPageViewFocusAnnotPerformOp 824
 AVPageViewGetActiveBead 825
 AVPageViewGetAnnotRect 826
 AVPageViewGetAperture 828
 AVPageViewGetAVDoc 829
 AVPageViewGetColor 830
 AVPageViewGetDevToPageMatrix 831
 AVPageViewGetFirstVisiblePageNum 833
 AVPageViewGetFocusAnnot 832
 AVPageViewGetGrayRect 834
 AVPageViewGetLastVisiblePageNum 835
 AVPageViewGetLayoutMode 836
 AVPageViewGetMousePosition 838
 AVPageViewGetMousePositionSnapped 839
 AVPageViewGetNextView 841
 AVPageViewGetNumVisibleInks 837
 AVPageViewGetPage 842
 AVPageViewGetPageNum 843
 AVPageViewGetPageToDevMatrix 844
 AVPageViewGetPageToDevScaling 845
 AVPageViewGetPixelInformationAtPoint 846
 AVPageViewGetSelectedAnnotPageNum 847
 AVPageViewGetThreadIndex 848
 AVPageViewGetVisibleAnnotPage 849
 AVPageViewGetVisibleInks 850
 AVPageViewGetZoom 851
 AVPageViewGetZoomType 852
 AVPageViewGhostRectOutline 853
 AVPageViewGoBack 854
 AVPageViewGoForward 855
 AVPageViewGoTo 856
 AVPageViewHighlightText 857
 AVPageViewInfoToDevice 858
 AVPageViewInfoToPoint 859
 AVPageViewInsetRect 860
 AVPageViewInvalidateRect 861
 AVPageViewInvalidateText 863
 AVPageViewInvertQuad 864
 AVPageViewInvertRect 865
 AVPageViewInvertRectOutline 866
 AVPageViewIsAnnotAtPoint 868
 AVPageViewIsAnnotOfTypeAtPoint 870
 AVPageViewIsBeadAtPoint 871
 AVPageViewIsFocusAnnot 872
 AVPageViewPageNumIsVisible 873
 AVPageViewPointInText 874
 AVPageViewPointToDevice 875
 AVPageViewPointToInfo 876
 AVPageViewReadPageDown 877
 AVPageViewReadPageUp 878
 AVPageViewRectToDevice 879
 AVPageViewReleaseMachinePort 880
 AVPageViewResumeOffscreenDrawing 881
 AVPageViewScrollTo 882
 AVPageViewScrollToAnnot 883
 AVPageViewScrollToRect 884
 AVPageViewSetAnnotLocation 886
 AVPageViewSetColor 887
 AVPageViewSetFocusAnnot 888
 AVPageViewSetInkPreview 889
 AVPageViewSetLayoutMode 890
 AVPageViewSetPageNum 891
 AVPageViewSetVisibleInks 892

AVPageViewShowControl 893
 AVPageViewSnapPoint 894, 896
 AVPageViewSnapRect 898
 AVPageViewStartReadingThread 900
 AVPageViewSuspendOffscreenDrawing 901
 AVPageViewToDestInfo 902
 AVPageViewToViewDest 903
 AVPageViewTrackText 904
 AVPageViewTransformRectRZ 905
 AVPageViewUpdateInfoPanel 906
 AVPageViewUseDestInfo 908
 AVPageViewUseThisDestination 909
 AVPageViewZoomTo 910
 AVRectHandleHitTest 456
 AVRectToRect 2131
 AVRegisterAuxDataHandler 572
 AVSweetPeaGetBasicSuiteP 911
 AVSweetPeaGetPluginRef 912
 AVSweetPeaGetResourceAccess 913
 AVSweetPeasADMAvailable 914
 AVSweetPeaProcessADMEvent 915
 AVSweetPeaSetResourceAccess 916
 AVSysAllocTimeStringFromTimeRec 917
 AVSysBeep 918
 AVSysGetCursor 919
 AVSysGetIconFromFilename 920
 AVSysGetIconFromMimeType 921
 AVSysGetIconFromTypeAndCreator 922
 AVSysGetModifiers 923, 924
 AVSysGetStandardCursor 925
 AVSysMouselsStillDown 926
 AVSysSetCursor 927
 AVSysSetWaitCursor 928
 AVToolBarAddButton 931
 AVToolBarEnumButtons 933
 AVToolBarGetButtonByName 935
 AVToolBarGetFrame 936
 AVToolBarGetNumButtons 937
 AVToolBarIsRoomFor 938
 AVToolBarNew 939
 AVToolBarNewFlyout 940
 AVToolBarUpdateButtonStates 941
 AVToolButtonDestroy 942
 AVToolButtonExecute 943
 AVToolButtonGetFlyout 944
 AVToolButtonGetIcon 945
 AVToolButtonGetLabelText 946, 965
 AVToolButtonGetMenu 947
 AVToolButtonGetName 948
 AVToolButtonIsEnabled 949
 AVToolButtonIsMarked 950
 AVToolButtonIsSeparator 952
 AVToolButtonIsVisible 951
 AVToolButtonNew 953
 AVToolButtonRemove 955
 AVToolButtonSetComputeEnabledProc 956
 AVToolButtonSetComputeMarkedProc 957
 AVToolButtonSetComputeVisibleProc 959
 AVToolButtonSetExecuteProc 960
 AVToolButtonSetExternal 961
 AVToolButtonSetFlyout 962
 AVToolButtonSetHelpText 963
 AVToolButtonSetIcon 964
 AVToolButtonSetMenu 966
 AVToolButtonSetNotifyTooltipProc 967
 AVToolGetType 929
 AVToolsPersistent 930
 AVUnregisterAuxDataHandler 573
 AVUtilGetBaseNameAndExtensionByPathName 457, 459
 AVUtilGetBaseNameAndExtensionByString 458
 AVWindowBecomeKey 973
 AVWindowBringToFront 974
 AVWindowCenter 975
 AVWindowDestroy 976
 AVWindowDoModal 977
 AVWindowDrawNow 978
 AVWindowEndModal 979
 AVWindowEnsureInBounds 980
 AVWindowGetBorderWidths 981
 AVWindowGetCursorAtPoint 2132
 AVWindowGetDesktopBounds 982
 AVWindowGetFrame 983
 AVWindowGetInterior 984
 AVWindowGetMinMaxSize 985
 AVWindowGetOwnerData 986
 AVWindowGetPlatformThing 987
 AVWindowGetTitle 988

AVWindowHandlePlatformEvent 989
 AVWindowHide 990
 AVWindowInvalidateRect 991
 AVWindowIsKey 992
 AVWindowIsVisible 993
 AVWindowMaximize 994
 AVWindowNew 995
 AVWindowNewFromPlatformThing 997
 AVWindowResignKey 999
 AVWindowSetFrame 1000
 AVWindowSetMinMaxSize 1001
 AVWindowSetOwnerData 1002
 AVWindowSetTitle 1003
 AVWindowSetWantsKey 1004
 AVWindowShow 1005
 AVWindowUserClose 1006

C

CosArrayGet 1009
 CosArrayInsert 1010
 CosArrayLength 1011
 CosArrayPut 1012
 CosArrayRemove 1013
 CosArrayRemoveNth 1014
 CosBooleanValue 1016
 CosCopyStringValue 1086
 CosCryptGetVersion 1094
 CosDecryptData 1095
 CosDecryptGetMaxKeyBytes 1096
 CosDictGet 1018
 CosDictGetXAPMetadata 1020
 CosDictKnown 1021
 CosDictPut 1023
 CosDictRemove 1025
 CosDictSetXAPMetadata 1026
 CosDocClose 1029
 CosDocCreate 1030
 CosDocEnumEOFs 1031
 CosDocEnumIndirect 1033
 CosDocGetID 1035
 CosDocGetInfoDict 1036
 CosDocGetObjByID 1037
 CosDocGetRoot 1038
 CosDocHasFullCompression 1039
 CosDocHasPartialCompression 1040
 CosDocObjsWithinRange 1052
 CosDocOpenWithParams 1041
 CosDocSaveToFile 1042
 CosDocSaveWithParams 1043
 CosDocSetDirty 1044
 CosEncryptData 1097
 CosEncryptGetMaxKeyBytes 1098
 CosFixedValue 1045
 CosIntegerValue 1047
 CosNameValue 1049
 CosNewArray 1015
 CosNewBoolean 1017
 CosNewDict 1028
 CosNewFixed 1046
 CosNewInteger 1048
 CosNewName 1050
 CosNewNull 1051
 CosNewObjCollection 1077
 CosNewStream 1078
 CosNewString 1088
 CosObjAddToCollection 1053
 CosObjCmp 1054
 CosObjCollectionEnum 1073
 CosObjCollectionEqual 1074
 CosObjCollectionIsNull 1075
 CosObjCollectionSize 1076
 CosObjCopy 1056
 CosObjDestroy 1057
 CosObjEnum 1058
 CosObjEqual 1060
 CosObjGetCollection 1061
 CosObjGetCompressibility 1062
 CosObjGetDoc 1063
 CosObjGetGeneration 1064
 CosObjGetID 1065
 CosObjGetType 1066
 CosObjHash 1067
 CosObjsCompressed 1068
 CosObjsIndirect 1069
 CosObjRefreshAfterLinearization 1070
 CosObjRemoveFromCollection 1071
 CosObjSetCompressability 1072

CosSetMaxDocStorage 1008
 CosStreamDict 1082
 CosStreamLength 1083
 CosStreamOpenStm 1084
 CosStreamPos 1085
 CosStringGetHexFlag 1089
 CosStringSetHexFlag 1090
 CosStringValue 1091
 CosStringValueSafe 1093

H

HFTDestroy 422
 HFTGetReplacedEntry 423
 HFTGetVersion 424
 HFTIsValid 425
 HFTNew 426, 427
 HFTReplaceEntry 428
 HFTReplaceEntryEx 430
 HFTServerDestroy 432
 HFTServerNew 433
 HFTUnreplaceEntry 434

P

PDAActionCanCopy 1127
 PDAActionCanPaste 1128
 PDAActionCopy 1129
 PDAActionDestroy 1130
 PDAActionDestroyClipboardData 1131
 PDAActionEqual 1132
 PDAActionFromCosObj 1133
 PDAActionGetCosObj 1134
 PDAActionGetDest 1135
 PDAActionGetFileSpec 1136
 PDAActionGetSubtype 1137
 PDAActionIsValid 1138
 PDAActionNew 1139
 PDAActionNewFromDest 1140
 PDAActionNewFromFileSpec 1141
 PDAActionPaste 1142
 PDAnnotCanCopy 1144
 PDAnnotCanPaste 1145
 PDAnnotCopy 1146

PDAnnotDestroyClipboardData 1147
 PDAnnotEqual 1148
 PDAnnotFromCosObj 1149
 PDAnnotGetColor 1150
 PDAnnotGetCosObj 1152
 PDAnnotGetDate 1153
 PDAnnotGetFlags 1154
 PDAnnotGetOCMD 1155
 PDAnnotGetRect 1156
 PDAnnotGetSubtype 1157
 PDAnnotGetTitle 1158
 PDAnnotIsCurrentlyVisible 1160
 PDAnnotIsValid 1161
 PDAnnotNotifyDidChange 1162
 PDAnnotNotifyWillChange 1163
 PDAnnotPaste 1164
 PDAnnotRemoveOCMD 1165
 PDAnnotSetColor 1166
 PDAnnotSetDate 1168
 PDAnnotSetFlags 1169
 PDAnnotSetOCMD 1170
 PDAnnotSetRect 1171
 PDAnnotSetTitle 1173
 PDApplyFunction 1101
 PDBeadAcquirePage 1177
 PDBeadDestroy 1178
 PDBeadEqual 1179
 PDBeadFromCosObj 1180
 PDBeadGetCosObj 1181
 PDBeadGetIndex 1182
 PDBeadGetNext 1183
 PDBeadGetPrev 1184
 PDBeadGetRect 1185
 PDBeadGetThread 1186
 PDBeadInsert 1187
 PDBeadIsValid 1188
 PDBeadNew 1189
 PDBeadSetPage 1190
 PDBeadSetRect 1191
 PDBookmarkAddChild 1192
 PDBookmarkAddNewChild 1193
 PDBookmarkAddNewSibling 1194
 PDBookmarkAddNext 1195
 PDBookmarkAddPrev 1196



PDBookmarkAddSubtree 1197
 PDBookmarkDestroy 1198
 PDBookmarkEqual 1199
 PDBookmarkFromCosObj 1200
 PDBookmarkGetAction 1201
 PDBookmarkGetByTitle 1202
 PDBookmarkGetColor 1203
 PDBookmarkGetCosObj 1204
 PDBookmarkGetCount 1205
 PDBookmarkGetFirstChild 1206
 PDBookmarkGetFlags 1207
 PDBookmarkGetIndent 1208
 PDBookmarkGetLastChild 1209
 PDBookmarkGetNext 1210
 PDBookmarkGetParent 1211
 PDBookmarkGetPrev 1212
 PDBookmarkGetTitle 1213
 PDBookmarkHasChildren 1214
 PDBookmarkIsOpen 1215
 PDBookmarkIsValid 1216
 PDBookmarkRemoveAction 1217
 PDBookmarkSetAction 1218
 PDBookmarkSetColor 1219
 PDBookmarkSetFlags 1220
 PDBookmarkSetOpen 1221
 PDBookmarkSetTitle 1222
 PDBookmarkUnlink 1223
 PDCharProcEnum 1224
 PDCharProcEnumWithParams 1225
 PDCharProcGetCosObj 1226
 PDCryptAuthorizeFilterAccess 1227
 PDDocAcquire 1228
 PDDocAcquirePage 1229
 PDDocAddThread 1230
 PDDocAuthorize 1231
 PDDocCalculateImplicitMetadata 1233
 PDDocClearFlags 1234
 PDDocClose 1235
 PDDocCopyToFile 1236
 PDDocCreate 1237
 PDDocCreateNameTree 1238
 PDDocCreatePage 1239
 PDDocCreateStructTreeRoot 1240
 PDDocCreateTextSelect 1241
 PDDocCreateThumbs 1242
 PDDocCreateWordFinder 1245, 1248
 PDDocCreateWordFinderUCS 1250
 PDDocDeletePages 1252
 PDDocDeleteThumbs 1254
 PDDocEnumFonts 1255
 PDDocEnumLoadedFonts 1257
 PDDocEnumOCConfigs 1258
 PDDocEnumOCGs 1259
 PDDocEnumResources 1260
 PDDocExportNotes 1262
 PDDocExportSomeNotes 1263
 PDDocFindPageNumForLabel 1265, 1266
 PDDocFlattenOC 1267
 PDDocFromCosDoc 1268
 PDDocGetBookmarkRoot 1269
 PDDocGetCosDoc 1270
 PDDocGetCryptHandlerClientData 1271
 PDDocGetFile 1272
 PDDocGetFlags 1273
 PDDocGetFullScreen 1274
 PDDocGetID 1275
 PDDocGetInfo 1276
 PDDocGetLabelForPageNum 1278, 1279
 PDDocGetNameTree 1280
 PDDocGetNewCryptHandler 1281
 PDDocGetNewSecurityData 1282
 PDDocGetNewSecurityInfo 1283
 PDDocGetNumOCGs 1284
 PDDocGetNumPages 1285
 PDDocGetNumThreads 1286
 PDDocGetOCConfig 1287
 PDDocGetOCContext 1288
 PDDocGetOCGs 1289
 PDDocGetOpenAction 1290
 PDDocGetPageLabel 1291
 PDDocGetPageMode 1292
 PDDocGetPageObjByNum 1293
 PDDocGetPermissions 1294
 PDDocGetSecurityData 1295
 PDDocGetStructTreeRoot 1296
 PDDocGetThread 1297
 PDDocGetThreadIndex 1298
 PDDocGetTrapped 1299, 1356



PDDocGetVersion 1300	PDDrawCosObjToWindow 1102
PDDocGetWordFinder 1301, 1305	PDDrawCosObjWithParams 1104
PDDocGetXAPMetadata 1302	PDEAcquire 1902
PDDocGetXAPMetadataProperty 1303	PDEAddTag 1903
PDDocImportCosDocNotes 1306	PDEAttrEnumTable 1739
PDDocImportNotes 1308	PDEBeginContainerCreate 1745
PDDocInsertPages 1309	PDEBeginContainerGetDict 1746
PDDocMovePage 1312	PDEBeginContainerGetMCTag 1747
PDDocNewSecurityData 1313	PDEBeginContainerSetDict 1748
PDDocOpen 1314	PDEBeginContainerSetMCTag 1749
PDDocOpenEx 1316	PDEBeginGroupCreate 1750
PDDocOpenFromASFile 1318	PDEClipAddElem 1751
PDDocOpenFromASFileEx 1320	PDEClipCopy 1752
PDDocOpenWithParams 1322	PDEClipCreate 1753
PDDocPermRequest 1323	PDEClipFlattenedEnumElems 1754
PDDocReadAhead 1325	PDEClipGetElem 1755
PDDocReadAheadEmbeddedFile 1326	PDEClipGetNumElems 1756
PDDocReadAheadPages 1327	PDEClipRemoveElems 1757
PDDocRelease 1328	PDEColorSpaceCreate 1758
PDDocRemoveNameTree 1329	PDEColorSpaceCreateFromCosObj 1759
PDDocRemoveOpenAction 1330	PDEColorSpaceCreateFromName 1760
PDDocRemovePageLabel 1331	PDEColorSpaceGetBase 1761
PDDocRemoveStructTreeRoot 1332	PDEColorSpaceGetBaseNumComps 1762
PDDocRemoveThread 1333	PDEColorSpaceGetCosObj 1763
PDDocReplaceOCG 1334	PDEColorSpaceGetCTable 1764
PDDocReplacePages 1335	PDEColorSpaceGetHiVal 1765
PDDocRequestEntireFile 1337	PDEColorSpaceGetName 1766
PDDocRequestPages 1338	PDEColorSpaceGetNumComps 1767
PDDocSave 1339	PDEContainerCreate 1768
PDDocSaveWithParams 1341	PDEContainerGetContent 1769
PDDocSetFlags 1342	PDEContainerGetDict 1770
PDDocSetFullScreen 1343	PDEContainerGetMCTag 1771
PDDocSetInfo 1344	PDEContainerGetXAPMetadata 1772
PDDocSetNewCryptFilterData 1345	PDEContainerSetContent 1773
PDDocSetNewCryptFilterMethod 1346	PDEContainerSetDict 1774
PDDocSetNewCryptHandler 1347	PDEContainerSetMCTag 1775
PDDocSetNewCryptHandlerEx 1349	PDEContainerSetXAPMetadata 1776
PDDocSetNewDefaultFilters 1351	PDEContentAddElem 1777
PDDocSetNewSecurityData 1352	PDEContentCreate 1779
PDDocSetOpenAction 1353	PDEContentCreateFromCosObj 1780
PDDocSetPageLabel 1354	PDEContentFlattenOC 1781
PDDocSetPageMode 1355	PDEContentGetAttrs 1782
PDDocSetXAPMetadata 1357	PDEContentGetDefaultColorSpace 1783
PDDocSetXAPMetadataProperty 1359	PDEContentGetElem 1784



PDEContentGetNumElems 1785
 PDEContentGetResources 1786
 PDEContentRemoveElem 1787
 PDEContentToCosObj 1788
 PDEDefaultGState 1740
 PDEDeviceNColorsCreate 1790
 PDEDeviceNColorsGetColorValue 1791
 PDEElementCopy 1792
 PDEElementGetAllVisibilities 1793
 PDEElementGetBBox 1794
 PDEElementGetClip 1795
 PDEElementGetGState 1796
 PDEElementGetMatrix 1798
 PDEElementGetOCMD 1799
 PDEElementHasGState 1800
 PDEElementIsAtPoint 1801
 PDEElementIsAtRect 1802
 PDEElementIsCurrentlyVisible 1803
 PDEElementMakeVisible 1804
 PDEElementRemoveOCMD 1805
 PDEElementSetClip 1806
 PDEElementSetGState 1807
 PDEElementSetMatrix 1808
 PDEElementSetOCMD 1809
 PDEEndContainerCreate 1810
 PDEEndGroupCreate 1811
 PDEEnumElements 1741
 PDEExtGStateAcquireSoftMask 1812
 PDEExtGStateCreate 1813
 PDEExtGStateCreateNew 1814
 PDEExtGStateGetAIS 1815
 PDEExtGStateGetBlendMode 1816
 PDEExtGStateGetCosObj 1817
 PDEExtGStateGetOpacityFill 1818
 PDEExtGStateGetOpacityStroke 1819
 PDEExtGStateGetOPFill 1820
 PDEExtGStateGetOPM 1821
 PDEExtGStateGetOPStroke 1822
 PDEExtGStateGetSA 1823
 PDEExtGStateGetTK 1824
 PDEExtGStateHasSoftMask 1825
 PDEExtGStateSetAIS 1826
 PDEExtGStateSetBlendMode 1827
 PDEExtGStateSetOpacityFill 1828
 PDEExtGStateSetOpacityStroke 1829
 PDEExtGStateSetOPFill 1830
 PDEExtGStateSetOPM 1831
 PDEExtGStateSetOPStroke 1832
 PDEExtGStateSetSA 1833
 PDEExtGStateSetSoftMask 1834
 PDEExtGStateSetTK 1835
 PDEFontCreate 1836
 PDEFontCreateFromCosObj 1838
 PDEFontCreateFromSysFont 1839
 PDEFontCreateFromSysFontAndEncoding 1841
 PDEFontCreateFromSysFontEx 1842
 PDEFontCreateFromSysFontWithParams 1844
 PDEFontCreateToUnicodeNow 1845
 PDEFontCreateWidthsNow 1846
 PDEFontCreateWithParams 1847
 PDEFontEmbedNow 1849
 PDEFontEmbedNowDontSubset 1850
 PDEFontGetAttrs 1851
 PDEFontGetCosObj 1852
 PDEFontGetCreateNeedFlags 1853
 PDEFontGetNumCodeBytes 1854
 PDEFontGetOneByteEncoding 1855
 PDEFontGetSysEncoding 1856
 PDEFontGetSysFont 1857
 PDEFontGetWidths 1858
 PDEFontGetWidthsNow 1859
 PDEFontIsEmbedded 1860
 PDEFontIsMultiByte 1861
 PDEFontSetSysEncoding 1862
 PDEFontSetSysFont 1863
 PDEFontSubsetNow 1864
 PDEFontSumWidths 1866
 PDEFontTranslateGlyphIdsToUnicode 1867
 PDEFormAcquireXGroup 1868
 PDEFormCreateClone 1869
 PDEFormCreateFromCosObj 1870
 PDEFormGetContent 1871, 1874
 PDEFormGetCosObj 1872
 PDEFormHasXGroup 1873
 PDEFormSetXGroup 1875
 PDEGetTag 1905
 PDEGroupCreate 1876
 PDEGroupGetContent 1877



PDEGroupSetContent 1878
 PDEImageCreate 1879
 PDEImageCreateFromCosObj 1881
 PDEImageDataIsEncoded 1882
 PDEImageGetAttrs 1883
 PDEImageGetColorSpace 1884
 PDEImageGetCosObj 1885
 PDEImageGetData 1886
 PDEImageGetDataLen 1888
 PDEImageGetDataStm 1889
 PDEImageGetDecodeArray 1890
 PDEImageGetFilterArray 1891
 PDEImageGetMatteArray 1892
 PDEImageGetSMask 1893
 PDEImageHasSMask 1894
 PDEImageIsCosObj 1895
 PDEImageSetColorSpace 1896
 PDEImageSetData 1897
 PDEImageSetDataStm 1898
 PDEImageSetDecodeArray 1899
 PDEImageSetMatteArray 1900
 PDEImageSetSMask 1901
 PDELogDump 1737
 PDEEmbedSysFontForPDFont 2018
 PDEMergeResourcesDict 1743
 PDEnumDocs 1105
 PDEnumSysFonts 2020
 PDEObjectDump 1738
 PDEObjectGetType 1906
 PDEPathAddSegment 1909
 PDEPathCreate 1911
 PDEPathGetData 1912
 PDEPathGetPaintOp 1913
 PDEPathSetData 1914
 PDEPathSetPaintOp 1915
 PDEPatternCreate 1916
 PDEPatternGetCosObj 1917
 PDEPlaceCreate 1918
 PDEPlaceGetDict 1919
 PDEPlaceGetMCTag 1920
 PDEPlaceSetDict 1921
 PDEPlaceSetMCTag 1922
 PDEPSCreate 1923
 PDEPSCreateFromCosObj 1924

PDEPSGetAttrs 1925
 PDEPSGetData 1926
 PDEPSGetDataStm 1927
 PDEPSSetData 1928
 PDEPSSetDataStm 1929
 PDEPurgeCache 1744
 PDERelease 1907
 PDERemoveTag 1908
 PDEShadingCreateFromCosObj 1941
 PDEShadingGetCosObj 1942
 PDESoftMaskAcquireForm 1930
 PDESoftMaskCreate 1931
 PDESoftMaskCreateFromCosObj 1932
 PDESoftMaskCreateFromName 1933
 PDESoftMaskGetBackdropColor 1934
 PDESoftMaskGetCosObj 1935
 PDESoftMaskGetName 1936
 PDESoftMaskGetTransferFunction 1937
 PDESoftMaskSetBackdropColor 1938
 PDESoftMaskSetTransferFunction 1939
 PDESoftMaskSetXGroup 1940
 PDETextAdd 1943
 PDETextAddItem 1945
 PDETextCreate 1946
 PDETextGetAdvance 1947
 PDETextGetAdvanceWidth 1949, 1972
 PDETextGetBBox 1951
 PDETextGetFont 1952
 PDETextGetGState 1953
 PDETextGetItem 1954
 PDETextGetMatrix 1955
 PDETextGetNumBytes 1956
 PDETextGetNumChars 1957
 PDETextGetNumRuns 1958
 PDETextGetQuad 1959
 PDETextGetRunForChar 1961
 PDETextGetState 1962
 PDETextGetStrokeMatrix 1963
 PDETextGetText 1964
 PDETextGetTextMatrix 1965
 PDETextGetTextState 1966
 PDETextIsAtPoint 1968
 PDETextIsAtRect 1969
 PDETextItemCopyText 1985



PDTextItemCreate 1986
 PDTextItemGetFont 1988
 PDTextItemGetGState 1989
 PDTextItemGetTextLen 1990
 PDTextItemGetTextMatrix 1991
 PDTextItemGetTextState 1992
 PDTextItemRemoveChars 1993
 PDTextItemReplaceChars 1994
 PDTextItemReplaceText 1995
 PDTextItemSetFont 1996
 PDTextItemSetGState 1997
 PDTextItemSetTextMatrix 1998
 PDTextItemSetTextState 1999
 PDTextRemove 1970
 PDTextReplaceChars 1973
 PDTextRunGetCharOffset 1975
 PDTextRunGetNumChars 1976
 PDTextRunSetFont 1977
 PDTextRunSetGState 1978
 PDTextRunSetMatrix 1979
 PDTextRunSetState 1980
 PDTextRunSetStrokeMatrix 1981
 PDTextRunSetTextMatrix 1982
 PDTextRunSetTextState 1983
 PDTextSplitRunAt 1984
 PDUnknownGetOpName 2000
 PDExGroupAcquireColorSpace 2001
 PDExGroupCreate 2002
 PDExGroupCreateFromCosObj 2003
 PDExGroupGetCosObj 2004
 PDExGroupGetIsolated 2005
 PDExGroupGetKnockout 2006
 PDExGroupSetColorSpace 2007
 PDExGroupSetIsolated 2008
 PDExGroupSetKnockout 2009
 PDXObjectCreate 2010
 PDXObjectGetCosObj 2011
 PDFileSpecAcquireASPath 1361
 PDFileSpecFromCosObj 1362
 PDFileSpecGetCosObj 1363
 PDFileSpecGetDIPath 1364
 PDFileSpecGetDoc 1365
 PDFileSpecGetFileSys 1366
 PDFileSpecGetFileSysName 1367
 PDFileSpecIsValid 1368
 PDFileSpecNewFromASPath 1369
 PDFindSysFont 2022
 PDFindSysFontEx 2023
 PDFindSysFontForPDEFont 2025
 PDFontAcquireEncodingArray 1370
 PDFontAcquireXlateTable 1371
 PDFontEncodingArrayRelease 1372
 PDFontEnumCharProcs 1373
 PDFontFromCosObj 1375
 PDFontGetASTextName 1376
 PDFontGetBBox 1377
 PDFontGetCharSet 1378
 PDFontGetCIDSystemInfo 1379
 PDFontGetCIDSystemSupplement 1381
 PDFontGetCosObj 1383
 PDFontGetDescendant 1384
 PDFontGetEncodingIndex 1385
 PDFontGetEncodingName 1386
 PDFontGetFontMatrix 1387
 PDFontGetMetrics 1388
 PDFontGetName 1390
 PDFontGetSubtype 1392
 PDFontGetWidths 1393
 PDFontIsEmbedded 1395
 PDFontSetMetrics 1396
 PDFontXlateString 1397
 PDFontXlateTableRelease 1398
 PDFontXlateToHost 1399
 PDFontXlateToUCS 1401
 PDFontXlateWidths 1403
 PDFormEnumPaintProc 1404
 PDFormEnumPaintProcWithParams 1405
 PDFormEnumResources 1406
 PDFormGetBBox 1407
 PDFormGetFormType 1408
 PDFormGetMatrix 1409
 PDFormGetXUIDCosObj 1410
 PDGetAnnotHandlerByName 1175
 PDGetHostEncoding 1106
 PDGetPDFDocEncoding 1107
 PDGraphicGetBBox 1411
 PDGraphicGetCurrentMatrix 1412
 PDGraphicGetState 1413



PDHostMBLen 1108
 PDImageColorSpaceGetIndexLookup 1414
 PDImageGetAttrs 1415
 PDImageSelAdjustMatrix 1418
 PDImageSelectAlternate 1416
 PDImageSelGetDeviceAttr 1419
 PDImageSelGetGeoAttr 1421
 PDInlinelImageColorSpaceGetIndexLookup 1423
 PDInlinelImageGetAttrs 1424
 PDInlinelImageGetData 1426
 PDLINKAnnotGetAction 1428
 PDLINKAnnotGetBorder 1429
 PDLINKAnnotRemoveAction 1430
 PDLINKAnnotSetAction 1431
 PDLINKAnnotSetBorder 1433
 PDNameTreeEnum 1435
 PDNameTreeEqual 1436
 PDNameTreeFromCosObj 1437
 PDNameTreeGet 1438
 PDNameTreeGetCosObj 1439
 PDNameTreeIsValid 1440
 PDNameTreeLookup 1441
 PDNameTreeNew 1443
 PDNameTreeNotifyNameAdded 1444
 PDNameTreeNotifyNameRemoved 1445
 PDNameTreePut 1446
 PDNameTreeRemove 1447
 PDNumTreeEnum 1448
 PDNumTreeEqual 1449
 PDNumTreeFromCosObj 1450
 PDNumTreeGet 1451
 PDNumTreeGetCosObj 1452
 PDNumTreeIsValid 1453
 PDNumTreeNew 1454
 PDNumTreePut 1455
 PDNumTreeRemove 1456
 PDOCConfigCreate 1457
 PDOCConfigDestroy 1458
 PDOCConfigGetAllRadioButtonGroups 1459
 PDOCConfigGetCosObj 1460
 PDOCConfigGetCreator 1461
 PDOCConfigGetInitState 1462, 1470
 PDOCConfigGetIntent 1463
 PDOCConfigGetName 1464
 PDOCConfigGetOCGOrder 1465
 PDOCConfigGetPDDoc 1466
 PDOCConfigGetRadioButtonGroupForOCG 1467
 PDOCConfigMakeRadioButtonGroup 1468
 PDOCConfigSetCreator 1469
 PDOCConfigSetIntent 1471
 PDOCConfigSetName 1472
 PDOCConfigSetOCGOrder 1473
 PDOCContextApplyAutoStateChanges 1474
 PDOCContextClearAllUserOverrides 1475
 PDOCContextContentIsVisible 1476
 PDOCContextFindAutoStateChanges 1477
 PDOCContextFree 1478
 PDOCContextGetIntent 1479
 PDOCContextGetNonOCDrawing 1480
 PDOCContextGetOCDrawEnumType 1481
 PDOCContextGetOCGStates 1482
 PDOCContextGetPDDoc 1483
 PDOCContextInit 1484
 PDOCContextMakeCopy 1485
 PDOCContextMakeCopyWithAutoStateChanges 1486
 PDOCContextNew 1487
 PDOCContextNewWithInitialState 1488
 PDOCContextNewWithOCDisabled 1489
 PDOCContextPopOCMD 1490
 PDOCContextPushOCMD 1491
 PDOCContextResetOCMDStack 1492
 PDOCContextSetIntent 1494
 PDOCContextSetNonOCDrawing 1495
 PDOCContextSetOCDrawEnumType 1496
 PDOCContextSetOCGStates 1497
 PDOCContextXObjectIsVisible 1498
 PDOCGCreate 1499
 PDOCGCreateFromCosObj 1500
 PDOCGDestroy 1501
 PDOCGGetCosObj 1502
 PDOCGGetCurrentState 1503, 1513
 PDOCGGetFromCosObj 1504
 PDOCGGetInitialState 1505
 PDOCGGetIntent 1506
 PDOCGGetName 1507
 PDOCGGetPDDoc 1508
 PDOCGGetUsageEntry 1509
 PDOCGGetUserOverride 1510



PDOCGHasUsageInfo 1511
 PDOCGRemoveInitialState 1512
 PDOCGSetInitialState 1514
 PDOCGSetIntent 1515
 PDOCGSetName 1516
 PDOCGSetUsageDictEntry 1517
 PDOCGSetUserOverride 1519
 PDOCGUsedInOCConfig 1520
 PDOCGUsedInOCContext 1521
 PDOCMDCreate 1522
 PDOCMDFindOrCreate 1523
 PDOCMDGetCosObj 1524
 PDOCMDGetFromCosObj 1525
 PDOCMDGetOCGs 1526
 PDOCMDGetPDDoc 1527
 PDOCMDGetVisPolicy 1528
 PDOCMDIsCurrentlyVisible 1529
 PDOCMDsAreCurrentlyVisible 1530
 PDOCMDsMakeContentVisible 1531
 PDPageAcquirePDEContent 1532
 PDPageAddAnnot 1534
 PDPageAddCosContents 1535
 PDPageAddCosResource 1536
 PDPageAddNewAnnot 1538
 PDPageCreateAnnot 1540
 PDPageDrawContentsPlaceWithParams 1542
 PDPageDrawContentsToWindow 1543
 PDPageDrawContentsToWindowEx 1546
 PDPageDrawContentsWithParams 1548
 PDPageEnumContents 1549
 PDPageEnumInks 1550
 PDPageEnumOCGs 1552
 PDPageEnumResources 1553
 PDPageFlattenOC 1554
 PDPageGetAnnot 1555
 PDPageGetAnnotIndex 1556
 PDPageGetAnnotSequence 1557
 PDPageGetBBox 1558
 PDPageGetBox 1559
 PDPageGetCosObj 1560
 PDPageGetCosResources 1561
 PDPageGetCropBox 1562
 PDPageGetDefaultMatrix 1563
 PDPageGetDoc 1564
 PDPageGetDuration 1565
 PDPageGetFlippedMatrix 1566
 PDPageGetMediaBox 1567
 PDPageGetNumAnnots 1568
 PDPageGetNumber 1569
 PDPageGetOCGs 1570
 PDPageGetPalette 1571
 PDPageGetPDEContentFilters 1572
 PDPageGetPDEContentFlags 1573
 PDPageGetRotate 1574
 PDPageGetTransition 1575
 PDPageGetVisibleBbox 1576
 PDPageHasTransition 1577
 PDPageHasTransparency 1578
 PDPageLabelEqual 1611
 PDPageLabelFromCosObj 1612
 PDPageLabelGetCosObj 1613
 PDPageLabelGetPrefix 1614
 PDPageLabelGetStart 1615
 PDPageLabelGetStyle 1616
 PDPageLabelValid 1617
 PDPageLabelNew 1618
 PDPageMakeSeparations 1579
 PDPageNotifyContentsDidChange 1580
 PDPageNotifyContentsDidChangeEx 1581
 PDPageNumFromCosObj 1582
 PDPagePDEContentWasChanged 1583
 PDPageRegisterForPDEContentChanged 1584
 PDPageRegisterForPDEContentNotCached 1585
 PDPageRelease 1587
 PDPageReleasePDEContent 1588
 PDPageRemoveAnnot 1589
 PDPageRemoveCosContents 1591
 PDPageRemoveCosResource 1592
 PDPageResumePDEContentChanged 1593
 PDPageSetBox 1594
 PDPageSetCropBox 1595
 PDPageSetDuration 1596
 PDPageSetMediaBox 1597
 PDPageSetPDEContent 1598, 1599
 PDPageSetPDEContentFilters 1600
 PDPageSetPDEContentFlags 1601
 PDPageSetRotate 1602
 PDPageSetTransition 1603

PDPageStmGetInlinelImage 1604
 PDPageStmGetToken 1606
 PDPageSuspendPDEContentChanged 1608
 PDPageUnRegisterForPDEContentChanged 1609
 PDPageUnRegisterForPDEContentNotCached 1610
 PDPathEnum 1620
 PDPathGetPaintOp 1621
 PDPrefGetColorCal 1110
 PDPrefSetColorCal 1111
 PDRegisterAnnotHandler 1143, 1176
 PDRegisterCryptHandler 1112
 PDRegisterCryptHandlerEx 1114
 PDRegisterFileSpecHandler 1116
 PDRegisterFileSpecHandlerByName 1117
 PDSAttrObjCreate 2040
 PDSAttrObjCreateFromStream 2041
 PDSAttrObjGetCosObj 2042
 PDSAttrObjGetOwner 2043
 PDSClassMapAddAttrObj 2044
 PDSClassMapGetAttrObj 2045
 PDSClassMapGetNumAttrObjs 2046
 PDSClassMapRemoveAttrObj 2047
 PDSClassMapRemoveClass 2048
 PDSElementAddAttrObj 2049
 PDSElementAddClass 2050
 PDSElementClearID 2051
 PDSElementCreate 2052
 PDSElementGetActualText 2053
 PDSElementGetAlt 2054
 PDSElementGetAttrObj 2056
 PDSElementGetClass 2057
 PDSElementGetCosObj 2058
 PDSElementGetFirstPage 2059
 PDSElementGetID 2061
 PDSElementGetKid 2062
 PDSElementGetKidEx 2064, 2066
 PDSElementGetLanguage 2067
 PDSElementGetNumAttrObjs 2068
 PDSElementGetNumClasses 2069
 PDSElementGetNumKids 2070
 PDSElementGetParent 2071
 PDSElementGetRevision 2072
 PDSElementGetStructTreeRoot 2073
 PDSElementGetTitle 2074
 PDSElementGetType 2075
 PDSElementHasActualText 2076
 PDSElementHasAlt 2077
 PDSElementHasLanguage 2078
 PDSElementIncrementRevision 2079
 PDSElementInsertKid 2080
 PDSElementInsertMCAsKid 2081
 PDSElementInsertOBJAsKid 2082
 PDSElementInsertStmMCAsKid 2083
 PDSElementRemoveAllAttrObjs 2085
 PDSElementRemoveAllClasses 2086
 PDSElementRemoveAttrObj 2087
 PDSElementRemoveClass 2088
 PDSElementRemoveKid 2089
 PDSElementRemoveKidMC 2090
 PDSElementRemoveKidOBJ 2091
 PDSElementReplaceKid 2092
 PDSElementReplaceKidMC 2093
 PDSElementReplaceKidOBJ 2094
 PDSElementSetActualText 2095
 PDSElementSetAlt 2096
 PDSElementSetID 2097
 PDSElementSetLanguage 2098
 PDSElementSetTitle 2100
 PDSElementSetType 2101
 PDSMCGetInfo 2102
 PDSMCGetParent 2103
 PDSMCGetPDEContainer 2104
 PDSMCIDGetParent 2105
 PDSOBJGetParent 2106
 PDSRoleMapCopy 2107
 PDSRoleMapDoesMap 2108
 PDSRoleMapGetDirectMap 2109
 PDSRoleMapMap 2110
 PDSRoleMapUnMapDst 2111
 PDSRoleMapUnMapSrc 2112
 PDSTreeRootCreateClassMap 2113
 PDSTreeRootCreateRoleMap 2114
 PDSTreeRootGetClassMap 2115
 PDSTreeRootGetElementFromID 2116
 PDSTreeRootGetKid 2117
 PDSTreeRootGetNumKids 2118
 PDSTreeRootGetRoleMap 2119
 PDSTreeRootInsertKid 2120

PDTreeRootRemoveClassMap 2121
 PDTreeRootRemoveKid 2122
 PDTreeRootRemoveRoleMap 2123
 PDTreeRootReplaceKid 2124
 PDTreeRootReplaceStreamRef 2125
 PDStyleGetColor 1622
 PDStyleGetFont 1623
 PDStyleGetFontSize 1624
 PDSysEncodingCreateFromBaseName 2012
 PDSysEncodingCreateFromCMapName 2013
 PDSysEncodingCreateFromCodePage 2014
 PDSysEncodingGetWMode 2015
 PDSysEncodingIsIdentity 2016
 PDSysEncodingIsMultiByte 2017
 PDSysFontAcquirePlatformData 2026
 PDSysFontGetAttrs 2027
 PDSysFontGetCIDSystemInfo 2029
 PDSysFontGetCreateFlags 2030
 PDSysFontGetEncoding 2031
 PDSysFontGetInfo 2032
 PDSysFontGetName 2033
 PDSysFontGetType0Widths 2034
 PDSysFontGetWidths 2036
 PDSysFontGetWidthsEx 2037
 PDSysFontReleasePlatformData 2038
 PDTextAnnotGetContents 1627
 PDTextAnnotIsOpen 1628
 PDTextAnnotSetContents 1629
 PDTextAnnotSetOpen 1630
 PDTextEnum 1625
 PDTextGetState 1626
 PDTextSelectCreatePageHilite 1631
 PDTextSelectCreatePageHiliteEx 1633
 PDTextSelectCreateRanges 1635
 PDTextSelectCreateRangesEx 1636
 PDTextSelectCreateWordHilite 1638
 PDTextSelectCreateWordHiliteEx 1640
 PDTextSelectDestroy 1642
 PDTextSelectEnumQuads 1643
 PDTextSelectEnumText 1644
 PDTextSelectEnumTextUCS 1645
 PDTextSelectGetBoundingRect 1646
 PDTextSelectGetPage 1647
 PDTextSelectGetRange 1648
 PDTextSelectGetRangeCount 1649
 PDThreadDestroy 1650
 PDThreadFromCosObj 1651
 PDThreadGetCosObj 1652
 PDThreadGetFirstBead 1653
 PDThreadGetInfo 1654
 PDThreadIsValid 1656
 PDThreadNew 1657
 PDThreadSetFirstBead 1658
 PDThreadSetInfo 1659
 PDTransEqual 1661
 PDTransFromCosObj 1662
 PDTransGetCosObj 1663
 PDTransGetDuration 1664
 PDTransGetSubtype 1665
 PDTransIsValid 1666
 PDTransNew 1667
 PDTransNewFromCosDoc 1668
 PDTransNull 1669
 PDViewDestCreate 1670
 PDViewDestDestroy 1672
 PDViewDestFromCosObj 1673
 PDViewDestGetAttr 1674
 PDViewDestGetCosObj 1676
 PDViewDestIsValid 1677
 PDViewDestResolve 1678
 PDWordCreateTextSelect 1679
 PDWordFilterString 1680
 PDWordFilterWord 1682
 PDWordFinderAcquireVisibleWordList 1712
 PDWordFinderAcquireWordList 1716
 PDWordFinderDestroy 1720
 PDWordFinderEnumWords 1721, 1723
 PDWordFinderEnumWordsStr 1725
 PDWordFinderGetLatestAlgVersion 1727
 PDWordFinderGetNthWord 1728
 PDWordFinderReleaseWordList 1729
 PDWordGetASText 1684
 PDWordGetAttr 1685, 1686
 PDWordGetByteIdxFromHiliteChar 1688
 PDWordGetCharacterTypes 1689
 PDWordGetCharDelta 1691
 PDWordGetCharEncFlags 1692
 PDWordGetCharOffset 1693

PDWordGetCharOffsetEx 1695
 PDWordGetCharQuad 1697
 PDWordGetLength 1698
 PDWordGetNthCharStyle 1699
 PDWordGetNthQuad 1700
 PDWordGetNumHiliteChar 1702
 PDWordGetNumQuads 1703
 PDWordGetString 1704
 PDWordGetStyleTransition 1706
 PDWordIMakeVisible 1709
 PDWordIsCurrentlyVisible 1707
 PDWordIsRotated 1708
 PDWordSplitString 1710
 PDXlateToHost 1119
 PDXlateToHostEx 1121
 PDXlateToPDFDocEnc 1123
 PDXlateToPDFDocEncEx 1125
 PDXObjectEnumFilters 1730
 PDXObjectGetCosObj 1731
 PDXObjectGetData 1732
 PDXObjectGetDataLength 1733
 PDXObjectGetSubtype 1734

R

RectToAVRect 2133

U

UnixAppAddModifierCallback 2134
 UnixAppClipboardGetItemId 2135
 UnixAppDispatchEvent 2136
 UnixAppGetAppShellWidget 2137
 UnixAppGetPlugInFilename 2139
 UnixAppLoadPlugInAppDefaults 2140
 UnixAppProcessEvent 2142
 UnixAppRemoveModifierCallback 2143
 UnixAppWaitForWm 2144
 UnixMachinePortGetOffscreenPixmap 2145
 UnixSysGetConfigName 2146
 UnixSysGetCursor 2147
 UnixSysGetCwd 2149
 UnixSysGetHomeDirectory 2150
 UnixSysGetHostname 2151

UnixSysGetIcon 2152
 UnixSysGetInstallDirectory 2154
 UnixSysGetPixmap 2155
 UnixSysGetString 2157
 UnixSysGetTempFileDirectory 2158
 UnixSysPrefInit 2159
 UnixSysPrefUpdate 2163

W

WinAppEnableIdleTimer 2165
 WinAppGetModalParent 2166
 WinAppGetPalette 2167
 WinAppGetPrinterHDC 2168
 WinAppRegisterInterface 2169
 WinAppRegisterModelessDialog 2170
 WinAppUnRegisterModelessDialog 2171