Overview

Adobe® Acrobat® SDK

November 2006

Version 8.0

Adobe® Acrobat® SDK 8.0 Overview for Microsoft® Windows®, Mac OS®, Linux®, and UNIX®

Edition 1.0, November 2006

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Contents

Preface

This guide introduces the Adobe® Acrobat® Software Development Kit (SDK). It provides an overview of what you can do with the SDK and the technologies that are available to you through the SDK.

Who should read this guide?

This guide is for both new and experienced Acrobat SDK developers.

For those new to the Acrobat SDK, this guide provides information about Acrobat SDK technologies and the many different ways that you can extend Acrobat using the SDK.

For developers who have used previous versions of the SDK, this guide provides an overview of both new and existing features in the SDK.

Related documentation

The resources in this table can help you learn about the Acrobat 8.0 SDK.

For information about	See
A guide to the documentation in the Acrobat SDK.	*Acrobat SDK Documentation Roadmap*
Known issues and implementation details.	*Readme*
Answers to frequently asked questions about the Acrobat SDK.	*Developer FAQ*
New features in this Acrobat SDK release.	*What's New*
A guide to the sections of the Acrobat SDK that pertain to Adobe Reader.	*Developing for Adobe Reader*
A guide to the sample code included with the Acrobat SDK.	*Guide to SDK Samples*
Prototyping code without the overhead of writing and verifying a complete plug-in or application.	*Snippet Runner Cookbook*
Configuring and administering a system for online collaboration using comment repositories, Acrobat and Adobe Reader®.	*Acrobat Online Collaboration: Setup and Administration*
Enabling Acrobat to save documents in a customized text-based format.	*Extending the Acrobat SaveAsXML Plug-in*
Using DDE, OLE, Apple events, and AppleScript to control Acrobat and Adobe Reader and to render Adobe PDF documents.	*Developing Applications Using Interapplication Communication*

For information about	See
Detailed descriptions of DDE, OLE, Apple event, and AppleScript APIs for controlling Acrobat and Adobe Reader or for rendering PDF documents.	*Interapplication Communication API Reference*
Detailed descriptions of JavaScript™ APIs for adding interactivity to 3D annotations within PDF documents.	*JavaScript for Acrobat 3D Annotations API Reference*
Using JavaScript to develop and enhance standard workflows in Acrobat and Adobe Reader.	*Developing Acrobat Applications Using JavaScript*
Detailed descriptions of JavaScript APIs for developing and enhancing workflows in Acrobat and Adobe Reader.	*JavaScript for Acrobat API Reference*
Using RSS to track remote resources in an occasionally-connected environment.	*Acrobat Tracker*
Detailed descriptions of APIs for controlling Acrobat Distiller for PDF file creation.	*Acrobat Distiller API Reference*
Specifying settings for the creation of PDF files.	*Adobe PDF Creation Settings*
A detailed description of an extension to the PostScript® language which allows the description of PDF features not found in standard PostScript.	*pdfmark Reference*
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 Adobe PDF Library applications.	*Acrobat and PDF Library API Reference*
Detailed descriptions of the APIs for using assistive technology with PDF documents.	*PDF Accessibility API Reference*
Using JavaScript to perform repetitive operations on a collection of files.	*Batch Sequences*
A detailed description of the parameters for opening PDF files and for performing actions on them using a URL or command.	*Parameters for Opening PDF Files*
A list of the U3D elements supported by Acrobat	*U3D Supported Elements*

For further information about the Acrobat SDK, see http://www.adobe.com/go/acrobat_developer.

1 Introduction

This chapter introduces the Acrobat SDK and its development technologies.

For a description of Acrobat, the family of Acrobat products, the list of tools used to create PDF files, and related Adobe technologies, see the *Developer FAQ*.

Acrobat SDK development technologies

The Acrobat SDK is a set of tools that help you develop software that interacts with Acrobat technology. The SDK contains header files, type libraries, simple utilities, sample code, and documentation.

Using the Acrobat SDK, you can develop software that integrates with Acrobat and Adobe Reader in several ways:

JavaScript — Write scripts, either in an individual PDF document or externally, to extend the functionality of Acrobat or Adobe Reader.

Plug-ins — Create plug-ins that are dynamically linked to and extend the functionality of Acrobat or Adobe Reader.

Interapplication communication — Write a separate application process that uses interapplication communication (IAC) to control Acrobat functionality. DDE and OLE are supported on Microsoft® Windows®, and Apple events/AppleScript on Mac OS. IAC is not available on UNIX®.

The Acrobat SDK provides support for development on both Windows and Apple Mac OS environments. For a UNIX version of the Acrobat SDK, see http://www.adobe.com/go/acrobat_developer.

Besides the technologies provided by the Acrobat SDK, you can also use the PDF Library (PDFL) to develop applications that create and manipulate PDF documents but do not interact with Acrobat. For more information, see "Adobe PDF Library" on page 12.

JavaScript

JavaScript is a cross-platform scripting language. Through its JavaScript extensions, Acrobat exposes much of the functionality of Acrobat and its plug-ins to the document author. The JavaScript objects, properties and methods can also be accessed through Visual Basic or C# to automate the processing of PDF documents.

Acrobat defines several objects that allow your code to interact with the Acrobat application, a PDF document, or fields within a PDF document. The most commonly used objects control the Acrobat or Adobe Reader application, the JavaScript console, the PDF document, SOAP web services, databases, security, searches, and JavaScript events.

JavaScript can be applied at a variety of levels. Each of the levels represents a context in which processing occurs, which affects when the scripts are loaded and how they are accessed inside and outside documents.

The placement of a script at a given level also determines its reusability. For example, folder level scripts are available within all documents, document level scripts are available to all fields within a given document, and field level scripts are visible only to the fields with which they are associated.

For more information on JavaScript objects, see *Developing Acrobat Applications Using JavaScript* and the *JavaScript for Acrobat API Reference*.

Plug-ins

Plug-ins are dynamically-linked extensions to Acrobat or Adobe Reader. They can hook in to the user interface in a number of ways and can register to be called when a variety of events occur in the application.

A plug-in is written in ANSI C/C++ and uses the Acrobat public APIs. It can add functionality to Acrobat Professional, Acrobat Standard, or Adobe Reader. A plug-in program file goes into a Plug_ins folder or directory and is initialized during Acrobat or Adobe Reader startup.

There are three types of plug-ins:

Regular Acrobat plug-ins — These plug-ins run on Acrobat Professional and Acrobat Standard. Plug-ins for Acrobat Professional can use any of the Acrobat SDK APIs. Plug-ins for Acrobat Standard do not have access to some APIs. For more information, see *Developing Plug-ins and Applications*.

Adobe Reader-enabled plug-ins — These plug-ins use a restricted set of APIs. Adobe Reader-enabled plug-ins are developed with permission from Adobe and require special processing to load under Adobe Reader. Plug-ins for Adobe Reader can use additional APIs if the PDF document has additional usage rights. For more information, see "Rights-Enabled PDF Documents and Security" on page 32.

Certified plug-ins — These plug-ins have undergone extensive testing to ensure that they do not compromise the integrity of Acrobat's security model. A checkbox in the Acrobat and Adobe Reader user interface can be used to ensure that only certified plug-ins are loaded. Certified plug-ins can be provided only by Adobe.

On Windows, plug-ins are DLLs. However, plug-in names must end in .API, not .DLL. On Mac OS, plug-ins are code fragments, whereas on UNIX, plug-ins are shared libraries.

Plug-in development environments

Windows developers can develop plug-ins using C and C++ with Visual Studio.

There is currently no support for development of plug-ins using managed languages such as C# or VB.NET. However, managed languages are supported for use with interapplication communication (IAC). This enables those languages to take full advantage of Acrobat's functionality through use of the JavaScript bridge. For more information, see "Interapplication communication" on page 11.

All plug-ins developed on Mac OS X must use the Mach-O runtime architecture and must be built as a bundle. Apple Xcode 2.3 is required because SDK projects depend on certain header files that are included with the Xcode development environment.

Acrobat core API

Plug-ins access and control the resources of the Acrobat application host environment using the Acrobat core API. The core API consists of a set of methods that operate on objects. The objects have types and encapsulate their data. This object orientation is a conceptual model, implemented using a standard ANSI C programming interface. Methods are C functions; objects are opaque data types. The core API is supported on Windows, Mac OS, and UNIX platforms.

The API is organized into several layers.

Layer	Description
Acrobat Viewer	The AV layer, also known as AcroView or AV Model, works with the Acrobat or Adobe Reader application. Its methods allow plug-ins to manipulate components of the Acrobat or Adobe Reader application itself, such as menus and menu items.
Portable Document	The PD layer, also known as PDModel, provides access to components of PDF documents. Its methods allow plug-ins to manipulate document components such as document pages and annotations.
Acrobat Support	The AS layer provides platform-independent utility functions and allows plug-ins to override the built-in file-handling mechanisms.
Cos	The Cos Object System layer provides access to the building blocks used to construct documents. Cos methods allow plug-ins to manipulate low-level data such as dictionary and string objects in a PDF file. Whenever possible, you should use higher level APIs to access and manipulate PDF files. Though you can use the Cos layer APIs to perform most types of access or manipulation of a PDF file, it can be difficult and requires in-depth knowledge of PDF file structure.

The core API also includes platform-specific plug-in utilities to handle issues that are unique to Windows, Mac OS, and UNIX platforms. For more information, see *Developing Plug-ins and Applications*.

Extended APIs for plug-ins

Plug-ins can expose their own functionality and make it available to other plug-ins in the same way that Acrobat functionality is available through the core API. Acrobat uses many plug-ins to implement features, such as the Search and Digital Signature plug-ins. In fact, the Acrobat architecture encourages the use of plug-ins to expose APIs for use by other plug-ins.

API exposure is accomplished through a mechanism called the Host Function Table (HFT). A plug-in can export an HFT for use by other plug-ins, and it can import the HFTs of other plug-ins. The following Adobe plug-ins export HFTs:

- Catalog
- Digital Signature
- Forms
- PDF Consultant

- Search
- Spelling
- Weblink
- SaveAsXML

For more information on plug-ins and HFTs, see *Developing Plug-ins and Applications*, the *Acrobat and PDF Library API Reference*, and *Extending the Acrobat SaveAsXML Plug-in*.

Interapplication communication

Acrobat provides support for interapplication communication (IAC) through OLE automation and DDE on Windows, and through Apple events and AppleScript on Mac OS. Adobe Reader also supports IAC, but does not support OLE on Windows.

IAC support allows programs to control Acrobat or Adobe Reader in much the same way a user would. You can also use IAC support to render a PDF file into an external application window instead of the Acrobat window. The IAC methods and events serve as wrappers for some of the core API calls in the SDK.

On Windows, you can develop IAC applications using Visual Basic .NET, Visual C++ .NET, or Visual C# .NET. On Mac OS, you develop IAC applications using Xcode. CodeWarrior is not supported.

For more information about IAC, see *Developing Applications Using Interapplication Communication* and the *Interapplication Communication API Reference*.

Windows support

Acrobat is an OLE server and also responds to a variety of OLE automation messages. You can embed PDF documents into documents created by an application that is an OLE client. Adobe Reader does not support OLE.

On Windows, you can display a PDF document in applications using simplified browser controls. In this case, the PDF is treated as an ActiveX document, and the interface is available in Adobe Reader.

Once the PDF document is loaded, you can implement browser controls to perform the following tasks:

- Determine which page to display
- Control view and zoom modes
- Determine whether to display bookmarks, thumbnails, scrollbars, and toolbars
- Print pages using various options
- Highlight a text selection

JavaScript support

Acrobat provides a rich set of JavaScript programming interfaces that are designed to be used from within the Acrobat environment. It also provides a mechanism that allows external clients to access the same functionality from environments such as VB .NET, Visual C++ .NET and Visual C# .NET.

Apple event support

The Acrobat viewers support Apple events and a number of Apple event objects on Mac OS. IAC support includes some of the objects and events described in *Apple Event Registry: Standard Suites*, as well as Acrobat-specific objects and events.

You can find information on Apple events supported by the Acrobat Search plug-in by referring to the *Acrobat and PDF Library API Reference*. Other plug-ins supporting additional Apple events are described in *Developing Plug-ins and Applications*.

Plug-ins for IAC

You can extend the functionality of the IAC interfaces by writing plug-ins that use core API objects that are not already part of the IAC support system. The Acrobat SDK provides a sample that demonstrates this. For more information, see the *Guide to SDK Samples*.

Adobe PDF Library

The Adobe PDF Library is based on the core technology of the Acrobat line of products and offers complete functionality for generating, manipulating, rendering, and printing Adobe PDF documents.

Designed specifically for OEMs, ISVs, system integrators, and enterprise IT developers, the Adobe PDF Library SDK contains a set of functions for developing third-party solutions and workflows around PDF. The library enables PDF functionality to be seamlessly embedded within applications without the presence of Acrobat or Adobe Reader. It also provides a reliable, accurate, and Adobe-supported implementation of the latest PDF specification.

There is significant overlap between the functionality provided by the PDF Library SDK and 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 user interface elements—that is, the AcroView (AV) layer of the core API.

For more information on the PDF Library, see the *Acrobat and PDF Library API Reference* and http://www.adobe.com/go/acrobat_developer.

2 Deciding Which Acrobat SDK Technology to Use

You can develop software that integrates with Acrobat and Adobe Reader in three ways: JavaScript, plug-ins, and IAC.

Based on your application's requirements, choose the appropriate technologies for development. In some situations, the desired functionality is only available using one technology. In other situations, you can choose between two or more technologies. For example, you can add menu items to Acrobat using either JavaScript or a plug-in.

You can also use more than one technology in a single application or single document. For example, you can use both plug-ins and JavaScript to implement a particular feature.

Deciding between plug-ins and JavaScript

This section presents topics to consider as you decide whether to use plug-ins or JavaScript to implement a particular functionality.

Comparison of plug-ins and JavaScript

	Plug-ins	JavaScript
Scope	A plug-in affects all PDF documents viewed by Acrobat.	JavaScript can affect either a single document or all PDF documents.
Installation and distribution	Plug-ins must be placed in the Plug_ins folder or directory by an installer or by the user.	Document-level JavaScript code is easier to distribute since it can be included directly within the PDF file and does not require an installer. Folder-level JavaScript code must be placed in the Acrobat application JavaScript folder or the user's JavaScript folder.
Low-level access	Plug-ins can access and manipulate low-level objects in the PDF object model, such as the Cos layer.	JavaScript can access a limited set of AV and PD layer objects.
Execution speed	Plug-ins are compiled and loaded when Acrobat initializes.	Execution of JavaScript code is slower than plug-in code because it is interpreted instead of compiled. However, the difference is noticeable only in computation-intensive applications, such as a full-text search in a large PDF file.

	Plug-ins	JavaScript
Ease of implementation	Plug-ins are developed in C or C++ and are compiled and linked in the appropriate development environment. You must include all necessary header files for your application.	JavaScript scripts are easier to write and implement since they are developed using the editor and debugger that come as part of Acrobat Professional. Developers can also use an external editor to create and edit JavaScript code.
Cross-platform compatibility	Plug-ins must be built on different platforms to handle certain platform-specific issues.	JavaScript is cross-platform compatible.

Situations where plug-ins are better than JavaScript

In general, plug-ins allow for more direct control over Acrobat than JavaScript. There is a richer set of APIs that you can use from a plug-in. For example, the following tasks can only be done using a plug-in; there is no equivalent JavaScript functionality:

- Accessing Cos and other low-layer objects
- Manipulating PDF content streams
- Creating new annotation or action types
- Modifying the ASFixed scale factor for large PDF file sizes
- Accessing platform-specific services or events
- Getting and setting wireframe drawing mode

Situations where JavaScript is better than plug-ins

The following example tasks can be done easily using JavaScript, but have no equivalent plug-in API:

- Using SOAP and other web services
- Manipulating multimedia in PDF documents
- Setting up an automated email review workflow
- Describing the state model for the review
- Searching Acrobat Help
- Using Acrobat security policies

Some of these example tasks, such as SOAP and web services, can in fact be done with a plug-in by using low-level APIs. However, this is a time-consuming approach and requires an in-depth knowledge of the low-level APIs.

When to use IAC

To take advantage of Acrobat functionality from within an external application, use interapplication communication (IAC).

Viewing PDF documents from an external application

If your Windows application only views a PDF document and does not need to edit it in any way, use the PDF Browser Controls to view the document from your external VB or C# application. When you open a document for viewing using the PDF Browser Controls, the document is displayed in the application window. Acrobat toolbars are also displayed and can be used with no additional API calls. The toolbars can be hidden. Acrobat or Adobe Reader must be installed for the PDF Browser Controls to work.

You can also use the IAC API to open and view a PDF document. However, when you use the IAC API, no toolbars are displayed. You must place your own buttons with corresponding API calls for standard toolbar tasks such as printing and searching.

Controlling Acrobat from an external application

If you need to do more than just view a PDF document from your application, you can use the IAC API to perform these tasks:

- Get annotations, text and form data from a PDF document
- Search a PDF document
- Manipulate a PDF document, editing and adding content
- Control Acrobat (but not Adobe Reader) remotely

You can also extend the functionality of the IAC interfaces by writing plug-ins that use core API objects not already part of the IAC support system. For more information on IAC, see *Developing Applications Using Interapplication Communication* and the *Interapplication Communication API Reference*.

Samples provided with the Acrobat SDK

The Acrobat SDK has a large number of sample applications, plug-ins, and scripts to demonstrate how to use the SDK technologies. Reviewing the samples will help you understand when to choose JavaScript, plug-ins, or IAC functionality for a particular implementation.

For more information on the samples, see the *Guide to SDK Samples* and the *Snippet Runner Cookbook*. Samples are regularly added to the Acrobat SDK. For the latest samples, see http://www.adobe.com/go/acrobat_developer.

3 PDF File Creation

This chapter provides information about how you can use the Acrobat SDK to create PDF files.

Creating PDF files from an authoring application

Authoring applications can simplify the creation of PDF files by making the following steps appear seamless:

- Invoking Acrobat Distiller® through Apple events or DDE
- Printing to a PostScript file
- Automatically generating advanced Acrobat features using pdfmark

Acrobat Distiller

Acrobat Distiller is essentially a PostScript interpreter that can be used to convert PostScript to PDF. Distiller is the PDF creation application intended for batch processing and for the creation of PDF files containing high-end print publishing features such as OPI comments, CMYK color spaces, and spot colors. Distiller also has the ability to interpret PostScript extensions called pdfmarks and to convert them into PDF objects such as links, bookmarks, optional content (OC), and annotations.

On Windows, you can automate Distiller through OLE automation (commonly referred to as ActiveX or COM). The automation interface makes it easy to start and control Distiller from any programming language that supports automation. Distiller supports programming environments written for Visual Basic and for Visual C++ with and without MFC.

The Mac OS version of Distiller supports Apple events. Apple events can be used from external applications written in programming languages such as C or from AppleScript. Because AppleScript is more straightforward, it is recommended.

For more information on Distiller, see the Distiller Help and the *Acrobat Distiller API Reference*.

Automated creation of PDF documents from Windows

The Acrobat SDK has a set of APIs that allow you to automate the creation of PDF files from an external Windows application.

When you use OLE automation, the Distiller splash screen and status are displayed even when the process is automated. With the API, there is no interaction with the user and no display on the screen. The API allows you to specify the name and location of the output PDF file. To do so, you must modify the Windows registry. For more information, see the *Acrobat Distiller API Reference*.

Automatic generation of advanced Acrobat features

The most convenient place to generate advanced Acrobat features is in the authoring application itself, because that application defines the structure of the document. Advanced PDF features are generated using the pdfmark PostScript operator. The authoring application must generate a PostScript language file that contains the appropriate pdfmark operators for the document structure. The resulting PostScript language file is converted into a PDF file by Distiller. In addition, Acrobat allows logical structure information to be added to a document. For instance, paragraphs in a page's contents can be associated with a structural element representing a paragraph.

For more information, see the *pdfmark Reference*.

Attaching a native document to a PDF file

Another way that an authoring application can integrate with Acrobat or Adobe Reader is to allow the user access to the original authoring document through the Acrobat user interface. Through the use of private data in a PDF file, an authoring application can embed the entire authoring document as part of the PDF file that represents it. This way, not only can the resulting electronic document be viewed by anyone using Acrobat, it can also be edited by users who have the authoring application.

You must write a plug-in for Acrobat to allow users to embed and extract the authoring document. This plug-in would add the authoring document as a private data stream when embedding, and, when extracting, save the stream to a temporary file and invoke the authoring application.

Note: Acrobat and Adobe Reader ignore private data. Embedding authoring documents in PDF files greatly increases the size of the PDF file.

An association between the PDF file and the authoring document can also be maintained through the use of links in the PDF file. Links can be created that invoke files and their associated applications. If a document management system places such a link in the PDF file, users can invoke the original authoring document by executing the link.

Batch processing with Distiller

You do not need to use the Distiller API to integrate Distiller with your product. Distiller has the ability to watch directories over a network and to convert any PostScript files saved to those directories to PDF. It is also possible to set different job options for each directory so that one directory can be used, for example, for high-end color output, while the other can generate a more compressed file suitable for web use. These features of Distiller are not supported by Acrobat Developer Support. Check the help documentation packaged with the product or books by Adobe Press and other publishers for using Distiller through the user interface.

The Distiller API can be used to programmatically process files and set the output path and file names. For more information, see the *Acrobat Distiller API Reference*. The Acrobat Professional product can also be used to set up watched folders.

Tagged PDF documents

PDF files are well known for representing the physical layout of a document; that is, the page markings that comprise the page contents. In addition, PDF provides a mechanism for describing logical structure in PDF files. This includes information such as the organization of the document into chapters and sections, as well as figures, tables, and footnotes.

Tagged PDF is a particular use of structured PDF that allows page content to be extracted and used for various purposes, including:

- Reflow of text and graphics
- Conversion to file formats such as HTML and XML
- Access for the visually impaired

For more information, see the *pdfmark Reference* and the *PDF Reference*.

Creating PDF files using plug-ins or JavaScript

You can use Acrobat and the Acrobat SDK to create a new, empty PDF file and to create a PDF file from supported file types.

Empty PDF files

Using either a plug-in or JavaScript, you can dynamically create a PDF file and modify its contents in an automated fashion. This can help make a document responsive to user input and enhance the workflow process.

JavaScript provides support for dynamic PDF file creation and content generation. Using a plug-in, you can also create a new, empty PDF file. You can then use methods to add content to the created file.

PDF files from multiple files

It is possible to use either a plug-in, IAC, or JavaScript to dynamically add content from other sources into a new PDF file. The sources can include files whose types conform to Multipurpose Internet Mail Extensions (MIME) type definitions.

Using JavaScript, you can import an external file into a PDF document. After doing this, it is possible to extract information from the data object for placement and presentation within the PDF document. You can automate the insertion of multiple PDF files into a single PDF document through a series of method calls, and you can also use a portion of the current document to create a new document.

With a plug-in or IAC, you can extract data from other file types, and then bind the resulting PDF files into one PDF file. The relevant APIs can also be executed directly from an IAC application.

Creating PDF files without using Acrobat

Some developers have developed the capability of generating PDF from their own applications without using Adobe products. Some of these developers have used the Adobe PDF Library product to extend their own application. Others have built PDF-generation capability into their applications from scratch. This type of development is not supported by Adobe.

The best resource for building PDF-generation capability is the *PDF Reference*, which fully documents the PDF specification. The PDF file format is complex, and developing code to generate it requires a significant amount of development.

4 Working with PDF Features

This chapter discusses how you can work with various features of PDF documents using the Acrobat SDK.

Navigation in PDF documents

This section describes ways that you can add navigation to a PDF file using the Acrobat SDK.

Bookmarks

A bookmark corresponds to an outline object in a PDF document. A document outline consists of a tree-structured hierarchy of bookmarks that displays the document's structure, allowing the user to navigate interactively from one part of the document to another. Each bookmark has a title that appears on screen and an action that specifies what happens when the user clicks on the bookmark.

Bookmarks can be either created interactively through the Acrobat user interface or generated programmatically. Typically, a user-created bookmark is used to move to another location in the current document, although other actions can be specified, such as opening a web link, opening a file, playing a sound, or executing JavaScript.

You can use JavaScript, a plug-in, or IAC to get or set the following properties:

- The open attribute of a bookmark
- The text used for the bookmark's title
- The action that is invoked when the bookmark is selected

Thumbnails

Page thumbnails are miniature previews of the pages in a document. You can use page thumbnails in Acrobat to jump quickly to a selected page and to adjust the view of the page.

When you move, copy, or delete a page thumbnail, you actually move, copy, or delete the corresponding page. You can embed page thumbnails in a PDF document so that they need not be redrawn every time you select the Pages tab. They can easily be removed and embedded again if necessary.

You can use the Acrobat SDK to create and remove thumbnails in a PDF document.

Links

Links, or hyperlinks, let users jump to other locations in the same document, to other electronic documents, or to websites. You can use links when you want to ensure that your reader has immediate access to related information. You can also use links to initiate actions.

The Acrobat SDK provides support for the addition, customization, or removal of links within PDF documents. These links can be used to access URLs, file attachments, or destinations within the document.

In addition, JavaScript can be used to specify whether cross-document links are opened in the same window or in a new one.

Actions for special effects

Thumbnails, bookmarks, links, and other objects have actions associated with them, and you can use JavaScript to customize their behavior. For example, you can use them to display messages, jump to destinations in the same document or any other, open attachments, open web pages, execute menu commands, or perform a variety of other tasks.

You can also customize these actions so that they change their appearance after the user has clicked on them.

PDF page manipulation

You can use the Acrobat SDK to insert or remove pages from a PDF document. For example, you can do the following tasks:

- Create an empty page in the current document (not with IAC)
- Insert pages from another document into the current document
- Move a page to another location in the same document (not with IAC and AppleScript)
- Replace pages with pages from another document
- Delete pages from the current document

You can also access JavaScript functionality from an external application.

Page content

Page content is a major component of a PDF file. It represents the visible marks on a page that are drawn by a set of PDF marking operators. The set of marking operators for a page is also referred to as a *display list*, since it is a list of marking operations that represent the displayed portion of a page. For more information on page content streams, see the *PDF Reference*.

You can use the PDFEdit API to access PDF page contents. With PDFEdit, your plug-in can treat a page's contents as a list of objects rather than manipulating the content stream's marking operators. There is no JavaScript equivalent to the PDFEdit API to allow you to manipulate page content. For more information on this API, see the *Acrobat and PDF Library API Reference* and the *Snippet Runner Cookbook*.

Document logical structure

You can insert logical structure into a PDF document by creating a tagged PDF document. The PDSEdit API provides the ability to add, modify, and view this logical structure. For more information, see the *Acrobat and PDF Library API Reference*.

Authoring applications can also add structure pdfmarks to the PostScript language code generated when a document is printed. For more information, see the *pdfmark Reference*.

Other ways of modifying PDF documents

You can use a plug-in or JavaScript to modify a PDF document by cropping and rotating pages, numbering pages, and adding headers and footers. For more information, see the *Acrobat JavaScript Scripting Guide* and the *Acrobat and PDF LIbrary API Overview*.

Watermarks

JavaScript provides methods to create watermarks within a document, and place them in optional content groups (OCGs). You can also add watermarks using C APIs. The Acrobat SDK contains a sample plug-in that adds a watermark. For more information, see the *Guide to SDK Samples*.

Spell-checking

Acrobat provides a Spelling plug-in that can scan a document for spelling errors. Using any of the Acrobat SDK technologies, you can do the following tasks:

- Add or remove a dictionary from the list of available dictionaries
- Add or remove a spelling domain (search scope) from the Spell Check dialog box
- Add or remove a word in the user's dictionary
- Check the spelling of an individual word
- Ignore all occurrences of a word in a document when spell-checking
- Scan a text buffer and return the next word
- Set the document's dictionary search order
- Set the document's dictionary search order from an array of ISO 639-2 and 3166 language codes, allowing you to identify a dictionary by language rather than by name

The following additional functionality is available to plug-ins and external applications, but is not available using JavaScript:

- Check a text object and optionally receive a callback for each change as the user interacts with the Spell Check dialog box
- Count the words in a text buffer that are contained in each of a set of dictionaries
- Create a new custom user dictionary and add it to the list of available dictionaries

Multimedia

Multimedia objects can be included in the content of PDF documents, as well as in annotations. You can only manipulate multimedia objects and players using JavaScript; you cannot use a plug-in. Using JavaScript, you can perform the following tasks:

- Customize the settings, renditions, and events associated with media players
- Access and control the properties for all monitors connected to the system
- Add movie and sound clips
- Add and edit renditions
- Control rendition settings
- Set multimedia preferences that apply throughout a document

For more information, see *Developing Acrobat Applications Using JavaScript* and the *JavaScript for Acrobat API Reference*.

Printing PDF files

Using the Acrobat SDK, you can control the way that Acrobat, Adobe Reader, or your external application prints PDF files. Using any of the Acrobat technologies, you can customize the way that a PDF document is printed.

Since printing involves sending pages to an output device, there are many options that can affect print quality. JavaScript can be used to enhance and automate the use of these options in print production workflows. For more information, see *Developing Acrobat Applications Using JavaScript*.

With plug-ins, you can use methods to customize and control how a PDF file is printed from Acrobat or Adobe Reader. Depending on the methods you use, a user interface may be invoked. For example, you can use a method to print a document to the current printer using the current print settings and job settings with no user interface. You can specify a page range, a PostScript version, and whether to shrink the pages to fit the printer. For more information, see *Developing Plug-ins and Applications*.

Using the IAC APIs, you can print a PDF file from an external application. For more information, see *Developing Applications Using Interapplication Communication*.

Embedded fonts

Acrobat Distiller and the PDF Library add font embedding information to fonts that are embedded in PDF files. With the inclusion of this information, your code can determine how an embedded font can be used. These operations also may apply to code used with the PDF Library SDK.

Acrobat plug-in developers can remove and embed fonts in an existing PDF document. You can also use fonts that are already embedded in a PDF document for preview and printing, as well as editing. However, allowing editing using embedded fonts is not recommended, and in some cases it is impractical. For example, Chinese, Japanese and Korean (CJK) fonts potentially include thousands of glyphs, so applications must subset these fonts when embedding them in a PDF file. This precludes embedded CJK fonts from being used for editing by a plug-in.

PDF Library users can perform these operations using an existing PDF document, or they can create a PDF document from scratch that includes embedded fonts. Creating a document from scratch cannot be performed by a plug-in, but it can be done by using PDF Library calls from within a compiled application that includes the PDF Library.

For more information on embedded fonts, see *Font Embedding Guidelines for Adobe Third-party Developers* at http://www.adobe.com/go/acrobat_developer.

5 User Interface Modifications

This chapter describes how you can modify menus and toolbars in the Acrobat or Adobe Reader user interface using the Acrobat SDK.

Menu items and menus

You can use the Acrobat SDK to manipulate menu items, menus, and menu bars. For complete information on what functionality is available with each SDK technology, see the *Acrobat and PDF Library API Reference,* the *JavaScript for Acrobat API Reference*, or the *Interapplication Communication API Reference*.

Menu items

You can use a plug-in or JavaScript to add or remove menu items or submenus to or from an existing menu. With IAC OLE automation, you can remove a menu item but you cannot add one.

Menu items added by plug-ins can have shortcuts (keyboard accelerators). Acrobat and Adobe Reader do not ensure that plug-ins add unique shortcuts, but it is possible for a plug-in to check which shortcuts are already in use before adding its own. The only way to ensure there are no shortcut conflicts is for all plug-ins to check for conflicts before adding their own shortcuts.

You are encouraged to have your plug-in add its menu items to the Tools menu. When it is launched, Acrobat or Adobe Reader automatically adds this menu, as well as the About Plug-ins and Plug-in Help menus (see "Plug-in help files" on page 24). After Acrobat or Adobe Reader loads all plug-ins, it checks these three menus and removes any that are empty.

Adobe keeps a registry of plug-in menu item names to help avoid conflicts between plug-ins. For more information on registering and using plug-in prefixes, see *Developing Plug-ins and Applications*.

Menus

You can create a new menu from a plug-in. However, you cannot create menus or add menus to menu bars using JavaScript or IAC.

Toolbars

You can use the Acrobat SDK to manipulate tool buttons and toolbars. For complete information on what functionality is available with each SDK technology, see the *Acrobat and PDF Library API Reference*, the *JavaScript for Acrobat API Reference*, or the *Interapplication Communication API Reference*.

Items on a toolbar

You can add or remove buttons to the toolbar, although the size and resolution of the user's monitor can limit the number of tool buttons that can be added.

Toolbar creation

Plug-ins also can create new toolbars, called flyouts, that can be attached to buttons on the main toolbar. The selection tools, for example, are all on a flyout. Not all tool buttons are located on the main toolbar; some may be located on a flyout.

You cannot create a flyout with JavaScript or IAC.

Customization of Acrobat Help

You can manipulate specific parts of the Acrobat Help system using a plug-in. There is no equivalent functionality from JavaScript or IAC.

About dialog box and splash screen

Plug-ins can set values in the preferences file to prevent the Acrobat or Adobe Reader About dialog box or splash screen from appearing before displaying the first document. These changes take effect the next time Acrobat or Adobe Reader is launched.

Plug-in help files

The Help directory that accompanies Acrobat or Adobe Reader provides a standard location for your plug-in help files. About Acrobat Plug-Ins is a standard menu item in the Help menu. This menu item contains a submenu. You are encouraged to have your plug-in add a menu item to the submenu to display its own About box.

You can place a help file either in the Help directory or in a subdirectory of the Help directory. If, for example, your plug-in is localized for Japanese, you can place the Japanese help file in its own subdirectory. For more information, see the *Acrobat and PDF LIbrary API Reference*.

Customization of the How To panel

The core API provides several methods to customize the How-To panel from a plug-in. For more information, see the *Acrobat and PDF LIbrary API Reference*.

6 Annotations and Online Collaboration

This chapter describes how to use the Acrobat SDK to assist with annotations and online collaboration workflows.

Acrobat 8.0 introduces the Shared Reviews feature, which you should use for most online collaboration processes. You cannot initiate a Shared Review using the Acrobat SDK. For information on Shared Reviews, see the Acrobat Help.

If you need to create a custom collaboration environment, Acrobat also provides a platform for the development of collaborative review systems to fit various needs. For information about setting up a custom online collaboration system, see *Acrobat Online Collaboration: Setup and Administration*.

About annotations

An annotation associates an object such as a note, sound, or movie with a location on a page of a PDF document, or provides a way to interact with the user through the mouse and keyboard. PDF includes a wide variety of standard annotation types, described in detail in the *PDF Reference*.

Many of the standard annotation types can be displayed in either the open or the closed state. When closed, they appear on the page in some distinctive form depending on the specific annotation type, such as an icon, a box, or a rubber stamp. When the user activates the annotation by clicking it with the mouse, it exhibits its associated object, such as opening a pop-up window displaying a text note or playing a sound or a movie.

You can access and manipulate annotation data from both a plug-in and JavaScript. However, you can only create new annotation types using a plug-in or the IAC API; you cannot create new annotation types from JavaScript.

Annotations and JavaScript

You can set, modify, and access annotation information using JavaScript. However, you cannot create a new annotation type from JavaScript. To do this, you must use a plug-in or the IAC API.

Using JavaScript, you can perform the following tasks:

- Add note comments
- Make text edits
- Highlight, cross out, or underline text
- Add or delete custom stamps
- Add comments in a text box
- Add attachments
- Spell check in comments and forms
- Add commenting preferences
- Change colors, icons, and other comment properties

Annotations with plug-ins or IAC

There is an abstract superclass for all annotations. Acrobat and Adobe Reader have two built-in annotation classes. Plug-ins can add movie, widget (form field), and other annotation types. You can define new annotation subtypes by creating new annotation handlers. For more information, see the *Acrobat and PDF Library API Reference*.

The IAC API has an object you can use to set, modify, and access annotation information from external applications. For more information, see *Developing Applications Using Interapplication Communication*.

New annotation types

PDF supports many standard annotation types. Your plug-in can create annotation types, including any data they need. For example, a custom annotation type might allow a user to draw (not just type) in an annotation, provide support for multiple fonts or text styles, or support annotations that can only be viewed by specific users.

Support for new annotation types can be added by defining and registering an annotation handler. The Acrobat Movie plug-in, for example, uses this to support video annotations.

To add a new annotation type, a plug-in must provide a set of callbacks, specify them in the appropriate structure, and register them. For more information, see the *Acrobat and PDF Library API Reference*.

7 XML and the Acrobat SDK

The Adobe XML architecture offers enterprises a step-by-step migration from manual, paper-based workflows to streamlined, automated processes that accelerate the flow of business-critical information between employees, customers, and suppliers. This chapter summarizes the Acrobat XML support and features.

Adobe XML architecture

The Adobe XML architecture leverages the capabilities of XML and PDF to support a variety of business applications, while offering connectivity to systems through a variety of industry-standard and Adobe technologies.

XML forms model

The Adobe XML forms model uses a Document Object Model (DOM) architecture to manage the components that comprise a form. These include the base template, the form itself, and the data contained within the form fields. In addition, all calculations, validations, and formatting are specified and managed within the DOM and XML processes.

Both static and dynamic XML forms are supported by Acrobat. Both types are created using Adobe LiveCycle® Designer. A static form presents a fixed set of text, graphics, and field areas at all times. Dynamic XML forms are created by dividing a form into a series of subforms and repeating subforms. They support dynamically changing fields that can grow or shrink based on content, variable-size rows and tables, and intelligent data import/export features. For more information, see "About XML forms" on page 30.

XML templates

JavaScript defines an object that supports interactive form architectures. In this context, a template is a named page within a PDF document that provides a convenient format to automatically generate and manipulate a large number of form fields. These pages contain visibility settings, and can be used to spawn new pages containing identical sets of form controls to those defined within the template.

Extensible Metadata Platform (XMP)

Adobe's Extensible Metadata Platform (XMP) is a labeling technology that allows you to embed metadata into the file itself. With XMP, desktop applications and back-end publishing systems use a common method for capturing, sharing, and leveraging metadata. For more information, see "Metadata, Accessibility, and PDF Layers" on page 34.

SOAP and web services

Acrobat supports SOAP 1.1 and 1.2 to enable PDF forms to communicate with web services. This makes it possible to include both SOAP header and body information, send binary data more efficiently, use document/literal encoding, and facilitate authenticated or encrypted communications. It also provides the ability to locate network services using DNS Service Discovery (DNS-SD). All of this enables the integration of PDF files into existing workflows by binding Adobe XML forms to schemas, databases, and web services. These workflows include the ability to share comments remotely or invoke web services through form field actions.

If the exact URL for a given service is not known, but it is available locally because it has been published using DNS Service Discovery, Acrobat provides JavaScript methods to locate the service on the network and bind to it for communications.

A SOAP-based collaboration server can be used to share comments remotely using a web-based comment repository. Through the DNS Service Discovery support, it is possible to enable dynamic discovery of collaboration servers, initiation workflows, and RSS feeds that can provide customized user interfaces, using XML, directly inside of Acrobat.

In addition, it is possible to deploy web-based JavaScript code that always maintain the most updated information and processes, and to connect to those scripts using form field actions that invoke web services.

For more information, see *Developing Acrobat Applications Using JavaScript* and *Acrobat Online Collaboration: Setup and Administration*.

Conversion of PDF documents to XML format

Because XML representation is the basis for the exchange of information with web services and enterprise infrastructures, it is often useful to convert your PDF documents into XML format. It is a straightforward process to do this using JavaScript. For more information, see *Developing Acrobat Applications Using JavaScript*.

Alternatively, you can use the SaveAsXML plug-in, which extends the Save As Type choices in the Save As dialog box to allow a Tagged PDF document to be saved in a number of XML, HTML, or similar text-based formats. The plug-in uses mapping tables to control the conversion process for the SaveAsXML feature. For more information, see *Extending the Acrobat SaveAsXML Plug-in*.

XML-based information

JavaScript provides support for XML-based information generated within workflows. For more information, see *Developing Acrobat Applications Using JavaScript*.

8 Forms and the Acrobat SDK

You can use the Acrobat SDK to extend the functionality of forms created in Adobe LiveCycle Designer. Your form design can have dynamically changing features such as the current date, as well as convenience options such as automatic generation of e-mail messages. It can even have a dynamically changing appearance and layout that is responsive to user interactions.

This chapter presents basic concepts about forms and discusses how you can use the technologies of the Acrobat SDK to manipulate forms.

Creating a new form, whether using LiveCycle Designer or programmatically using JavaScript, requires you to consider many specific issues. For guidelines, see *Developing Acrobat Applications Using JavaScript*. For information about creating forms, see the LiveCycle Designer documentation.

Types of forms

Forms can be either XML forms or Acrobat forms.

XML forms — A dynamic XML-based form is derived from a form design that you create using LiveCycle Designer. The form design specifies a set of layout, presentation, and data capture rules, including calculating values based on user input. The rules are applied whenever a form is filled with data. How the form is filled with data depends on the type of form. Use XML forms, which are based on the XML Forms Architecture specification, for all new forms development.

Acrobat forms — This older type of form could be used to retrieve form data using JavaScript or using the Forms API with plug-ins and external applications. With Acrobat forms, you can author form fields and retrieve data from those form fields. For Adobe Reader, the Forms plug-in does not allow form authoring, but allows users to fill in data and print Acrobat forms. The Adobe Reader Forms plug-in also does not allow users to save data to the local hard disk. Both Acrobat and Adobe Reader allow web designers to send data from the form back to a web server. For new forms development, use XML forms instead.

Workflows for forms

There are three basic workflows for forms:

- Forms are filled in on the screen and then printed out. They are then returned by traditional methods such as fax or postal mail.
- Forms contain a Submit button that enables the sending of an email message with an attached data file that contains only the form data.
- Forms submit form data to a web server much like forms on the Internet. The user must be online to submit the information.

About XML forms

For XML forms, properties and methods available only from JavaScript allow you to access XFA-specific objects. You cannot access these objects from a plug-in or external application.

Form population — You can populate forms from a database or using a standards-based network protocol such as SOAP. XML forms are particularly well suited for populating forms from an external database. For more information, see "SOAP and web services" on page 28.

Web-ready forms — XML forms can be used in workflows that require the exchange of information over the web. You can create forms that run in web browsers, and can submit and retrieve information between the client and server by making a Submit button available in the form. The button can perform tasks similar to those of HTML scripting macros.

Data collection — You can save your form data in pure XML format, or, to take advantage of the rich functionality offered by the XFA plug-in, you can save your forms in the XML Data Package format (XDP). The XDP format allow you to package units of PDF content within a surrounding XML container, and is thus considered an XML-based companion to PDF. The advantage of this format is that XML parsers provide direct access to the XML form-data subassembly of the PDF document.

Using JavaScript, form data can be saved in either FDF or XFDF format in a separate file that can subsequently be used in the next step within a workflow. This approach minimizes the file size to just the amount needed to store your data, thus decreasing network traffic and processing time for the next step in the workflow.

Once you've collected PDF form data in FDF or XML format from multiple users, you can organize the form data into a comma-delimited spreadsheet file (CSV). After exporting the form data to a CSV file, you can work with the data in a spreadsheet application, such as Microsoft Excel. You can also save form data as a tab-delimited file. Tab-delimited files can be imported where required.

Data from various attachments can also be imported into an XML or XFA-based document, and submitted to a server for processing.

For more information on XML forms, see the *Acrobat JavaScript Scripting Guide* or the *JavaScript for Acrobat API Reference*.

About Acrobat forms

For Acrobat forms, a rich set of APIs allows you to create and manipulate form fields, and to retrieve form data using JavaScript, a plug-in, or an external application. Though you can manipulate form fields and form data from a plug-in, it is much quicker to develop Acrobat forms using JavaScript. Using the Acrobat SDK, you can perform the following tasks:

- Create Adobe PDF forms from scratch or from a template
- Add or remove form fields
- Set form field properties
- Make forms web-ready
- Extract and export form data
- Make forms accessible

You can extend the functionality of Acrobat forms with JavaScript. For example, you can use JavaScript to do the following tasks:

- Automate formatting, calculations, and data validation
- Develop customized actions assigned to user events
- Interact with databases and web services

Forms API

The Forms plug-in for Acrobat allows plug-in developers to author Acrobat form fields.However, it cannot be used to edit LiveCycle Designer XML forms.

For Adobe Reader, the Forms plug-in does not allow form authoring, but allows users to fill in data and print Acrobat forms. In general, the Adobe Reader Forms plug-in does not allow users to save data to the local hard disk. However, if the PDF document has additional usage rights, then it may be able to save the document or perform other functions. For more information, see "Rights-Enabled PDF Documents and Security" on page 32.

Both Acrobat and Adobe Reader allow web designers to send data from the form back to a web server.

OLE automation

The Acrobat Forms plug-in works as an automation server on Windows. There is no equivalent support on Mac OS. OLE automation is particularly useful for creating custom forms from an external application. Methods and properties are provided that allow you to programmatically associate actions and JavaScript with form fields. You can also add document-level JavaScript. For more information, see the *Acrobat and PDF Library API Reference*.

9 Rights-Enabled PDF Documents and Security

This chapter discusses rights-enabled PDF documents and security issues related to the Acrobat SDK.

About rights-enabled documents

When creating a PDF document, it is possible to assign it special rights that enable users of Adobe Reader to fill in forms, participate in online reviews, and attach files. LiveCycle Reader Extensions can be used to activate additional functionality within Adobe Reader for a particular document, thereby enabling the user to save, fill in, annotate, sign, certify, authenticate, and submit the document. This streamlines collaboration for document reviews and the automating of data capture with electronic forms. Users of Acrobat Professional can also Reader-enable collaboration. You cannot programmatically rights-enable a PDF file.

Creation of rights-enabled documents

During the design process, a PDF document can be created through the usage of LiveCycle Designer, Adobe LiveCycle Forms, or Adobe LiveCycle Assembler. The document creator can assign appropriate usage rights using LiveCycle Reader Extensions. When the PDF document is made available on the web, users can complete the form on the website, or save it locally and subsequently complete and annotate it, digitally sign it, add attachments, and return it.

In effect, LiveCycle Reader Extensions enables additional functionality within Adobe Reader for a given document. After the user has finished working with the document, those functions are disabled until the user receives another rights-enabled PDF file.

A major benefit of LiveCycle Reader Extensions is that it supports data automation through XML-based representation and transfer using SOAP, ensuring seamless integration with business logic and advanced data transport capabilities available within enterprise applications.

For more information, see *Developing Acrobat Applications Using JavaScript*.

Validation of usage rights

Certain editing features normally available within Acrobat Standard and Acrobat Professional are disabled for documents having Reader Extensions. This ensures that the user cannot inadvertently invalidate the additional usage rights in a document under managed review before passing the document on to an Adobe Reader user for further commenting. For more information, see *Developing Acrobat Applications Using JavaScript* and *Developing Plug-ins and Applications*.

Document security

You can use Acrobat security features to lock a PDF document. For example, you can use passwords to restrict users from opening, printing, or editing PDF documents. You can use digital signatures to certify PDF documents, and you can encrypt PDF documents so that only an approved list of users can open them. To save security settings for later use, you can create a security policy that stores security settings.

You can use the following to enhance document security using the Acrobat SDK:

Password security — Add passwords and set security options to restrict opening, editing, and printing PDF documents. Acrobat supports 128-bit RC4 and 128-bit AES (Advanced Encryption Standard) security methods. You can choose which method to use when securing documents.

Certification security — Encrypt a document so that only a specified set of users have access to it.

Adobe LiveCycle Policy Server — Apply server-based security policies to PDF documents. Server-based security policies are especially useful if you want others to have access to PDF documents only for a limited time.

Document certification — When an author digital signature is added, editing changes are restricted and detected.

For more information, see *Developing Plug-ins and Applications*, *Developing Acrobat Applications Using JavaScript*, and the Acrobat security site at http://www.adobe.com/go/acrobat_developer.

For information on how to view or modify security using Acrobat Standard or Acrobat Professional, see the Help for each product.

10 Metadata, Accessibility, and PDF Layers

This chapter discusses several topics related to usage of the Acrobat SDK: metadata, accessibility, and PDF layers.

Metadata

A PDF document can include general information such as the document's title, author, and creation and modification dates. Such global information about the document itself (as opposed to its content or structure) is called metadata, and is intended to assist in cataloguing and searching for documents in external databases.

Metadata properties and values are represented in the World Wide Web Consortium's Resource Definition Format (RDF), which is a standard metadata format based on XML. The set of standard metadata items is organized into schemas, each of which represents a set of properties from a particular application or industry standard. The schemas, as well as the physical representation, are defined by Adobe's Extensible Metadata Platform (XMP). For more information, see the *PDF Reference*.

Extensible Metadata Platform (XMP)

Adobe's Extensible Metadata Platform (XMP) is a labeling technology that allows you to embed metadata into the file itself. With XMP, desktop applications and back-end publishing systems gain a common method for capturing, sharing, and leveraging this valuable metadata — opening the door for more efficient job processing, workflow automation, and rights management, among many other possibilities.

You can use JavaScript to access the XMP metadata embedded in a PDF document, and you can search metadata by using JavaScript or a plug-in.

Plug-ins can also get or set XMP metadata. Furthermore, you can get or set the XMP metadata associated with an internal PDF dictionary or stream. For more information, see the *Acrobat and PDF Library API Reference*

Acrobat Distiller also supports embedding of XMP metadata in PDF files.

Adobe XMP Toolkit

The XMP Toolkit is designed to help applications handle XMP operations such as the creation and manipulation of metadata. The toolkit makes it easier for developers to support XMP metadata, and helps to standardize how the data is represented and interchanged. The XMP Toolkit can be licensed, royalty-free, from Adobe. For more information, see http://www.adobe.com/go/acrobat_developer.

Accessibility

Acrobat and the Acrobat SDK provide extensive support for accessibility of documents to visually-impaired users. To enable proper vocalization, either through a screen reader or by some more direct invocation of a text-to-speech engine, PDF supports the following features:

- Specifying the natural language used for text in a PDF document—for example, as English or Spanish
- Providing textual descriptions for images or other items that do not translate naturally into text, or replacement text for content that does translate into text but is represented in a nonstandard way (such as with a ligature or illuminated character)
- Specifying the expansion of abbreviations or acronyms

The core of this support lies in the ability to determine the logical order of content in a PDF document, independently of the content's appearance or layout, through logical structure and tagged PDF documents. Tagged PDF documents are created using the Acrobat Professional user interface or using the Acrobat SDK.

If a PDF document is untagged, you can convert it to a tagged PDF document using Acrobat Professional. Acrobat Standard provides only minimal support for tagging and no way to review or repair accessibility problems.

An accessible application can extract the content of a document for presentation to a user who is disabled by traversing the structure hierarchy and presenting the contents of each node. For this reason, producers of PDF files must ensure that all information in a document is reachable through the structure hierarchy.

For more information on accessibility support in PDF files, see the *PDF Reference*. For more information on implementing accessibility support, see the Acrobat Help and *Developing Acrobat Applications Using JavaScript*.

PDF layers

Adobe PDF layers are components of content that can occupy the same physical space as other components. Multiple components may be visible or invisible depending on their settings, and may be used to support the display, navigation, and printing of layered PDF content by various applications.

Using the Acrobat SDK, you can display, navigate and print layered PDF documents.

Acrobat supports the display, navigation, and printing of layered Adobe PDF content output by applications such as Adobe InDesign®, AutoCAD, and Microsoft Visio. PDF layers are supported through the usage of PDF Optional Content Group (OCG) objects. Optional content refers to content in a PDF document that can be selectively viewed or hidden.

Note the following:

- You can rename and merge layers, change the properties of layers, and add actions to layers. You can also lock layers to prevent them from being hidden.
- You can control the display of layers using the default and initial state settings. For example, if your document contains a copyright notice, you can easily hide the layer whenever the document is displayed onscreen but ensure that the layer always prints.

- Acrobat does not allow you to author layers that change visibility according to the zoom level, but it does support this capability from other authoring applications.
- To direct users to a custom view using a particular layer set, you can add bookmarks to a PDF document that contains layers. Use this technique to highlight a portion of a layer that is especially important. You can add links so that users can click on a visible or invisible link to navigate to or zoom in on a layer.
- To create layers while exporting InDesign CS or later documents to PDF, make sure that Compatibility is set to the latest Acrobat version and that Create Acrobat Layers is selected in the Export PDF dialog box.

Creation of layered PDF files

When creating PDFs, several engineering applications, including Microsoft Visio and AutoCad, automatically generate the necessary ProcSets to create layered PDF documents.

You create layered PDF documents with the ProcSet used to build Optional Content (OC) into PDF through Acrobat Distiller.

Third-party developers must insert OC ProcSet information into the PostScript stream. For more information, see "Creating PDF files from an authoring application" on page 16 and the *Acrobat Distiller API Reference*.

What you can do with layers

Since information can be stored in different layers of a PDF document, navigational controls can be customized within different layers, whose visibility settings may be dynamically customized so that they are tied to context and user interaction. For example, if the user selects a given option, a set of navigational links belonging to a corresponding optional content group can be shown.

Using JavaScript, you can determine the order in which layers are displayed in the user interface. You can also use JavaScript to perform the following tasks:

- Merge layers in a PDF document
- Flatten layers in a PDF document
- Combine PDF layered documents

For more information, see *Developing Acrobat Applications Using JavaScript*.

For plug-ins, you use an object to represent an optional-content group. This corresponds to a PDF 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. From a plug-in, you can perform the following tasks:

- Create an OCG
- Get and set the current state of an OCG
- Get and set the initial OCG state
- Get and set document configurations

For more information, see the *Acrobat and PDF Library API Reference*.

11 Searching and Indexing

With Acrobat and the Acrobat SDK, you can use the Search plug-in to locate an occurrence of a word in document content, metadata, attachments or other objects. You can use the Catalog plug-in to create a full-text index of a PDF document or document collection. This chapter summarizes these plug-ins.

Search plug-in

With Acrobat, Adobe provides a full-text search system. Using the Acrobat search system, you can perform the following tasks:

- Initiate a search with options. You can pass query expressions and search settings (for example, whole words only, word stemming, case sensitive, and so forth) to the Acrobat search plug-in and initiate the search. Search results will be presented to the user.
- Control the list of indexes to search. The search interface allows searches to be performed on one or more of the available indexes. You can control the list of active indexes.

The Acrobat Search plug-in allows users to perform text searches in PDF documents. It adds menus, menu items, toolbar buttons, and a Search panel to Acrobat or Adobe Reader.

You can communicate with the Search plug-in through its plug-in API, IAC (DDE or Apple events) or JavaScript. You can perform all search options from each of these development environments.

The Acrobat Search plug-in provided with Acrobat is a true Acrobat plug-in. You can remove it from the system by simply removing it from the Plug-ins directory and restarting Acrobat.

Indexes and the Catalog plug-in

You can use the Acrobat SDK to create a full-text index of a set of PDF documents. A full-text index is a searchable database of all the text in the documents. After building an index, you can use search the entire library quickly. You can build and manipulate indexes from a plug-in, through JavaScript, or from an external application using IAC (DDE or Apple events) calls.

For indexing PDF files, Acrobat provides text extraction APIs. Text extraction also supplies position information that can be used to highlight search hits in the original PDF file. The text extraction tools are provided as calls in the plug-in API on Windows and Mac OS. You can extract ASCII text from a PDF file using a plug-in or using JavaScript. You can also save the PDF document as text or rich text.

Acrobat Catalog is a plug-in that allows you to create a full-text index of a set of PDF documents. The Catalog plug-in has an HFT consisting of several methods that plug-in developers can import and use. In addition, Catalog supports DDE, and broadcasts several Windows messages.

For more information, see *Developing Plug-ins and Applications*, *Acrobat and PDF Library API Reference*, and *Developing Acrobat Applications Using JavaScript*.

Index