Adobe PDF Library Overview

Adobe® PDF Library® SDK

December 2016

Version 15.0.4

Contents

Preface **5**
What's in this guide? 5
Who should read this guide? 5
Related documentation 5
1 Introduction **6**
2 About the PDF Library SDK **8**
Supported environments 8
Contents 8
SnippetRunner 8
Installing the PDF Library SDK 14
Updating to PDF Library SDK XV 15

Preface

This document provides an overview of the Adobe PDF Library SDK.

What's in this guide?

This guide provides an introduction to development using the Adobe PDF Library. It describes the Adobe PDF Library, the Adobe PDF Library Software Development Kit (SDK), support and licensing options, and basic development tasks.

Who should read this guide?

This guide is intended for programmers who need basic information on the Adobe PDF Library and its capabilities. It assumes familiarity with C programming, common development tasks, and the use of methods exported by an object code library.

Related documentation

The following resources provide further information about the PDF Library SDK, as well as additional documents that you should have available for reference.

For information about	See
A detailed description of the PDF file format.	*PDF Reference*
Developing plug-ins for Acrobat and Adobe Reader, as well as for PDF Library applications.	*Developing Plug-ins and Applications*
Detailed descriptions of the APIs for Acrobat and Adobe Reader plug-ins, as well as for PDF Library applications.	*Acrobat and PDF Library API Reference*
An overview of the SnippetRunner tool and the plug-in snippets provided with the Acrobat SDK.	*SnippetRunner Cookbook*
Licensing and other information about the PDF Library SDK.	www.adobe.com/go/acrobat_developer
Information about Adobe enterprise developer resources.	www.adobe.com/enterprise/developer

1 Adobe PDF Library Overview

Introduction

Designed specifically for OEMs, ISVs, system integrators, and enterprise IT developers, the Adobe PDF Library SDK contains a powerful set of functions for developing third-party solutions and processes around Adobe PDF. The PDF Library is based on the core technology of the Adobe Acrobat line of products and offers complete functionality for generating, manipulating, rendering, and printing PDF documents.

The library enables PDF functionality to be seamlessly embedded within applications. It also provides reliable, accurate and Adobe-supported implementation of the latest PDF specification.

The PDF Library and the PDF viewers

The PDF Library shares much of the same source code as the Adobe PDF viewing applications (Acrobat Professional, Acrobat Standard and Adobe Reader). There is a large degree of overlap between the functionality provided by the PDF Library Software Development Kit (SDK) and that of the Acrobat SDK. They differ in providing access to the Acrobat user interface:

- The Acrobat SDK is meant for the plug-in environment, and allows you to control and interact with the Acrobat user interface.
- The PDF Library SDK is intended for interaction between PDF content and other applications, such as high volume batch processing and PDF generation applications. It does not export methods for creating or managing Acrobat UI elements—that is, the AcroView (AV) layer of the core API.

If you are interested in the documentation and samples in the Acrobat Software Development Kit (SDK), see the Acrobat Developer Center.

New in version 15.0.4

- Now, tags can be preserved in PDF during PDF/A conversion.

The following table specifies the supported platforms, operating systems, and compilers for PDFL SDK XV:

Platform	Processor	Compiler	Additional OS
Windows-7 32-bit and 64-bit	Intel	Visual Studio 2013 Update 3	Win 8.1and Win 10
Linux 32-bit and 64-bit (RHEL 7)	Intel	gcc version 4.8.2, glibc version 2.17	
Mac 64-bit (Mac OS X 10.11)	Intel	XCode 7	MAC OS X 10.9

New Guide for PDF Extensions

PDF Library SDK XV includes the guide *Adobe Supplement to ISO32000*, which was previously named *Acrobat Implementation of the PDF Standard*. The guide can be found within the portfolio called PDF

Reference and Related Documentation, which is located at the Adobe PDF Technology Center. *Adobe Supplement to ISO32000* describes PDF extensions supported in PDF Library SDK XV, the Adobe Acrobat DC family of products, and related Adobe products. This guide also contains implementation notes for Acrobat.

The PDF specification is currently represented by the following documents:

- *PDF Reference, sixth edition, version 1.7*
- *PDF Redaction: Addendum to the PDF Reference, sixth edition, version 1.7*
- *Errata for the PDF Reference, sixth edition, version 1.7*

This guide reflects Adobe's commitment to transferring responsibility of the PDF specification to the International Standards Organization (ISO).

Packaging of the PDF Specification

PDF Library SDK XV packages the *PDF Reference, sixth edition, version 1.7* in a PDF portfolio that also includes the following documents:

- *PDF Redaction: Addendum to the PDF Reference, sixth edition, version 1.7*
- *Errata for the PDF Reference, sixth edition, version 1.7*
- *Adobe Supplement to ISO32000*

A PDF portfolio is a single PDF file that enables a viewer application to describe and navigate between the files it contains. The file name of this PDF portfolio is pdf_reference.pdf.

2 About the PDF Library SDK

This chapter helps you get started with development using the PDF Library SDK. It describes the contents of each directory in the SDK installation, lists available code samples, and provides platform-specific information on how to set up the development environment.

Supported environments

PDF Library SDK XV and its project files are supported for the platforms, operating systems, and compilers listed in *Developing Plug-ins and Applications*. While it may be possible to use the library in other development environments, such use is not supported by Adobe Developer Support.

Contents

The PDF Library SDK XV consists of the following components:

- Core libraries that provide the functionality
- Header files that provide access to the libraries
- Fonts used in the library's basic operations
- Sample applications and code snippets showing how to use the library for a variety of purposes
- Documentation of installation and development techniques and of the PDF Library APIs

The components that are shipped with the PDF Library XV SDK are listed in *Developing Plug-ins and Applications*. These include the PDF Library, helper libraries, headers, and samples.

SnippetRunner

The SnippetRunner is based on the SnippetRunner provided with the Acrobat 8 SDK, but is adapted to the PDF Library environment. It has a command-line interface and prints output to the console. It is found in the SnippetRunner subdirectory of the Samples directory. For more information, see the *Snippet Runner Cookbook*.

The following table shows the commands that you can use to control the snippet runner in a command shell. The snippets (see `PDF Library snippets`) are arranged in a directory-like hierarchy. You can use the `findsnip` command to locate a snippet and the `cd` command to switch between directories.

Note: The command names are not case-sensitive. However, the snippet names and directory names are case-sensitive.

Snippet Runner commands

Command	Description
`cd`	Navigate the hierarchy of snippets using standard file path semantics.
`findsnip`	Search for a snippet based on a specified substring of the snippet identification string. If the result is a single file, prompts for execution.

Command	Description
`snipinfo`	Display the description of a specified snippet.
`docname`	Display the name and file information for the current document.
`setpage`	Specify the current page of the document.
`opendoc`	Open the specified document. This document becomes the new current document. If the previously open document was modified, the command prompts to save changes.
`closedoc`	Close the current document. If it was modified, the command prompts the user to save changes.
`savedoc`	Save the document to the specified file.
`revertdoc`	Revert the current document to its last saved condition.
`run`	Run the specified snippet. The snippet must exist in the current directory—you cannot specify a path.
`ls`	List the snippets (and hierarchy nodes) in the current directory.
`pwd`	Print the current directory.
`help`	List all the commands with their synopses, or provide help for a specified command.
`quit`	Quit the application.
`whichPage`	Display the current selected page.

The snippets provided include most of those available for the Acrobat SDK. Snippets include the following:

PDF Library snippets

Snippet	Description
`ACEEnumProfilesSnip`	Enumerates the profiles of a given type installed on the system. Makes a profile list, gets a count of how many profiles are in the list, gets the description for each item in the list, and dumps the information in the DebugWindow.
`ACEEnumSettingsSnip`	Demonstrates how to programmatically access color setup files using the Adobe Color Engine (ACE).
`ACEGetWorkingSpaceSnip`	Gets working space profile (RGB, grayscale, and CMYK) information such as size, description and color spaces, and lists the information in the DebugWindow.
`ACETransPDETextColorSnip`	Takes an input profile (DeviceCMYK) and an output profile (sRGB), creates a color transform from them, and then applies the transform to a single color of a PDF text object.
`AddImageMetadataSnip`	Shows how to add XMP metadata to an image XObject in a PDF document.

Snippet	Description
AddImageSnip	Creates an image XObject resource from a JPEG file and adds it to the displayed page.
AddPageMetadataSnip	Shows how to add XMP metadata to a page in a PDF document.
AddStructureSnip	Shows how to work with logical structure within a PDF document. Tags **PDEText** elements within the displayed page.
AddTagSnip	Shows how to add a partial structure tree to an un-tagged PDF document, add tags, and add three types of content items (marked content, structure elements, and complete PDF objects).
AddTextSnip	Adds a string of English, Chinese, Korean, or Japanese text to the bottom left corner of the current viewing page. It demonstrates font embedding/subsetting and support for CJK double-byte fonts in Acrobat. Fonts used in this sample are Times-Roman for English, SimSun for Chinese, Batang for Korean, and MS Mincho for Japanese.
AddXObjectStructureSnip	Shows how to work with logical structure within a PDF document. Tags image XObjects within the displayed page.
ASBigFileSnip	Demonstrates how to manipulate large files.
ASCabPutGetSnip	Demonstrates how to put arbitrary key-value pairs into an **ASCab** object and how to retrieve the corresponding value for a specified key from the **ASCab** object.
ASChangeTempFileSysSnip	Demonstrates how to change the temporary file system implementation for a platform.
ASDateSnip	Gets today's date, and adds different time spans to a date.
ASFileIteratorSnip	Iteratively visits all files in a given folder and its sub-folders, outputting the file names to the DebugWindow.
ASGetConfigurationSnip	Checks for the current configuration of Acrobat, and displays an alert with the return value.
ClassMapSnip	Demonstrates that an application using the PDF Library that processes logical structure can attach additional information (attributes) to any structure element.

Snippet	Description
`ColorSetupSnip`	Shows how to programmatically access the color management settings using Adobe Color Engine (ACE). It enumerates all available system color settings, ICC profiles, and conversion engines. For example: On Windows 2000: C:\Program Files\Common Files\Adobe\ Color\Profiles C:\WINNT\system32\spool\drivers\color On Mac OS: System Folder:Application Support:Adobe:Color: Systems Folder:ColorSync Profiles:
`ConvertOCGsToRadioButSnip`	Creates a radio-button relationship among the OCGs in the document, so that only one OCG in the document can be displayed at any one time.
`CosCryptGetVersionSnip`	Reports the version of the current encryption algorithm, and the maximum number of bytes that can be used for the key when encrypting or decrypting with this version.
`CosDoc64Snip`	Demonstrates 64-bit file position access into CosDoc objects.
`CosObjCompressionSnip`	Shows the use of new Cos layer methods to perform full compression of indirect objects in a document to reduce PDF file size.
`CosObjDecompressionSnip`	Shows the use of new Cos layer methods to perform decompression of compressed indirect objects in a document so as to restore backward viewer compatibility of the document.
`CosObjectExplorerSnip`	Prints out either a shallow or a more complete description of a Cos object.
`CosStream64Snip`	Demonstrates 64-bit APIs for Cos stream object construction and info access. The snippet creates a new PDF document with an embedded image taken from an input file represented as a Cos stream.
`CreateAnnotOCSnip`	Create an optional content group and associates it to any link annotations found on the first page of the front document through an optional content membership dictionary.
`CreateContentXORSnip`	Shows an exclusive OR relationship between optional content groups. When the content of a page is turned off through the layers panel UI, an alternative text object is turned on.

Snippet	Description
`CreateDocStructSnip`	Shows how to create a document structure tree that conforms to tagged PDF conventions.
`CreateImageContentOCSnip`	Converts images within the first page of the front document to optional content. It iterates through images found on the first page of the document and associates them with a newly created optional content group (OCG).
`CreateTextContentOCSnip`	Shows how to convert text into optional text, Converts text within the first page of the front document to optional content.
`ExploreMetadataSnip`	Shows how to access the XMP metadata stream embedded in a PDF file and write metadata to an XML file.
`ExploreStructSnip`	Explores the structure and content of a tagged PDF document and dumps the structure information to the console or to the debug window.
`FontInfoSnip`	Extracts font information from a PDF document, including the name, subtype, and whether the font is embedded.
`GetDocKeywordSnip`	Extracts the keywords from a document's Info dictionary.
`GetDocMetadataSnip`	Shows how to extract PDF document XMP metadata in XML format.
`ImageInfoSnip`	Shows how to obtain specification information of images in a PDF document, including size, width, height, filters, bits per component, and rendering intent.
`JPXColorSpaceExplorerSnip`	Displays color space characteristics of JPX image XObjects.
`JPXPaletteExplorerSnip`	Displays palette information of JPX image XObjects.
`MakeBookmarkSnip`	Makes a bookmark named 'Current Page' for the current page at the top (visible) level of the bookmark tree.
`ObjShiftSnip`	Demonstrates translating page object positions with PDFEdit APIs.
`OCActionControlSnip`	Shows how to control the visibility of optional content using `PDActions`.
`OCGUIReorderSnip`	Reorders and categorizes the OCGs shown in the layers panel UI.

Snippet	Description
`OCTextAutoStateSnip`	Shows how to use an autostate with optional content. Creates an optional content group for the text on the first page of the front document. Sets the OCG's usage dictionary to indicate the text should be visible when the zoom factor is between 1.0 (100%) and 1.5 (150%). Updates the document's optional content properties catalog to indicate that the OCG's zoom category should be used to determine visibility for View events. The text disappears when the zoom is out of the specified range.
`PDCreateMasterOCGSnip`	Creates a parent control OCG for use in the UI. Child OCGs cannot be manipulated through the UI while the parent OCG is Off, although the parent's state does not alter the visibility of the child OCGs.
`PDDocDidDeletePagesNotSnip`	Shows how to register for and receive `PDDocDidDeletePages` notifications.
`PDEContentExplorerSnip`	Writes information about the PDE content for page zero to the DebugWindow.
`PDEPathDrawCurveSnip`	Demonstrates simple spline curve drawing with PDEPath marking APIs.
`PDEPathDrawLineSnip`	Demonstrates simple line drawing with PDEPath marking APIs.
`PDEPathDrawRectSnip`	Demonstrates simple rectangle drawing with PDEPath marking APIs.
`PDEPathExplorerSnip`	Explores PDEPath objects on the current page by breaking them down to PDEPath marking operators and operand(s).
`PDOCConfigCreateSnip`	Creates an optional content configuration, with the OCGs currently in the ON state assigned the default value of ON, and presenting only these OCGs in the layers panel UI.
`PDOCConfigExplorerSnip`	Uses an optional content group callback to display diagnostic information about the OCGs in a document to the DebugWindow.
`PDOCGChangeLockedStateSnip`	Toggles the locked state of each optional content group within the document.
`PDOCGExplorerSnip`	Displays information about optional content configurations to the DebugWindow. Shows the configuration's name, its default state, and lists of OCGs that are on, off and/or locked by default.
`PDOCGToggleIntentSnip`	Toggles the intent of an optional content group between "design" and "view."
`PDOCSetDefaultConfigSnip`	Updates the document's default optional content configuration to use the specified alternative configuration.

Snippet	Description
`PDPageSetTransparencySnip`	Changes the transparency of the elements in the content of the first page of the current document.
`RaiseExcepSnip`	Shows how to register custom exceptions with the application. The exception text associated with the custom exception is passed in as a parameter. We use the error code we get from registering the error string to raise an exception. This exception is caught by the SnippetRunner backstop exception handler.
`RemoveEmbeddedFontSnip`	Removes embedded Roman fonts from a PDF file. An embedded font is removed only if it is not multi-byte, if encoding is not "Identity," and if the charset is Roman.
`RoleMapSnip`	Enumerates all existing role maps in the PDF document. Also provides an example of how to create new role maps or change existing role maps in a tagged PDF document.
`SecureDocumentSnip`	Shows how to apply security settings programmatically.
`SetDocBaseURLSnip`	Shows how to set and get a PDF document metadata property. Sets the base URL from user input.
`SimpleSnip`	Shows how to create a snippet.
`TextChangeColourSnip`	Changes the `PDEText` object of the first page of the front document to specified RGB values.
`TextExtractionSnip`	Demonstrates how to perform text extraction with the new WordFinder creation APIs and configuration structure.
`UserPropertiesExplorerSnip`	Demonstrates the use of PDSEdit APIs to enumerate and obtain UserProperties attributes for PDS elements.

Installing the PDF Library SDK

The Adobe PDF Library SDK installation process differs depending on the target platform.

- Windows — Do one of the following:
  - For the standard version of the PDF Library SDK, unzip **`AdobePDFLSDKMinSize15.x.x.zip`**.
  - For the speed-optimized version of the PDF Library SDK, unzip **`AdobePDFLSDKMaxSpeed15.x.x.zip`**.
- Mac OS — Mount the dmg image, **`AdobePDFLSDK15.x.x.dmg`**, and copy the PDFL SDK 15.0 folder to your local drive.
- Linux — A gzip compressed tar file, **`AdobePDFLSDK15.x.x.tar.gz`**, is provided; developers determine where to decompress and install the SDK.

Updating to PDF Library SDK XV

All PDF Library SDK XI APIs are still supported in PDF Library SDK XV. Other than a few minor code changes, all you should need to do is recompile your application with the new SDK headers using the recommended development environment.

All platforms

Update your font and CMap resources. The following fonts are included with PDF Library SDK XV:

- AdobeArabic-Bold
- AdobeArabic-BoldItalic
- AdobeArabic-Italic
- AdobeArabic-Regular
- AdobeFanHeitiStd-Bold
- AdobeGothicStd-Bold
- AdobeHebrew-Bold
- AdobeHebrew-BoldItalic
- AdobeHebrew-Italic
- AdobeHebrew-Regular
- AdobeHeitiStd-Regular
- AdobeMingStd-Light
- AdobeMyungjoStd-Medium
- AdobePiStd
- AdobeSongStd-Light
- AdobeThai-Bold
- AdobeThai-BoldItalic
- AdobeThai-Italic
- AdobeThai-Regular
- CourierStd-Bold
- CourierStd-BoldOblique
- CourierStd-Oblique
- CourierStd
- KozGoPr6N-Medium

Redirect the path environmental variable as needed, if you do not install the libraries in the same directory as the executable.

Windows

If applicable, change **`PDFL70.lib`** to **`AdobePDFL.lib`** in your project settings.

Mac OS

Refer to the sample projects for information on updating. PDF Library SDK XV uses frameworks instead of shared libraries. For convenience, the sample applications are built to include the PDF Library frameworks. The libraries are in the **`Libs:Mac:AdobePDFL.framework`** folder. Aliases or proper paths need to be set to ensure that your applications run properly.

Note: Regular support for Mac-32 bit has been withdrawn.

Include PDFLInitCommon.c in the **`Include:Headers`** folder to your client project. All C source files must be compiled as C++ files using the following steps:

1. In the project source pane, select the file(s) and click Get Info.
2. In the General panel, change the File Type: popup from sourcecode.c.c to sourcecode.cpp.cpp.

The **`Needs Raise Aware`** flag has been turned off on Mac OS.

Linux

Update your makefile to reflect any new libraries in the **`libs`** folder.

Note: Install libX11 (and its dependent libraries) on RHEL7. This library, which is not a part of core package, is required for building samples and plug-ins on RHEL7.