

Adobe PDF Library 7.0.5



Adobe PDF Library Overview

November 16, 2005



Adobe Solutions Network — <http://partners.adobe.com>

Copyright 2005 Adobe Systems Incorporated. All rights reserved. Adobe® PDF Library Overview.

If this guide is distributed with software that includes an end user agreement, this guide, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. Except as permitted by any such license, no part of this guide may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Adobe Systems Incorporated. Please note that the content in this guide is protected under copyright law even if it is not distributed with software that includes an end user license agreement.

Please remember that existing artwork or images that you may want to include in your project may be protected under copyright law. The unauthorized incorporation of such material into your new work could be a violation of the rights of the copyright owner. Please be sure to obtain any permission required from the copyright owner.

Any references to company names in sample templates are for demonstration purposes only and are not intended to refer to any actual organization.

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, Acrobat Capture, the Adobe PDF logo, 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.

Portions include software under the following terms:

Portions derived from the RSA DataSecurity, Inc. MD5 Message-Digest Algorithm.

This product contains either BISAFE and/or TIPEM software by RSA Data Security, Inc.

Portions of the software were developed by the University of California, Berkeley.

The author of this software is David M. Gay. Copyright (c) 1991 by AT&T. Permission to use, copy, modify, and distribute this software for any purpose without fee is hereby granted, provided that this entire notice is included in all copies of any software which is or includes a copy or modification of this software and in all copies of the supporting documentation for such software. THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED WARRANTY. IN PARTICULAR, NEITHER THE AUTHOR NOR AT&T MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.

Portions of the software were developed at Cygnus Solutions.

Portions copyright (c) 1995-2003 International Business Machines Corporation and others. All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE. Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

Portions copyright (c) 1994 Hewlett-Packard Company. Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. Hewlett-Packard Company makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

Copyright (c) 1996 Silicon Graphics Computer Systems, Inc. Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. Silicon Graphics makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

The TWAIN Toolkit is distributed as is. The developer and distributors of the TWAIN Toolkit expressly disclaim all implied, express or statutory warranties including, without limitation, the implied warranties of merchantability, noninfringement of third party rights and fitness for a particular purpose. Neither the developers nor the distributors will be liable for damages, whether direct, indirect, special, incidental, or consequential, as a result of the reproduction, modification, distribution or other use of the TWAIN Toolkit.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA. Notice to U.S. Government End Users. The Software and Documentation are "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

1

Introduction

About This Document

This document 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. Developers with questions on any of these topics or those new to development with the Adobe PDF Library are encouraged to read this document.

This document 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.

What Is the Adobe PDF Library?

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 process around the Adobe Portable Document Format (PDF). The Adobe 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 Adobe PDF documents.

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

The Adobe PDF Library and the Adobe Acrobat Viewer

The Adobe PDF Library shares much of the same source code as the Adobe PDF viewing applications (Adobe Acrobat 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 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:

<http://partners.adobe.com/public/developer/acrobat/sdk/index.html>

New in the Version 7.0.5 Release

The SnippetRunner can now be run by means of a graphical user interface (see “The SnippetRunner” on page 8).

Contact

For licensing and other information about the PDF Library SDK, see <http://partners.adobe.com/public/developer/pdf/library/index.html>.

Information about Adobe enterprise developer resources can be found at <http://www.adobe.com/enterprise/developer/>

2

About the PDF Library SDK

This chapter helps you get started with development using the Adobe PDF Library Software Developers Kit (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

The Adobe PDF Library 7.0.5 is supported for the platforms, operating systems and compilers listed in the table below.

TABLE 2.1 *Supported Environments for PDF Library 7.0.5*

Platform	Operating System	Compiler
Macintosh PowerPC	Mac OS X 10.2 or higher	CodeWarrior Pro 9.2 (Mach-O)
Windows 32-bit	Windows 2000 /XP/2003 Server	Microsoft Visual Studio .NET 2003
SunOS 32-bit	Solaris 8, 9	gcc 3.2
IBM AIX	AIX 5.1, 5.2	Visual Age (xLC) 6.0
Linux	Red Hat 7.2	gcc 3.2

For Mac OS and Windows, later versions of the same development environment may be supported after they have been tested.

While it may be possible to use the library in other development environments, such use is not supported by Adobe Developer Support. The project files for the sample applications are created and supported only in the listed compiler versions.

Contents of the Software Development Kit

The Adobe PDF Library 7.0.5 SDK consists of:

- 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.

Libraries, Headers, and Fonts

The following components are shipped with the PDF Library 7.0.5 SDK:

- The Acrobat PDF Library

This is a dynamic link library on the Windows platform and a shared object library on UNIX and Mac OS. In Windows, an interface library must be included in your MSVC project. The file names of these libraries are:

 - **PDFL70.lib** — The interface library for the Windows PDF Library DLL
 - **PDFL70.dll** — The Windows PDF Library DLL
 - **libpdf1.so** — The shared object library for supported UNIX platforms
 - **AdobePDFL.framework** — The framework for Mac OS
- Helper Libraries

These libraries are used by the PDF Library internally. Developers should not call any method directly from these libraries.

 - **ACE** — Used for color management
 - **AGM** — Used for rasterization
 - **AdobeXMP** — Support for the Extensible Metadata Platform (XMP)
 - **ARE** — Support for Adobe Raster Express (ARE)
 - **AXE16SharedExpat, AXE8SharedExpat** — Adobe XML Engine
 - **BIB, BIBUtils** — Used for interfacing with other internal helper libraries
 - **CoolType** — Used for font handling
 - **JP2KLib** — Support for JPEG 2000
- Headers

The **include** directory of the SDK contains the required headers for accessing the API methods.

Sample Code

Samples are provided for Mac OS, UNIX, and Windows platforms in two forms:

- Standalone sample programs
- The SnippetRunner, an environment and infrastructure for code snippets that illustrate specific functions or techniques. See [“The SnippetRunner” on page 8](#).

Sample code is intended to demonstrate the use of the PDF Library API and is not necessarily robust enough for a final implementation. The sample code itself is platform-

independent, as is the majority of the PDF Library API; the only difference between the sample source code for different platforms is the line-endings.

The Macintosh samples are provided as application packages. This format is normal for double-clickable applications, but they can also be run from the command line. To run them from the command line, you can either specify the command line arguments in the CodeWarrior project file and execute within the IDE, or you can target the actual executable, which is in the **Contents/MacOS** folder inside the package. For example, from the Terminal window:

```
$ cd helowrld.app/Contents/MacOS/
$ helowrld
```

The “**MT**” (multithreading) samples require command line arguments (a default set is added to the project files). Therefore, execution from within the IDE is preferred. Also, for those samples you must use absolute paths for the command line arguments.

Standalone Samples

The standalone samples include the following:

TABLE 2.2 *PDF Library Samples*

Sample	Description
addelem	Shows how to modify existing pages in a PDF file. It adds a footer to each page and shifts the first line of each text run.
all	Used to compile all samples at the same time. Available on Windows and Macintosh only.
CreatePattern	Shows how to create tiling patterns in a PDF document.
Decryption	Shows how to programmatically decrypt a PDF document encrypted with Acrobat Standard Security options.
drawtomemory	Shows how to render a page to memory using the PDFPageDrawContentsToMemory PDF Library method, and creates a PDF file with a bitmap image rendered on the page.
fontembd	Shows font enumeration and font embedding.
helowrld	Shows the basics of creating a PDF document.
JPXEncode	Re-encodes PDF embedded images with the JPX filter and writes out a new PDF file with the re-encoded images embedded.
mergepdf	Shows how to merge two PDF files.
MTInMemFS	Demonstrates use of an in-memory file system for a simple workflow within a multithreaded context.
MTSerialNums	Demonstrates creation of multiple threads to simultaneously generate multiple PDFs.

TABLE 2.2 PDF Library Samples

MTTextExtract	Demonstrates multiple threads concurrently processing multiple PDF documents.
Peddler	Shows how to add hyperlinking (specifically targetting URIs) capabilities to an existing PDF document.
printpdf	Shows how to print a PDF file to a printer or to a file using the PDF Library method PDFLPrintDoc .

The SnippetRunner

SnippetRunner allows you to quickly prototype code containing PDF Library API calls without the overhead of writing and verifying a complete application. It provides an infrastructure and utility functions to support execution and testing of code snippets, which are small but complete portions of PDF Library application code.

SnippetRunner consists of these major components:

- An application that acts as a back-end server and provides the basic functionality, including a parameter input mechanism, debug support, and exception handling.
- A graphical user interface that acts as a client to the back-end server. (This user interface, called the *Common User Interface*, is also provided with the Acrobat SDK, which uses an Acrobat plug-in for its back end.)

The *SnippetRunner Cookbook* document is provided with the PDF Library SDK. It contains complete information on how the SnippetRunner works and how to write snippets.

In versions of the PDF Library SDK earlier than 7.0.5, the Common User Interface was not provided, and PDF Library clients controlled SnippetRunner through a command-line interface that printed output to the console. You can still use this command-line interface. To do so, build SnippetRunner and run it as you would any other sample (see previous section). You can type **help** at the command line to get a list of SnippetRunnerCommands.

The snippets themselves are arranged in a directory-like hierarchy. The snippets provided include most of those available for the Acrobat SDK. Snippets include the following:

TABLE 2.3 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 access color Setup files programmatically using the Adobe Color Engine (ACE).

TABLE 2.3 PDF Library Snippets

Snippet	Description
ACEGetWorkingSpaceSnip	Gets working space profiles (RGB, Gray, and CMYK) information such as size, description and color spaces information, and dumps 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	(New in 7.0) Shows how to add XMP metadata to an image XObject in PDF.
AddImageSnip	Creates an image XObject resource from a JPEG file and adds it to the displayed page.
AddPageMetadataSnip	(New in 7.0) 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, add tags, and add three types of content items (marked content, structure element, 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 CJK double-byte fonts support 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	(New in 7.0) Demonstrates how to manipulate large files.

TABLE 2.3 PDF Library Snippets

Snippet	Description
<code>ASCabPutGetSnip</code>	(New in 7.0) Demonstrates how to put arbitrary key-value pairs into an <code>ASCab</code> object and how to retrieve the corresponding value for a specified key from the <code>ASCab</code> object.
<code>ASChangeTempFileSysSnip</code>	(New in 7.0) Demonstrates how to change the temporary file system implementation for a platform.
<code>ASDateSnip</code>	Gets today's date, and adds different time spans to a date.
<code>ASFileIteratorSnip</code>	Iteratively visits all files in a given folder and its sub-folders, outputting the filenames to the <code>DebugWindow</code> .
<code>ASGetConfigurationSnip</code>	Checks for the current configuration of Acrobat, and displays an alert with the return value.
<code>ClassMapSnip</code>	Demonstrates that an application using PDF Library that processes logical structure can attach additional information (attributes) to any structure element.
<code>ConvertOCGsToRadioButSnip</code>	Creates a radio-button relationship among the OCGs in the document, so that only one OCG in the document can be on at any one time.
<code>CosCryptGetVersionSnip</code>	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.
<code>CosDoc64Snip</code>	(New in 7.0) Demonstrates 64-bit file position access into <code>CosDoc</code> objects.
<code>CosObjCompressionSnip</code>	Shows the use of new Cos layer methods to perform full compression of indirect objects in a document to reduce PDF file size.
<code>CosObjDecompressionSnip</code>	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.
<code>CosObjectExplorerSnip</code>	Prints out either a shallow or a more complete description of a Cos object.

TABLE 2.3 PDF Library Snippets

Snippet	Description
<code>CosStream64Snip</code>	(New in 7.0) 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.
<code>CreateAnnotOCsSnip</code>	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.
<code>CreateContentXORSnip</code>	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.
<code>CreateDocStructSnip</code>	Shows how to create a document structure tree that conforms to tagged PDF conventions.
<code>CreateImageContentOCsSnip</code>	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).
<code>CreateTextContentOCsSnip</code>	Shows how to convert text into optional text, Converts text within the first page of the front document to optional content.
<code>ExploreMetadataSnip</code>	Shows how to access the XMP metadata stream embedded in PDF and write metadata to an XML file.
<code>ExploreStructSnip</code>	Explores the structure and content of a tagged PDF and dumps the structure information to the console or the debug window.
<code>FontInfoSnip</code>	Extracts font information from a PDF document, including the name, subtype, and whether the font is embedded.
<code>GetDocKeywordSnip</code>	Extracts the keywords from a document's Info dictionary.
<code>GetDocMetadataSnip</code>	Shows how to extract PDF document XMP metadata in XML format.

TABLE 2.3 PDF Library Snippets

Snippet	Description
<code>ImageInfoSnip</code>	Shows how to obtain specification information of images in PDF, including size, width, height, filters, bits per component, and rendering intent.
<code>JPXColorSpaceExplorerSnip</code>	(New in 7.0) Displays color space characteristics of JPX image XObjects.
<code>JPXPaletteExplorerSnip</code>	(New in 7.0) Displays palette information of JPX image XObjects.
<code>MakeBookmarkSnip</code>	Makes a bookmark named 'Current Page' for the current page at the top (visible) level of the bookmark tree.
<code>ObjShiftSnip</code>	Demonstrates translating page object positions with PDFEdit APIs.
<code>OCActionControlSnip</code>	Shows how to control the visibility of optional content using PDActions .
<code>OCGUIReorderSnip</code>	Reorders and categorizes the OCGs shown in the layers panel UI.
<code>OCTextAutoStateSnip</code>	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.
<code>PDCreateMasterOCGSnip</code>	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.
<code>PDDocDidDeletePagesNotSnip</code>	Shows how to register for and receive PDDocDidDeletePages notifications.
<code>PDEContentExplorerSnip</code>	Writes information about the PDE content for page zero to the DebugWindow.

TABLE 2.3 PDF Library Snippets

Snippet	Description
<code>PDEPathDrawCurveSnip</code>	Demonstrates simple spline curve drawing with PDEPath marking APIs.
<code>PDEPathDrawLineSnip</code>	Demonstrates simple line drawing with PDEPath marking APIs.
<code>PDEPathDrawRectSnip</code>	Demonstrates simple rectangle drawing with PDEPath marking APIs.
<code>PDEPathExplorerSnip</code>	Explores PDEPath objects on the current page by breaking them down to PDEPath marking operators and operand(s).
<code>PDOCConfigCreateSnip</code>	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.
<code>PDOCConfigExplorerSnip</code>	Uses an optional content group callback to display diagnostic information about the OCGs in a document to the DebugWindow.
<code>PDOCGChangeLockedStateSnip</code>	(New in 7.0) Toggles the locked state of each optional content group within the document.
<code>PDOCGExplorerSnip</code>	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.
<code>PDOCGToggleIntentSnip</code>	Toggles the intent of an optional content group between "design" and "view."
<code>PDOCSetDefaultConfigSnip</code>	Updates the document's default optional content configuration to use the specified alternative configuration.
<code>PDPageSetTransparencySnip</code>	Changes the transparency of the elements in the content of the first page of the current document.

TABLE 2.3 PDF Library Snippets

Snippet	Description
<code>RaiseExcepSnip</code>	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 <code>SnippetRunner</code> backstop exception handler.
<code>RemoveEmbeddedFontSnip</code>	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.
<code>RoleMapSnip</code>	Enumerates all existing role maps in the PDF. Also provides an example of how to create new role maps or change existing role maps in a tagged PDF.
<code>SecureDocumentSnip</code>	Shows how to apply security settings programmatically.
<code>SetDocBaseURLSnip</code>	Shows how to set and get a PDF document metadata property. Sets the base URL from user input.
<code>SimpleSnip</code>	Shows how to create a snippet.
<code>TextChangeColourSnip</code>	Changes the <code>PDEText</code> object of the first page of the front document to specified RGB values.
<code>TextExtractionSnip</code>	Demonstrates how to perform text extraction with the new <code>WordFinder</code> creation APIs and configuration structure.
<code>UserPropertiesExplorerSnip</code>	(New in 7.0) Demonstrates the use of <code>PDSEdit</code> APIs to enumerate and obtain <code>UserProperties</code> attributes for PDS elements.

PDF Library Documentation

The following documents are helpful in developing with the Adobe PDF Library and are included in the SDK distribution.

- *Adobe PDF Library Overview*—This document, which contains an introduction to programming with the Adobe PDF Library using the SDK.

- *Acrobat and PDF Library API Overview*—An overview of the design and structure of the Acrobat and Adobe PDF Library API. The API is organized into layers as follows:
 - The Acrobat Support (AS) layer provides a variety of utility methods, including platform-independent memory allocation and fixed-point math utilities.
 - The Acrobat Viewer (AV) layer deals with the viewer’s user interface. The AV layer is *not* part of the PDF Library.
 - The Cos Object System (Cos) layer provides access to the low-level building blocks of PDF files.
 - The Portable Document (PD) layer provides access to components of PDF documents.
 - The PDFEdit layer provides access to PDF page contents.
 - The PDSEdit layer provides access to the logical structure of a PDF document.
 - OS-specific functions.
 - Acrobat extended APIs, such as forms and digital signatures. These are *not* part of the PDF Library.

- *Acrobat and PDF Library API Reference*—Documents all of the available API methods, as well as structures used in conjunction with the methods. This document includes APIs that are common to both Acrobat and the Adobe PDF Library, as well as those that are unique to each.

In versions earlier than 7.0, this information was contained in two documents: the *Acrobat Core API Reference* and the *PDF Library Supplement* (which contained information about PDF Library-specific APIs). Some other information that was previously contained in those documents, such as macros, lists, errors, and platform availability, has been moved to the *Acrobat and PDF Library API Overview*, described above.

- *PDF Reference, version 1.6, fifth edition*—Describes the PDF file format and suggests how to produce efficient PDF files. It is intended primarily for developers who wish to produce PDF files directly. This document also contains enough information to allow developers to write applications that read and modify PDF files. This document is also available in published form and can be purchased from <http://www.adobepress.com>.
- *SnippetRunner Cookbook*—Describes architecture of the SnippetRunner tool, how to use it, and how to write snippets.
- *3D Annotations Tutorial*—Describes how to programmatically create 3D annotations in PDF documents.

Installing the PDF Library SDK

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

- Windows and Mac OS — An executable installer is provided for each of these platforms. Run the installer and follow the default installation instructions.
- Solaris, AIX, and Linux — A gzip compressed tar file is provided; developers determine where to decompress and install the SDK.

Updating to PDF Library Version 7

All PDF Library SDK 6.0 APIs are still supported in PDF Library 7.0. 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.

For all platforms:

Update your font and CMap resources. The following fonts are included with PDF Library SDK 7.0.5:

- AdobeMingStd-Light (Traditional Chinese)
- AdobeSongStd-Light (Simplified Chinese)
- AdobeMyungjoStd-Medium (Korean)
- KozMinProVI-Regular and KozGoPro-Medium (Japanese)
- AdobePiStd
- CourierStd, CourierStd-Bold, CourierStd-BoldOblique, CourierStd-Oblique
- Symbol
- Adobe Sans MM, Adobe Serif MM

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

For Mac OS:

Refer to the sample projects for information on updating. PDF Library 7.0 uses frameworks instead of shared libraries. For convenience, the sample applications are built to include the PDF Library frameworks.

For Windows:

Change `PDFL60.lib` to `PDFL70.lib` in your project settings.

For UNIX:

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

3

Developing With the Adobe PDF Library

This chapter introduces the essentials required to develop applications using the Adobe PDF Library. The sample applications that accompany the product, which should build and run directly after installation, demonstrate these concepts.

Building Applications with the Adobe PDF Library

This section details the compiler environment variables (macros) required to build applications against the Adobe PDF Library.

On all platforms, you must define the **PRODUCT** macro for the preprocessor.

```
PRODUCT=\"Library.h\"
```

This macro is used as a trigger for conditional compilation and allows the same headers to be used for both the Acrobat plug-in API and the Adobe PDF Library.

Windows

The following macros must also be defined in the IDE project settings for applications to compile correctly on the Windows platform:

```
WIN_ENV  
WIN32  
WIN_PLATFORM
```

The Adobe PDF Library is compiled with code generation set to “Multithreaded.” Applications linking with the Adobe PDF Library must have code generation settings that match or there will be conflicts between the Microsoft libraries **MSVCRT** and **LIBCMT**.

The **Project Settings > Link > Input > Ignore libraries** settings should *not* ignore **LIBCMT** (other versions of PDF Library do ignore it).

The Adobe PDF Library is distributed as an interface library (**PDFL70.lib**) and matching DLL (**PDFL70.dll**). You should link the interface library into your application.

The operating system must be able to access the Adobe PDF library at runtime. It does so by searching the paths specified by the **PATH** environment variable, as well as the folder in which the application was launched.

Mac OS

The Mac OS libraries use a precompiled header and prefix file to define the appropriate macros. See **Precompile.pch** in the **Samples:utils** directory of the Adobe PDF Library SDK for the macros required to successfully compile the samples.

UNIX

The following macros must be defined for the headers to compile correctly on the UNIX platform:

```
UNIX_ENV=1
UNIX_PLATFORM=1
```

Before you can compile the samples, you must point the makefiles to your **gcc 3.2** or **xlc 6.0** compiler. Make sure the permissions on all libraries are set so that the dynamic loader can find and load the libraries.

```
chmod o+x libraryname
```

Shared objects are provided for AIX, Solaris and Linux. Alter the common makefile for each individual platform/os (i.e., **linux.mak**) under **samples/Utils** directory to specify the **gcc** or **g++** and static library access path.

You will need to set the environment variable **LD_LIBRARY_PATH** to the location of the libraries so that the application will find the shared object libraries at run time. This can be accomplished with the command

```
setenv LD_LIBRARY_PATH path
```

For more information, see your platform's manual pages for the keyword **LD**.

Before you run your application, set the **PSRESOURCEPATH** and **ACRO_RES_PATH** environment variables to point to your fonts.

For example, to set these environment variables manually before you run your application:

```
setenv PSRESOURCEPATH /user/yourname/PSFont
setenv ACRO_RES_PATH /user/yourname/PSFont
```

Alternatively, you can define the environment variables within the application using the **putenv** system call.

Initialization and Termination

Applications must initialize and terminate the Adobe PDF Library appropriately:

- Call **PDFLInit** to set up internal data structures, locate required resources such as fonts, and perform initialization (such as setting client-provided memory allocation routines). Calling most library functions without successfully initializing the library results in error conditions. The rest of this section provides details on using **PDFLInit**.
- Call **PDFLTerm** to clean up before an application terminates or when access to PDF Library functionality is no longer needed.

Since the PDF Library supports thread-safety (since version 6.1.2), initialization and termination are handled on a per-thread basis. See [“Multithreading” on page 19](#) for more details

Initialization details

The `PDFLInit` function takes as a parameter a `PDFLData` structure, defined in the API header file `PDFInit.h`. See the *API Reference* for details on this structure. You must provide valid values for the following members of the structure before passing it to `PDFLInit`:

- `size` denotes the size of the structure and can be obtained with `sizeof(PDFLDataRec)`.
- `listLen` is the number of directories listed in `dirList`.
- `dirList` is an array of directories that contain font resources. The following discussion explains how to use this member on each of the supported platforms.

On Windows and Mac OS, the PDF Library searches for fonts in the default system and in their subdirectories (to 99 levels). You can specify additional font directories to search (also to 99 levels) in the `dirList` array. (Note that this can affect performance.)

Here is an example showing how to pass the font paths to `dirList` for Windows:

```
pdfLibData.dirList[0]=strdup("C:\\Myfontfolder\\CMap");
pdfLibData.dirList[1]=strdup("C:\\Myfontfolder\\CIDFont");
pdfLibData.dirList[2]= strdup("C:\\Myfontfolder\\Font");
```

The paths can be either full paths or paths relative to the directory from which the executable linking in the Adobe PDF Library was launched.

NOTE: You can set the value `kPDFLInitIgnoreDefaultDirectories` in the `flags` field of the `PDFLData` structure to indicate that the default font directories should not be searched but only the directories provided in `dirList`.

For more details, see the functions `PDFLGetDirList_Win` and `PDFLGetDirList_Mac` in the `MyPDFLibUtils.cpp` file in the `Samples/Utils` directory.

On UNIX, the PDF Library searches by default for fonts in the directory from which the application was launched. Use `dirList` member to specify additional locations of font resources.

For more details, see the function `PDFLGetDirList_Unx` in the `MyPDFLibUtils.cpp` file in the `Samples/Utils` directory.

Multithreading

When using the thread-safe PDF library, initialization and termination now additionally need to be performed for each thread that calls into the library, as well as at the process level. The interfaces for per-thread initialization/termination are the same as before (`PDFLInit` and `PDFLTerm`—see “[Initialization and Termination](#)” on page 18).

Since each thread acquires an independent PDF Library memory context, clients are advised not to share PDF Library data and resources among threads. This includes sharing the same PDF file.

Using Threads

The Adobe PDF 7.0 libraries are thread-safe. To use threads, simply make the appropriate system call (`_beginthreadex` on Windows, and `pthread_create` on UNIX). Multiple threads cannot share PDF Library datatypes. However, they share the same process heap; therefore, an application can share generic datatypes between threads. Multiple threads can open the same file read-only; however, multiple threads should not attempt to write to the same PDF document.

NOTE: On Windows, `CreateThread` is not recommended if the application is using most `stdio.h`-defined functions, including file I/O and string manipulation. It is best to use `_beginthreadex` on Windows, which performs extra bookkeeping to ensure thread-safety.

See the section “Thread-safe Multithreading APIs” in the *API Overview* for more information.