



Updates to the Portable Document Format Reference Manual

Adobe Developer Support

Technical Note #5156

08 December 1994

Adobe Systems Incorporated

Corporate Headquarters
1585 Charleston Road PO Box 7900
Mountain View, CA 94039-7900
(415) 961-4400 Main Number
(415) 961-4111 Developer Support
Fax: (415) 969-4138

Adobe Systems Europe B.V.
Europlaza
Hoogoorddreef 54a
1101 BE Amsterdam Z-O, Netherlands
+31-20-6511 355
Fax: +31-20-6511 313

Adobe Systems Eastern Region
24 New England
Executive Park
Burlington, MA 01803
(617) 273-2120
Fax: (617) 273-2336

Adobe Systems Japan
Swiss Bank House 7F
4-1-8 Toranomom, Minato-ku
Tokyo 105, Japan
+81-3-3437-8950
Fax: +81-3-3437-8968

Copyright © 1994 by Adobe Systems Incorporated. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the publisher. Any software referred to herein is furnished under license and may only be used or copied in accordance with the terms of such license.

PostScript is a 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.

Any references to a "PostScript printer," a "PostScript file," or a "PostScript driver" refer to printers, files, and driver programs (respectively) which are written in or support the PostScript language. The sentences in this book that use "PostScript language" as an adjective phrase are so constructed to reinforce that the name refers to the standard language definition as set forth by Adobe Systems Incorporated.

Adobe, the Adobe logo, Acrobat, Catalog, Distiller, and PostScript are trademarks of Adobe Systems Incorporated or its subsidiaries and may be registered in certain jurisdictions. Apple and Macintosh are registered trademarks and TrueType is a trademark of Apple Computer, Inc. Microsoft and MS-DOS are registered trademarks and Windows is a trademark of Microsoft Corporation. UNIX is a trademark registered in the United States and other countries and licensed exclusively to X/Open Company, Ltd. Other brand or product names are the trademarks or registered trademarks of their respective holders.

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.

Contents

Updates to the Portable Document Format Reference Manual 5

2 Overview (updated) 6

Portability 6

4 Objects (updated) 7

4.4 Strings 7

4.8 Streams 9

4.11 Object references 9

5 File Structure (updated) 9

5.1 Introduction 9

5.2 Header 10

5.5 Trailer 10

5.7 Encryption 11

6 Document Structure (updated) 11

6.2 Catalog 11

6.4 Page objects 12

6.6 Annotations 12

Link annotations 14

Destinations and actions 14

File specifications 18

Named destinations 19

6.7 Outline tree 20

6.8 Resources 20

Font resources 20

Font descriptors 21

Font files 21

Color space resources 22

Device color space resources 23

Device-independent color space resources 23

Indexed color space resources 25

Default color space resources 25

XObject resources 26

Image resources 26

Pass-through PostScript language resources 27

6.9 Info dictionary 28

6.10 Articles 29

6.11 File ID 31

6.12 Encrypt dictionary 32

7 Page description (updated) 33

7.2 Graphics state 33

Fill color space 33

Stroke color space 34

Rendering intent 34

7.4 Color operators 34

Text string operators 35

7.9 In-line image operators 35

7.11 In-line pass-through PostScript fragments 35

7.12 Compatibility operators 36

Appendix F PDF name registry 36

Appendix G Compatibility 37

G.1 Version numbers 38

G.2 Viewer compatibility behavior 39

Dictionary keys 40

Annotations 40

Destinations and actions 40

XObjects 41

Color spaces 41

Filters 42

Page description operators 43

Procedure sets 43

Copyrights and permissions to use PDF 43

Bibliography 44

Errata for the Portable Document Format Reference Manual 45

Updates to the Portable Document Format Reference Manual

This document is an addendum to the *Portable Document Format Reference Manual*. It describes version 1.1 of the Portable Document Format. It does not specify the format in its entirety but only the changes from version 1.0 as described in the original specification. PDF 1.1 is the native file format of the Adobe[™] Acrobat[™] 2.0 family of products.

The PDF specification is defined independently of any particular implementation of a PDF generator or consumer. To provide guidance to implementors, however, the behavior of Acrobat viewers (both 1.0 and 2.0) when they encounter the changes documented herein is described in *Implementation Notes* that accompany the specification and in Appendix G, “Compatibility.”

The PDF 1.1 specification, like the PDF 1.0 specification, defines a minimum interchange level of functionality. The Portable Document Format is an extensible format which means that PDF files may contain objects not defined by this specification. Consumers, applications that read PDF files and interpret their contents, are expected to correctly implement the semantics of objects that are specified by PDF 1.1 and, as gracefully as possible, to ignore any objects not understood by the consumer. Appendix G, “Compatibility,” provides guidance on how a consumer should handle objects it does not understand

Implementation note

Some Acrobat 2.0 products provide an interface that allows plug-in extensions. These extensions can use and/or put private data objects within a PDF file. Appendix G, “Compatibility,” indicates the kinds of private data that can be used and Appendix F, “PDF name registry,” defines a registry for this data. The registry can be used to avoid conflicts in indentifying data from independent extensions.

The new features introduced in PDF 1.1 and Acrobat 2.0 include the following:

- the ability to protect documents with a password and to restrict operations on documents,

- the ability to tie blocks of text together into “articles,” making reading easier,
- the generalization of link and bookmark destinations to “actions,” which include links to other PDF files and foreign files,
- the ability to define new annotation types and to provide additional attributes for existing types,
- the ability to specify default settings and actions when a document is opened,
- device-independent color,
- an ID included in all files to make it easier to find a file on a network, given a name,
- a binary option that allows files to be smaller
- a new date format that allows programmatic comparison of dates,
- the ability to provide additional document information.

This document is organized in the same way as the original PDF specification. Changes are described as additions to the appropriate sections of that specification. Section 2 contains updates to Chapter 2 of the *PDF Reference Manual*, including provision for binary PDF files. Section 4 contains updates to Chapter 4 of the *PDF Reference Manual*, including the date format, and changes and clarifications to the representation of streams. Section 5 contains updates to Chapter 5 of the *PDF Reference Manual*, including a new version number, and new trailer keys for encryption and file identification. Section 5 contains updates to Chapter 6 of the *PDF Reference Manual*, including article threads, named destinations, annotation actions, device independent color resources and the Encrypt dictionary. Section 7 contains updates to Chapter 7 of the *PDF Reference Manual*, including operators to use device independent color spaces. Finally, the appendices describe the PDF name registry, the behavior of Acrobat viewers when presented with PDF 1.1 or greater files, and references to other documents.

Note In PDF 1.1, dictionary key names are often one or two letters in order to conserve space in files. When these keys are described below, they are followed in parentheses by their full name. However, only the actual one- or two-letter name may be used in a PDF file.

2 Overview (updated)

This section includes the changes required to update Chapter 2 of the *PDF Reference Manual* to version 1.1.

2.3.2 Portability

Replace the text in this section with the following text:

A PDF file is either a 7-bit ASCII file or a binary file. If it is a 7-bit ASCII file only the printable subset of the 7-bit ASCII code plus space, tab and newline (return or linefeed) is used. If it is a binary file, the entire 8-bit range of characters may be used.

ASCII is the most portable form, since it is the only form that will fit through channels that are not 8-bit clean or are subject to end-of-line translation, etc. A binary file simply cannot be transported in such cases.

Unfortunately, some agents, when presented with information labelled as "text", take unreasonable liberties with the contents. For example mail transmission systems may not preserve certain 7-bit characters and may change line endings. This can cause damage to PDF files.

Therefore, in situations where it is possible to label PDF files as "binary", we recommend that this be done. One method for encouraging such treatment is to include some binary characters (codes greater than 127) in a comment near the beginning of the file, as described in section 5.1, even if the rest of the file is ASCII.

This ensures that a PDF file will be treated as binary when this is possible, while still allowing it to be transferred through a non-binary channel without damage.

Use of PDF files actually containing binary information should be restricted to closed environments which are known to transport and store binary files safely or where some external means is used to convert the file into and out of a transport independent form, such as the UNIX uuencode.

Implementation note

The Acrobat 1.0 viewer for UNIX will directly read uuencoded PDF files.

4 Objects (updated)

This section includes the changes required to update Chapter 4 of the *PDF Reference Manual* to version 1.1. These changes include adding a new date format for strings and some changes to the way streams are represented.

4.4 Strings

Add the following text to the end of the section:

In versions 1.1 and later, it is not necessary to represent strings using only the printable 7-bit ASCII character set. PDF 1.1 strings may contain non-printable ASCII codes—in fact, any 8-bit value. In particular, when a document is encrypted (see section 5 on page 9), all its strings are encrypted. In PDF 1.0, PDF strings are mostly or entirely printable ASCII. Encrypted strings are not.

Implementation note

The Acrobat 1.0 viewers can read strings which include non-printable ASCII.

Strings can be used for many purposes and can be formatted in different ways. When a string is used for a specific purpose, to represent a date, for example, it is useful to have a standard format for that purpose. This would let a user find all documents with a creation date later than a certain date.

Such formats are conventions for interpreting strings and are not types themselves. The use of a particular format is indicated with the definition of the string object that uses the format

PDF 1.1 defines a standard date format. The PDF date format closely follows the format defined by the international standard ASN.1 (Abstract Syntax Notation One, defined in CCITT X.208 or ISO/IEC 8824). A date is a string of the form

(D:YYYYMMDDHHmmSSOHH'mm')

where

- *YYYY* is the year
- *MM* is the month (01–12)
- *DD* is the day (01–31)
- *HH* is the hour (00–23)
- *mm* are the minutes (00–59)
- *SS* are the seconds (00–59)
- *O* is the relation of local time to GMT where + indicates that local time is later than GMT, – indicates that local time is earlier than GMT, and Z indicates that local time is GMT
- *HH'* is the absolute value of the offset from GMT in hours
- *MM'* is the absolute value of the offset from GMT in minutes

The “D:” prefix permits arbitrary keys to be recognized as dates. However, it is not required. Trailing fields other than the year are also optional. The default value for day is 1; all other numerical fields default to 0. If no GMT information is specified, the relation of the specified time to GMT is considered unknown. Whether the time zone is known or not, the rest of the date should be specified in local time.

Implementation note

*The Acrobat 1.0 viewers report date strings as ordinary strings. The Acrobat 2.0 viewers report date strings as dates when used as the value of the **CreationDate** or **ModDate** in the Info dictionary or as the value of the Date key in annotations. The 2.0 viewers ignore the GMT information.*

4.8 Streams

Add the following clarifying text after the syntax for a <stream> and before the reference to Table 4.2:

PDF 1.1 is more restrictive than PDF 1.0 with respect to the specification of stream objects. All streams must be indirect objects (see Section 4.10 of the *PDF Reference Manual*). The stream dictionary must be a direct object. The keyword **stream** that follows the stream dictionary should be followed by a carriage return and linefeed or just a linefeed.

Implementation note

Otherwise, it is not possible to differentiate a stream that uses carriage return as end of line and whose first byte of data is a linefeed from a stream that uses carriage return-linefeed pairs as end of line.

4.11 Object references

Add the following text at the end of this section:

PDF 1.1 defines links to external files but does not define directly reference objects in other PDF files. It is planned that a future version of PDF will define *foreign references*. In PDF 1.1 only a format for such references is reserved. A foreign reference is an indirect reference to an indirect object in another file, and consists of the foreign *file number*, the indirect object's object number, its generation number and the **F** keyword:

```
<foreign reference> ::=  
    <file number>  
    <object number>  
    generation number  
    F
```

A file number is a non-negative integer, but PDF 1.1 does not define its interpretation. To be compatible with future versions of PDF, PDF 1.1 consumers should treat all foreign references as null objects.

5 File Structure (updated)

This section includes the changes required to update Chapter 5 of the *PDF Reference Manual* to version 1.1. These include relaxing the requirement for 7-bit ASCII, adding a new option for the version number in the header, and adding file identifier and encryption keys in the trailer dictionary.

5.1 Introduction

Add the following text at the end of this section:

Because the requirement to use 7-bit ASCII does not guarantee file transmission transparency and because requiring use of 7-bit ASCII can cause a 20% expansion in the size of objects such as images that are naturally binary data, PDF 1.1 relaxes the requirement for 7-bit ASCII.

PDF 1.1 allows files to contain binary data in strings and streams. In fact, experiments have found that PDF files are less likely to be corrupted by system utilities if they do contain binary data. It is, therefore, recommended that the second line of a PDF file be a comment that contains at least four binary characters.

To accommodate binary data the restriction on line lengths is also relaxed in PDF 1.1. PDF 1.1 files with binary data may have arbitrarily long lines. However, to increase compatibility with other applications that process PDF files, all lines that are not part of stream object data shall be no longer than 255 characters.

Implementation note

The Acrobat 1.0 viewers successfully read files that contain binary data. In addition, the PDF 1.0 specification states that lines in a PDF file may not be longer than 255 characters. This restriction is not enforced by any Acrobat viewer.

Implementation note

The Acrobat 1.0 products on the Apple® Macintosh® computer create files with type 'TEXT'. Since PDF 1.1 may not necessarily be all ASCII, Acrobat 2.0 products create files with type 'PDF '. A user can open these documents from a 1.0 viewer but not from the Finder.

5.2

Header

Replace the sentence that begins “The current version is 1.0, ...” with

The current version is 1.1; the first line of a 1.1-conforming PDF file should be **%PDF-1.1**. However, a 1.0-conforming file is also a 1.1-conforming file and may begin with either **%PDF-1.1** or **%PDF-1.0**.

5.5

Trailer

Add the following dictionary keys to Table 5.1:

Table 5.1 *Trailer attributes (extended)*

Key	Type	Semantics
ID	array	<i>(Optional)</i> An array of two strings, each of which is an ID. The first ID is established when the file is created and the second ID is changed each time the file is updated. IDs are describe in Section 6.11, “File ID.”
Encrypt	dictionary	<i>(Required if document is encrypted)</i> Information used to decrypt a document, described in Section 6.12, “Encrypt dictionary.”

5.7 Encryption

Add this new section to Chapter 5:

PDF 1.1 allows documents to be encrypted to protect their content from unauthorized access. Access to the document content is controlled by the security handler identified by the Encrypt dictionary. The encryption dictionary is the value of the new **Encrypt** key in the trailer dictionary and is described in Section 6.12, “Encrypt dictionary.” The name of this handler is specified by the Encrypt dictionary’s **Filter** key

The standard security handler permits users to protect documents with a password and to limit operations on documents, for example, preventing documents from being printed or modified. All strings and streams, except those in the Encrypt dictionary, are encrypted using the RC4 encryption algorithm. This prevents unauthorized users from simply removing the password or access rights information from a PDF file to gain full access. The security handler must encrypt strings in the Encrypt dictionary itself.

Implementation note

While the Acrobat 2.0 viewers provide standard user authorization, a plug-in may provide an alternative method for authorizing users. On opening a protected document, a 1.0 viewer will report that an error was found while processing a page. An Acrobat 2.0 viewer will report that a plug-in is required to open the document if the handler for the document is not available.

6 Document Structure (updated)

This section includes the changes required to update Chapter 6 of the *PDF Reference Manual* to version 1.1. These changes include article threads, named destinations, link actions, device independent color resources, and the Encrypt dictionary.

6.2 Catalog

Add the following to the list of allowed values of the PageMode key:

FullScreen —Open document in full-screen mode; in full-screen mode, there is no menu bar, window controls, nor any other window present.

Add the following keys to the Catalog dictionary Table 6.1:

Table 6.1 Catalog attributes (extended)

Key	Type	Semantics
OpenAction	array or dictionary	(Optional) Any legal action as described in Section 6.6.3, “Destinations and actions.” If the value of this key is an array, it must be a destination. If it is a dictionary, it must be an action. If no action is specified, the top of the first page will appear at default zoom.
Threads	array	(Optional) An array of thread dictionaries as described in Section 6.10, “Articles.”
Dests	dictionary	(Optional) A dictionary in which the keys are the names of destinations and the values are destinations as specified in Section 6.6.5, “Named destinations.”

Implementation note If **FullScreen** is specified as the **PageMode** to an Acrobat 1.0 viewer, it is ignored. **OpenAction**, **Threads** and **Dests** are also ignored.

6.4 Page objects

Add the following keys to Table 6.3:

Table 6.3 Page attributes (extended)

Key	Type	Semantics
B (Beads)	array	(Recommended if page contains article beads) An array whose elements are composed of indirect references to each article bead on the page, in drawing order (the same order as the Annots array). Articles are described in Section 6.10, “Articles.”

Implementation note The Acrobat 2.0 viewers will rebuild the *Beads* array for all pages of a document containing beads if the first page with a bead does not have a *Beads* array.

6.6 Annotations

Immediately before Section 6.6.1 add:

PDF 1.1 supports several new annotation properties for text annotations and links. These keys apply to Link and Text annotations, as noted in Table 6.4a. Other annotation types may choose to honor these keys as appropriate.

Table 6.4a *Annotation attributes (extended)*

<i>Key</i>	<i>Type</i>	<i>Semantics</i>
C (Color)	array	(<i>Optional</i>) The annotation color. For links, this is the border color. For text annotations, it is the background color of a closed annotation's icon, the title bar color of an active open annotation's window, and the window frame color of an inactive open annotation. A color is specified as an array of three numbers between 0 and 1, inclusive, representing a color in DeviceRGB space.
T (Title)	string	(<i>Optional</i>) An arbitrary text label associated with the annotation. It is displayed in an active open text annotation's title bar and can be edited from the annotation's properties dialog.
M (ModDate)	string	(<i>Optional</i>) The last time an annotation was modified A text annotation's modification date is updated each time the text is changed. The preferred string value is the date format described in Section 4.4, "Strings" but viewers should accept and display any string.
F (Flags)	integer	(<i>Optional</i>) Collection of flags defining various characteristics of the annotation. The least significant is the invisible flag. If this flag's value is 1, and the viewer does not provide a handler for the annotation's Subtype, the annotation will not be displayed. Otherwise, the annotation will appear as an unknown annotation. (See below.) All other bits are reserved and must be set to 0. The default value for this key is 0.

Implementation note *The Acrobat 2.0 viewers only update the ModDate string for text annotations.*

The **Border** key, as in PDF 1.0, describes the border of link annotations. In PDF 1.0, the border is an array of three numbers. In PDF 1.1, a border is an array of three or four elements. The first two elements should be zero. The third element specifies the border width in points (as in PDF 1.0). The optional fourth element is a dash array that allows specification of solid and dashed borders. The dash array contains on and off stroke lengths for drawing dashes in the same format as the setdash marking operator, d. An example of a dashed border is [0 0 1 [3]].

Implementation note *Although PDF 1.1 does not limit the dash array length, Acrobat 2.0 viewers support a maximum of 10 entries*

Implementation note *The PDF 1.0 Reference Manual states that the first two numbers describe the horizontal and vertical corner radii, but 1.0 viewers ignore this information. This information will also be ignored by 2.0 and future viewers.*

Implementation note *If an Acrobat 2.0 viewer encounters an annotation of a type it does not understand, the viewer will display it as an unknown annotation unless the annotation's **F** (Flags) keys specifies that the invisible flag is set. The **C**, **T**, **M**, and **F** keys are ignored by Acrobat 1.0 viewers.*

6.6.2 Link annotations

Add the following key to Table 6.5:

Table 6.5 *Link annotation attributes (extended)*

Key	Type	Semantics
A (Action)	dictionary	<i>(Required unless Dest key is present)</i> The action to be performed on activating this link annotation; see Section 6.6.3, “Destinations and actions.”

On the Dest key, change “(Required)” to “(Required unless the A key is present).” Following the Table add:

Goto actions should be specified using the **Dest** key for compatibility with viewers implementing the PDF 1.0 specification.

6.6.3 Destinations and actions

Retitle Section 6.6.3 as “Destinations and actions.” After the first paragraph add:

The notion of a destination is expanded in PDF 1.1. In addition to specifying a destination it is possible to specify an action to be performed when a link or bookmark entry is activated or a document is opened. PDF 1.1 supports four types of actions:

- **GoTo**. — Changes the current page view to a specified page and zoom factor. This is based on the PDF 1.0 notion of destination.
- **GoToR** — Opens a remote PDF file at a specified page and zoom factor.
- **Launch** — Launch an application, usually to open a file.
- **Thread** — Begins reading an article thread, possibly in another PDF file. Section 6.10, “Articles” further describes article threads.

The latter three actions are new in PDF 1.1. In the future, PDF may define additional action types.

Implementation note *It is intended that plug-in extension may add new actions and described in Appendix G, “Compatibility.”*

An action is represented as a dictionary. All actions must contain a **Subtype** key. Other keys may be present, depending on the action type. The tables below list the attributes of the four specified action types.

Table 6.6a *GoTo action attributes*

<i>Key</i>	<i>Type</i>	<i>Semantics</i>
Type	name	(<i>Optional</i>) Object type. Always Action .
S (Subtype)	name	(<i>Required</i>) Action type. Always GoTo .
D (Dest)	array or name	(<i>Required</i>) The view destination, as described in Table 6.6. The destination page may be specified by an indirect reference to the page object or by a page number (the first page in a document is page 0). Destinations with page numbers are ignored by 1.0 viewers. Destinations that are names are described in Section 6.11, “Named destinations.”

As an accelerator for the GoTo action type, it is possible to omit the action dictionary and specify only a destination array as described in section 6.6.3 of the *PDF Reference Manual*. This is the preferred form for this action type. It is more compact and (most importantly) is the only form specified in PDF 1.0, and thus can be acted on by 1.0 viewers.

Add the following keys to Table 6.6:

Table 6.6 *Annotation destination specification (extended)*

<i>Value of /Dest key</i>	<i>Semantics</i>
[page /FitB]	Fit the bounding box of all the objects on the page and within the crop box to the window.
[page /FitBH top]	Fit the width of the bounding box of all the objects on the page and within the crop box to the window. <i>top</i> specifies the y-coordinate of the top edge of the window.
[page /FitBV left]	Fit the height of the bounding box of all the objects on the page and within the crop box to the window. <i>left</i> specifies the x-coordinate of the left edge of the window.

Add the following text after Table 6.6:

The GoToR action behaves similarly to the GoTo action. However, it includes an additional parameter, the **File** key that specifies the PDF file that contains the action’s destination.

Table 6.6b *GoToR action attributes*

<i>Key</i>	<i>Type</i>	<i>Semantics</i>
Type	name	(<i>Optional</i>) Object type. Always Action .

Table 6.6b GoToR action attributes (Continued)

Key	Type	Semantics
S (Subtype)	name	(Required) Action type. Always GoToR .
D (Dest)	array or name	(Required) The desired view destination, as described in Table 6.6. The destination page must be specified by a page number, with 0 the first page. Destinations that are names are described in Section 6.6.5, “Named destinations.”
F (File)	string or dictionary	(Required) The file containing the destination view. See Section 6.6.4, “File specifications,” for the interpretation of the File key.

The Launch action specifies an application to launch or document to open. The action must specify the application or document as a file, using the **F** key.

The PDF 1.1 specification also allows platform-specific information to be included in the Launch dictionary where that information is needed for specific platform. The key **Win** is used for information related to Microsoft Windows launches; the key **Unix** is used for information related to UNIX system launches. If there is no platform specific key, then the **F** key is used

Implementation note Some implementations of Acrobat 2.0 viewers may check for alternative keys whose values provide platform-specific parameters for the Launch action. For example, the Acrobat 2.0 viewer for Windows will use the dictionary corresponding to the **Win** key to determine its launch parameters.

Table 6.6c Launch action attributes

Key	Type	Semantics
Type	name	(Optional) Object type. Always Action .
S (Subtype)	name	(Required) Action type. Always Launch .
F (File)	string or dictionary	(Required if there is no alternative key) The file to use in performing the specified action. See Section 6.6.4, “File specifications” for the interpretation of the F key. A viewer that encounters an action with no F key and for which it does not understand any of the alternative keys will do nothing.
Win	dictionary	(Optional) Windows-specific launch parameters as described in Table 6.6d.
Unix	string	(Optional) Not yet defined.

Implementation note *The Acrobat 2.0 viewers for Windows use the Windows function ShellExecute() to launch an application. The **Win** dictionary entries correspond to the parameters of ShellExecute().*

Table 6.6d *Windows-specific launch attributes*

<i>Key</i>	<i>Type</i>	<i>Semantics</i>
F (File)	string	<i>(Required)</i> The document or application to launch, specified as a DOS file name using standard DOS syntax. If the string includes a backslash ('\'), the backslash must be preceded by a backslash according to the usual rules for specifying PDF strings.
O (Operation)	string	<i>(Optional)</i> The operation to perform: “open” or “print.” Open is the default. If the F key specifies an application, this key is ignored and the application is launched.
P (Parameters)	string	<i>(Optional)</i> The parameters passed to the application specified by the F key. If the F key specifies a document, this key should not be provided.
D (Directory)	string	<i>(Optional)</i> The default directory, specified using standard DOS syntax.

When a viewer performs a Thread action, it goes to the specified thread and enters thread mode. The thread need not be in the current PDF file.

Table 6.6e *Thread action attributes*

<i>Key</i>	<i>Type</i>	<i>Semantics</i>
Type	name	<i>(Optional)</i> Object type. Always Action .
S (Subtype)	name	<i>(Required)</i> Action type. Always Thread .
D (Dest)	dictionary, number, or string	<i>(Required)</i> The desired thread destination. The destination may be an indirect reference to a thread dictionary, a number that specifies the thread’s index in the document (the index of the first thread in a document is 0), or a string that is the thread’s title. If more than one thread has the same title, the first thread in the document’s list of threads with that title will be chosen. Threads in external files must be specified by index or title.
F (File)	string or dictionary	<i>(Optional)</i> The file containing the destination thread. See Section 6.6.4, “File specifications” for the interpretation of the F key.
B (Bead)	dictionary or number	<i>(Optional)</i> The desired bead destination. The value of B may be an indirect reference to a bead dictionary or a number that specifies the bead’s index in the thread (the index of the first bead in a thread is 0).

Implementation note *Acrobat 1.0 viewers do not report an error when a user activates a link or bookmark that has an unknown destination type or is missing a destination. Links and bookmarks with an **Action** key will appear to have no destination. The Acrobat 2.0 viewers will report an error when the destination or action type is unknown.*

6.6.4 File specifications

Add this new subsection to Section 6.6:

A file specification describes the location of a file. It is usually a string. Components of the specification should be separated by slash characters (the Unix file separator). When a viewer processes a file specification, it will replace separators with the appropriate system-dependent separator. Slashes that are part of a component should be preceded by a backslash. (In a PDF file, the backslash itself needs to be escaped by preceding it with another backslash.)

If a file specification should be considered relative to the current file, the first part of the specification should be a component. If the specification is absolute, the first part should be a slash. It is possible to specify an absolute DOS path without a drive by making the first component empty. (Empty components are ignored by other platforms.)

A DOS path can include a logical drive or a network resource name as returned by the Windows function WNetGetConnection(). A resource name includes both a drive and volume. The drive and volume, like all path components, should be preceded by a slash.

The component “.” refers to the parent of the current directory, where current is defined as the directory specified by the part of the file specification preceding the “.”. Table 6.6f provides examples of file specifications on various platforms.

Table 6.6f *Examples of file specifications*

<i>System</i>	<i>System-dependent path</i>	<i>PDF path</i>
Mac	Macintosh HD:PDFDocs:spec.pdf	/Macintosh HD/PDFDocs/spec.pdf
DOS	\pdfdocs\spec.pdf (no drive)	//pdfdocs/spec.pdf
DOS	r:\pdfdocs\spec.pdf	/r/pdfdocs/spec.pdf
DOS	pcadobe/eng:\pdfdocs\spec.pdf	/pcadobe/eng/pdfdocs/spec.pdf
Unix	/user/fred/pdfdocs/spec.pdf	/user/fred/pdfdocs/spec.pdf
Unix	pdfdocs/spec.pdf (relative)	pdfdocs/spec.pdf

A file specification can also be a dictionary. This dictionary contains one or more keys that specify alternative ways of locating a file. The key specifies how to interpret the value. A viewer should look for keys in some order until it finds one it understands. The alternative keys need not specify the same file. This allows a single file specification to describe appropriate but different files for different platforms.

Implementation note For example, the Acrobat 2.0 viewer for Macintosh first looks for the **Mac** key and if it does not find it, looks for **F** (File). Note that the Acrobat 2.0 viewer for Windows looks for the **DOS** key, not the **Win** key.

Table 6.6g describes the dictionary attributes.

Table 6.6g File specification attributes

Key	Type	Semantics
FS (FileSystem)	name	(Optional) The name of the “file system” to be used to interpret this file specification. A viewer or extension can register a file system. A file system interprets file specifications, opens files, and provides the usual input and output operations. If a file specification includes a file system, all other keys are interpreted by the file system.
F (File)	string	(Required if no other keys are present) A file specification using the string format described above. A viewer that encounters an action with no F key and that does not understand any of the alternative keys need not do anything.
Mac	string	(Optional) A string that specifies a Macintosh file name using the string format describe above.
DOS	string	(Optional) A string that specifies a DOS file name using the string format describe above.
Unix	string	(Optional) A string that specifies a Unix file name using the string format describe above.
ID	array	(Optional) An array of two strings. The ID is a file ID as described in Section 6.12, “File ID.” This allows a viewer to find the exact match more often, and it allows viewers to warn a user when the file has changed since the link was made.

6.6.5 Named destinations

Add this new subsection to Section 6.6:

The destination key of an action may be a name as well as an array. This allows destinations to be specified indirectly even if the destination is in another file. For example, one file may contain links to the first page of chapters in another file. If the links use named destinations, then the second file’s pages can change without invalidating the links in the source file. Named destinations can also be used for thread actions and for destinations within a single file.

The Catalog of any document may contain a **Dests** key whose value is a dictionary. The keys in this dictionary are destination names, and the values are destinations. The destinations may either be arrays or dictionaries. If the value is an array, it must be an ordinary destination. If it is a dictionary, it in turn must have a **D** key that must be an array that is an ordinary destination. This enables named destinations to have additional attributes.

If an action does not contains a file specification, then a named destination refers to a name in the current document's Dests dictionary. If an action contains a file specification, then the named destination refers to a name in the document specified by the file specification.

6.7 Outline tree

Add the following key to Table 6.8:

Table 6.8 Outline entry attributes (extended)

Key	Type	Semantics
A (Action)	dictionary	<i>(Required unless Dest key is present)</i> The action to be performed on activating this link annotation; see Section Destinations and actions..

On the *Dest* key, change “(Required)” to “(Required unless the *A* key is present)” and following the Table add:

Goto actions should be specified using the **Dest** key for compatibility with veiwers implementing the PDF 1.0 specification.

6.8 Resources

PDF 1.1 adds no new resource types but enhances font resources and font descriptors and adds new subtypes of ColorSpace and XObject resources.

6.8.2 Font resources

Add the following at the end of the section labelled “Type 1 fonts”:

PDF 1.1 permits documents to include subsets of Type 1 fonts. The font resource and font descriptor that describe a font subset are slightly different than those for ordinary fonts. These changes allow a viewer to merge documents containing subsets of the same font.

The value of the font resource's **BaseFont** key and the font descriptor's **FontName** key must use the following format:

pseudo-unique-tag+PostScript-name

The pseudo-unique tag must consist of exactly six uppercase alphabetic characters. The PostScript language name must be the name of the complete Type 1 font. A plus sign separates the tag and the PostScript language name.

Implementation note

These restrictions make font subsets compatible with 1.0 viewers, enable the Distiller™ application to recognize font subsets in its input stream, and enable Acrobat 2.0 viewers to merge documents containing subsets.

6.8.4 Font descriptors

Add the following keys Table 6.17 (to support font subsets and embedded TrueType fonts):

Table 6.17 Font descriptor attributes (extended)

Key	Type	Semantics
CharSet	string	(Optional) A string which lists the glyph names corresponding to the entries in the CharStrings dictionary if the font described is a subset font. Each name must be preceded by a slash. The names may appear in any order. The name “.notdef” should be omitted; it is assumed to exist in the font subset.
FontFile2	stream	(Optional) A stream which defines a TrueType font.

In addition change the semantics of **FontFile** to be “A stream which defines a Type 1 font.

Font files

Replace the first two sentences of the second paragraph, “Currently, only Type 1 fonts may be embedded in a PDF file. The are embedded using the FontFile mechanism,” with:

Type 1 fonts may be embedded in a PDF 1.1 file using the FontFile mechanism.

Delete the second to the last sentence of the second paragraph, “Because the stream containing Type 1 font data contains binary data, the stream **must** be converted to ASCII using either ASCII hexadecimal or ASCII base-85 encoding.”

Add a new paragraph following the second paragraph:

TrueType™ fonts are embedded using the FontFile2 mechanism. The font descriptor for an embedded TrueType font should contain a **FontFile2** key whose value is a stream that contains the TrueType font definition as described in *TrueType 1.0 Font Files*. The stream itself should include a **Length1** key as specified in Table 6.18; that key specifies the length in bytes of the font file after it has been decoded using the filters specified by the stream’s **Filter** key. The other keys in Table 6.18 should not to be used for TrueType fonts.

Because the stream containing Type 1 or TrueType font data may include binary data, the stream it may be desirable convert this data to ASCII using either the ASCII hexadecimal or ASCII base-85 encoding.

Implementation note

Embedded TrueType fonts are ignored by Acrobat 1.0 viewers.

6.8.5 Color space resources

*PDF 1.1 extends the PDF color model to include device-independent color for both images and spot color (text and paths). There are three new color spaces: CalGray, CalRGB, and Lab. These correspond to calibrated gray, calibrated RGB, and CIE 1976 (L*a*b*)-space. The following changes should be made to this section:*

*Replace the third sentence of paragraph 1, “PDF supports four types of color spaces: **DeviceGray**, **DeviceRGB**, **DeviceCMYK** and indexed” with the following sentence:*

PDF supports seven types of color spaces: **DeviceGray**, **DeviceRGB**, **DeviceCMYK**, **CalGray**, **CalRGB**, **Lab** and indexed. In addition, provisions have been made for **CalCMYK**, although the attributes of this type of space have not yet been defined.

Replace the first sentence of paragraph 2, “All the color spaces support by PDF are device color spaces” with:

PDF 1.1 supports both device and device-independent color spaces. In a device color space, the color values are interpreted as specifying the fraction of device color intensity achievable on the device being used for imaging.

Add the following after paragraph 2:

In a device-independent color space, color values are defined by a mapping from the device-independent color space into a standard color space, the CIE (Commission Internationale de l’Eclairage) 1931 XYZ color space. Since the values in the XYZ space can be colorimetrically measured, this establishes a device-independent specification of the desired color. When a device-independent color value is rendered on a device, the rendered color should have the same colorimetric value as the specified color. It is the responsibility of the device rendering system to achieve the colorimetric match.

Physical devices have a limited range of colors that they can reproduce. This is called their *gamut*. Therefore, some trade-offs may need to be made in rendering the desired color. See the discussion of “intents” in the “Image resources” subsection of Section 6.8.6, “XObject resources.”

See the *PostScript Language Reference Manual, Second Edition* for further explanation of device-independent color.

Implementation note

The Acrobat 2.0 viewers allow a user to approximate device-independent colors with device-dependent colors with no transformation. I.e., CalGray colors are treated as DeviceGray, and CalRGB are treated as DeviceRGB.

Immediately following the above addition add the following heading:

Device color space resources

Immediately following the paragraph describing **DeviceCMYK**, add the following heading and text:

Device-independent color space resources

Colors in the **CalGray** color space are represented by a single value. Input **CalGray** values are in the range [0, 1] where 0 is black, 1 is white and intermediate values are gray.

A **CalGray** color space is specified by a two component array of the form

[/CalGray dict]

where the contents of dict are described in Table 6.19a.

Table 6.19a CalGray attributes

Key	Type	Semantics
WhitePoint	array	(Required) Three numbers [Xw Yw Zw] that specify the CIE 1931 (XYZ)-space tristimulus value of the diffuse white point. The numbers Xw and Zw must be positive and Yw must be equal to 1. See discussion in 4.8.3 in the <i>PostScript Language Reference Manual, Second Edition</i> for further details.
BlackPoint	array	(Optional) Three numbers [Xb Yb Zb] that specify the CIE 1931 (XYZ)-space tristimulus value of the diffuse black point. The numbers must be non-negative. Default value: [0 0 0]. See discussion in 4.8.3 in the <i>PostScript Language Reference Manual, Second Edition</i> for further details.
Gamma	number	(Optional) Defines the exponential relationship between the gray component and Y. The governing equation is $Y = \text{gray}^{\text{Gamma}}$. Gamma must be positive and will generally be greater than or equal to 1. Default value: 1.

Colors in the **CalRGB** color space are represented by three values: the red, green and blue components of the color. Each of the three **CalRGB** input values is in the range [0, 1].

A **CalRGB** color space is specified by a two component array of the form:

[/CalRGB dict]

where the contents of dict are described in Table 6.19b.

Table 6.19b CalRGB attributes

Key	Type	Semantics
WhitePoint	array	(Required) Same as for CalGray .
BlackPoint	array	(Optional) Same as for CalGray .

Table 6.19b *CalRGB attributes*

<i>Key</i>	<i>Type</i>	<i>Semantics</i>
Gamma	array	(<i>Optional</i>) Three numbers [Gr Gg Gb] that specify the gamma for the red, green, and blue components respectively. The governing equations are $R' = R^{G_r}$, $G' = G^{G_g}$, and $B' = B^{G_b}$, where R, G, and B are the input calibrated RGB values, and R', G', and B' are the gamma-modified values. Default value: [1 1 1].
Matrix	array	(<i>Optional</i>) Nine numbers [X _r Y _r Z _r X _g Y _g Z _g X _b Y _b Z _b] that specify the linear interpretation of the gamma-modified R, G, and B components, R', G', and B'. Default value: the identity matrix [1 0 0 0 1 0 0 0 1]. The transformation from R'G'B' to XYZ is given by: $X = R' \times X_r + G' \times X_g + B' \times X_b$ $Y = R' \times Y_r + G' \times Y_g + B' \times Y_b$ $Z = R' \times Z_r + G' \times Z_g + B' \times Z_b$

Colors in the **Lab** color space are represented by three values: the L*, a* and b* components of the color. The ranges of each of the three **Lab** input values are specified under the **Range** key in Table 6.19c.

A **Lab** color space is specified by a two component array of the form:

[/Lab dict]

where the contents of dict are described in Table 6.19c Lab attributes.

Table 6.19c *Lab attributes*

<i>Key</i>	<i>Type</i>	<i>Semantics</i>
WhitePoint	array	(<i>Required</i>) Same as for CalGray .
BlackPoint	array	(<i>Optional</i>) Same as for CalGray .
Range	array	(<i>Optional</i>) Four numbers [a _{min} a _{max} b _{min} b _{max}] specifying the range of the a* and b* components. That is, a* and b* are limited by a _{min} ≤ a* ≤ a _{max} , b _{min} ≤ b* ≤ b _{max} . Default value: [-100 100 -100 100]. The range of L* is always [0, 100].

A **CalCMYK** color space is specified by a two component array of the form:

[/CalCMYK dict]

where the contents of dict are not defined. These contents will be defined in a future version of PDF.

Implementation note

The CalCMYK color space resource type has been partially defined with the expectation that its definition will be completed in a future version of PDF. PDF 1.1 viewers should ignore CalCMYK color space attributes and render colors specified in this color space as if they had been specified using DeviceCMYK.

An example CalRGB color space resource is shown here for D65 white point, 1.8 gammas, and Trinitron phosphor chromaticities.

```
12 0 obj
[/CalRGB
<<
/WhitePoint [0.9505 1 1.0890]
/Gamma [1.8 1.8 1.8]
/Matrix [0.4497 0.2446 0.0252 0.3163 0.6720 0.1412 0.1845 0.0833 0.9227]
>>]
endobj
```

Add the following heading immediately following the above text:

Indexed color space resources

*In this subsection, replace the second sentence of the third paragraph, “If a name is used it must be **DeviceGray**, **DeviceRGB** or **DeviceCMYK**” with the following sentence:*

If a name is used it must be one of the allowed device (**DeviceGray**, **DeviceRGB** or **DeviceCMYK**) or device-independent (**CalGray**, **CalRGB**, **Lab**, or **CalCMYK**) color spaces.

Add the following heading and text at the end of section 6.8.5

Default color space resources

PDF 1.1 adds device-independent color spaces to the color spaces defined in PDF 1.0. Because viewers for PDF 1.0 may not expect these new color spaces and default gracefully when they are used, a second method for specifying the use of a device independent color space is provided in PDF 1.1. This second method allows an appropriate color space to be substituted for either the **DeviceGray** or **DeviceRGB** color spaces. The substitution is controlled by two special keys, **DefaultGray** and **DefaultRGB**, that can be used in the ColorSpace dictionary of the Resources dictionary of the current page.¹ They are used as follows.

1. A page’s resources are specified by the **Resources** key. The **Resources** may be inherited from a Pages object that is an ancestor of the page.

When a viewer is performing an operation that results in rendering to a medium, there is always a current color space which is established using the operators of Section 7.4 of the *Portable Document Format Reference Manual* or using the **ColorSpace** key of an image resource or an in-line image. When the current color space is **DeviceGray**, the ColorSpace dictionary of the Resources dictionary of the current page is checked for the presence of the **DefaultGray** key. If this key is present, then the color space that is the value of that key is used as the color space for the operation currently being performed. The value of the **DefaultGray** key may be either a **DeviceGray** or a **CalGray** color space specification.

Similarly, when the current color space is **DeviceRGB**, the ColorSpace dictionary of the Resources dictionary of the current page is checked for the presence of the **DefaultRGB** key. If this key is present, then the color space that is the value of that key is used as the color space for the operation currently being performed. The value of the **DefaultRGB** key may be either a **DeviceRGB** or a **CalRGB** color space specification.

Implementation note *The Acrobat 1.0 viewer ignores **DefaultRGB** and **DefaultGray**.*

6.8.6 XObject resources

Image resources

Add the following dictionary key to Table 6.20:

Table 6.20 *Image attributes (extended)*

Key	Type	Semantics
Intent	name	(Optional) A name which is a color rendering intent indicating the style of color rendering that should occur. For example, one might want to render images in a perceptual or pleasing manner while rendering spot colors with exact color matches. Intents are meaningful only for the device-independent color spaces

Implementation note *The Acrobat 1.0 viewers display an error if an image specifies an Intent.*

Add the following immediately after Table 6.20:

The **ColorSpace** key in an image resource may reference any color space defined in PDF 1.1, including, a device-independent color space. However, for compatibility with 1.0 viewers, the **DefaultRGB** or **DefaultGray** key should be used to reference a device-independent color space, as described in the Section “Default color space resources” on page 25.

The supported color rendering intents and their meanings are given below in Table 6.20a. Other intents are permitted, but a viewer based on the PDF 1.1 specification will most likely ignore its value. Default behavior in the absence of the **Intent** key or when it is not one of the three standard intents, is **RelativeColorimetric**.

Table 6.20a *Color rendering intents*

<i>Intent name</i>	<i>Semantics</i>
AbsoluteColorimetric	Provides for exact color reproductions. Example usage would be for spot colors. Reproductions are made with respect to the reference illuminant.
RelativeColorimetric	Provides for relative color reproductions. Example usage would be for spot colors. Reproductions are made with respect to the unmarked media, for example the white of paper.
Perceptual	Provides a perceptual or pleasing rendering for both in- and out-of-gamut colors. Example use is scanned images.
Saturation	Provides a rendering in which saturation is emphasized. Example use is business graphics.

Images contain optional Decode arrays that define the mapping from image data values to colors in the image color space. Add the following key to Table 6.21:

Table 6.21 *Default Decode arrays for various color spaces (extended)*

<i>Color space</i>	<i>Default decode array</i>
CalGray	[0 1]
CalRGB	[0 1 0 1 0 1]
Lab	[0 100 a _{Min} a _{Max} b _{Min} b _{Max}] where a _{Min} , a _{Max} , b _{Min} , and b _{Max} correspond to the entries in the Range array of the image dictionary ColorSpace entry. 0 and 100 are the first two entries since the range on L* is always [0, 100].

Pass-through PostScript language resources

Add this new subsection at the end of Section 6.8.6, “XObject resources”:

PDF 1.1 enables a document to include PostScript language fragments in a page description. These fragments are printer-dependent and only take effect when printing on a PostScript printer. They have no effect either when viewing the file or when printing to a non-PostScript printer. In addition, any

other applications which understand PDF are unlikely to be able to interpret the PostScript language fragments. Hence, this capability should only be used if there is no other way to achieve the same result.

A PostScript resource is an XObject whose **Subtype** is **PS**. When a document is printed to a PostScript printer, the contents of the resource stream replace the **Do** command that references the resource. This stream is copied without interpretation and may include PostScript comments. In any other case, the resource is ignored. When printing to a PostScript Level 1 printer, if the XObject contains a **Level1** key, the value of that key, which must be a stream, will be used instead of the resource stream contents.

The PostScript fragment may use Type 1 and TrueType fonts listed in the resources of the page containing the fragment. It may not use Type 3 fonts.

The list of PostScript operators that will be allowed in pass-through PostScript resources is currently being defined and will be published shortly.

The PostScript XObject is not compatible with 1.0 viewers. The following method can be used instead to create PostScript pass-through data when compatibility with 1.0 viewers is necessary. A form should be defined with an empty stream content. It should include a **BBox** of all zeros, a **FormType** of 1, and a **Matrix** that is the identity matrix. It should include a **Subtype2** key whose value is **PS** and a **PS** key whose value is a stream that contains the PostScript language pass-through data. It may also contain a **Level1** key as described previously in this section.

6.9 Info dictionary

Replace the first paragraph of this section with the following:

A document’s trailer may contain a reference to an Info dictionary that provides information about the document. This optional dictionary may contain one or more keys whose values should be strings. These strings may be displayed in an Acrobat viewer’s Document Info dialog. The characters in these strings are encoded using the predefined encoding **PDFDocEncoding**, described in Appendix C.

Add the following keys to Table 6.23:.

Table 6.23 PDF Info dictionary attributes (extended)

Key	Type	Semantics
Title	string	(Optional) The title of the document.
Subject	string	(Optional) The subject of the document.
Keywords	string	(Optional) Keywords associated with the document.

Table 6.23 *PDF Info dictionary attributes (extended)*

<i>Key</i>	<i>Type</i>	<i>Semantics</i>
ModDate	string	(<i>Optional</i>) The date the document was last modified. It should be in the format described in Section Strings

Add the following text after Table 6.23:

If any Info string follows the date format described in Section 4.4, “Strings,” that string should be displayed as a human-readable date. In particular, the 1.0 key **CreationDate** and the 1.1 key **ModDate** should use this format. However, the “D:” prefix is not optional for Info strings. Info strings are otherwise uninterpreted.

Info keys and strings may be added to or changed by users or extensions, and some extensions may choose to permit searches on these keys. PDF 1.1 does not define short names for the keys in Table 6.23, to make it easier to browse and edit Info dictionary entries. New names should be chosen with care so that they make sense to users.

While private data can be stored in the Info dictionary, it is more appropriate to store it in the Catalog. This allows a user or program to alter entries in the Info dictionary with less chance of unforeseen side effects.

6.10 Articles

Add this new section to Chapter 6:

Article threads allow associating related document elements in a way that enables the user to more easily read through a flow of text or other information that may span multiple columns or pages.

A PDF document may include one or more article threads. Each thread has a title and a list of thread elements, which are referred to as *beads*. A viewer may allow the user to select a particular thread and then navigate through the thread with the viewer automatically maintaining a comfortable reading zoom level while moving from one bead to the next rather than from one page to the next.

If a document includes any threads, they are stored in an array as the value of the **Threads** key in the Catalog object. Each thread and its beads are dictionaries. Table 6.24 lists the attributes of athread dictionary, and Table 6.25 lists the attributes of a bead dictionary.

Table 6.24 *Thread attributes*

<i>Key</i>	<i>Type</i>	<i>Semantics</i>
F (First)	dict	<i>(Required; must be indirect reference)</i> Specifies the bead that is the first element of this thread.
I (Info)	dict	<i>(Optional)</i> Information about the thread, similar to the document's Info dictionary. Entries in this dictionary should be strings encoded using the predefined encoding PDFDocEncoding , described in Appendix C of the PDF Reference Manual.

Table 6.25 *Bead attributes*

<i>Key</i>	<i>Type</i>	<i>Semantics</i>
T (Thread)	dict	<i>(Required for the first bead of a thread; must be indirect reference)</i> Specifies the thread of which this bead is the first element.
V (Prev)	dict	<i>(Required; must be indirect)</i> Specifies the previous bead of this thread; for the first bead in a thread, V specifies the last bead in the thread.
N (Next)	dict	<i>(Required; must be indirect)</i> Specifies the next bead of this thread; for the last bead in a thread, N specifies the first bead in the thread.
P (Page)	dict	<i>(Required; must be indirect)</i> Specifies the page on which this bead appears.
R (Rect)	array	<i>(Required)</i> Rectangle specifying the location of this bead.

Here is an example of a thread with three beads:

```

22 0 obj
<< /F 23 0 R /I << /Title (Man Bites Dog) >> >>
endobj
23 0 obj
<< /T 22 0 R /V 25 0 R /N 24 0 R /P 8 0 R /R [158 247 318 905] >>
endobj
24 0 obj
<< /V 23 0 R /N 25 0 R /P 8 0 R /R [322 246 486 904] >>
endobj
25 0 obj
<< /V 25 0 obj /N 23 0 obj /P 10 0 obj /R [157 254 319 903] >>
endobj

```

The Page object for each page on which beads appear should contain a **B** key, as described in Section 6.4, “Page objects.” The value of this key is an array of indirect references to each bead on the page, in drawing order.

Implementation note *The thread array and dictionary objects are invisible to 1.0 viewers on all platforms. Consequently, insert and delete pages operations will not carry along any threads.*

6.11 File ID

A PDF file may contain a reference to another PDF file. Storing a filename, even in a platform-independent format, does not guarantee that the file can be found, even if it exists and its name has not been changed. Different server software applications often present different names for the same file. For example, servers running on DOS platforms must convert all file names to eight letters and a three-letter extension. Different servers use different strategies for converting long names to this format.

References to PDF files can be made more reliable by making the PDF file reference consist of two parts: (1) a normal operating system-based file reference and (2) a file ID. The file ID is a short value that characterizes the file and is stored with the file. Placing a file ID with the file reference and in the file itself increases the chances that a file reference can be resolved correctly. Matching the file ID in the reference with the file ID in the file indicates whether the desired file was found.

Implementation note *The indexes created by Adobe Catalog™ also contain references to PDF files.*

PDF 1.1 recommends that files have an **ID** key in their trailer. The value of this key is an array of two strings. The first element is a permanent ID based on the contents of the file at the time the file was created. This ID does not change when the file is incrementally updated. The second element is a changing ID based on the contents of the file at the time the file is incrementally updated. When a file is first written, both IDs are set to the same value. When resolving a file reference, if both IDs match, it is very likely that the correct file has been found. If only the first ID matches, then a different version of the correct file has been found.

Implementation note *Although this key is not required, all Adobe producers will include this key. Acrobat Exchange will add this key when saving a file if it is not present*

To help insure the uniqueness of the file ID, it is recommended that file ID be computed using a message digest algorithm such as MD5, as described in *RFC 1321: The MD5 Message-Digest Algorithm*. It is recommended that the following information be fed to the message digest algorithm:

- the current time
- a string representation of the location of the file, usually a path name
- the document size in bytes
- the values of all the entries in the document's Info dictionary.

Implementation note

Adobe applications use the above information and the MD5 message digest algorithm to calculate fileIDs. Note that the calculation of the fileIDs need not be reproducible. All that matters is that the file IDs are likely to be unique. For example, two implementations of this algorithm might use different formats for the current time. This will cause them to produce different file IDs for the same file created at the same time, but this does not affect the uniqueness of the ID.

6.12 Encrypt dictionary

PDF 1.1 allows strings and streams in documents to be encrypted to protect their content from unauthorized access. A protected document contains an additional entry in the trailer dictionary, the **Encrypt** key, whose value is the Encrypt dictionary. Access to the document content is controlled by the security handler that is identified by the Encrypt dictionary. The name of this handler is specified by the Encrypt dictionary's **Filter** key. The security handler permits users to protect documents with a password and to limit operations on documents, for example, preventing documents from being printed or modified.

The Encrypt dictionary may contain additional information required by the security handler to determine if a user should be allowed to access the document. For example, the security handler may use a dialog to obtain a password that allows access to the document. Table 6.26 describes the required keys in the Encrypt dictionary defined by the security handler built into the Acrobat 2.0 viewers. However, the dictionary may include additional keys not specified here.

Table 6.26 Standard security handler attributes

Key	Type	Semantics
Filter	name	(Required by all security handlers) Name of security handler. The name of the handler built into Acrobat 2.0 is Standard.
R (Revision)	number	(Required) Revision number of algorithm used to encode data in this dictionary. The revision number for the Standard security handler in Acrobat 2.0 is 2.
O (Owner)	string	(Required) Data describing password needed to gain full access to file.
U (User)	string	(Required) Data describing password needed to open file.
P (Permissions)	string	(Required) Data describing permissions granted to user who opens a file with the U password.

All strings and streams in a protected document, except those in the Encrypt dictionary, are encrypted using the RC4 encryption algorithm. Streams are encrypted after all stream encoding filters have been applied (and are decrypted before the stream decoding filters are applied). Decryption of

strings, other than those in the Encrypt dictionary, is done after escape sequence processing and hex decoding as appropriate to the string representation form as described in Section 4.4, “Strings.” Strings in the Encrypt dictionary are encrypted and decrypted by the security handler itself.

Implementation note

The Acrobat 2.0 viewer provides a Standard security handler that requires a password to access a protected document and allows control over which operation that viewer will allow to be performed on the content of the document. Other security handlers may be used, but such usage is dependent on having a plug-in extension to process the information in the Encrypt dictionary.

The Standard security handler will be described in a separate note that will appear in the near future.

7 Page description (updated)

This section includes the changes required to update Chapter 7 of the *PDF Reference Manual* to version 1.1. These include the addition of operators to specify device independent colors and to pass through PostScript in-line.

7.2 Graphics state

Remove the “fill color” and “stroke color” entries in the original Table 7.1 and add the following entries:

Table 7.1 *General graphics state parameters (extended)*

<i>Parameter</i>	<i>Operator</i>	<i>Operator may not appear...</i>
fill color space	g, rg, k, cs	within a path
fill color	g, rg, k, sc	within a path
stroke color space	G, RG, K, CS	within a path
stroke color	G, RG, K, SC	within a path
rendering intent	ri	within a path

Add the following three new sections:

7.2.12 Fill color space

The color space in which the fill color is specified. See Section 7.4, “Color operators.”

7.2.13 Stroke color space

The color space in which the stroke color is specified. See Section 7.4, “Color operators.”

7.2.14 Rendering intent

A name which is a color rendering intent indicating the style of color rendering that should occur. See Section 6.8.6, “XObject resources,” and especially Table 6.20a, “Color rendering intents,” for further detail.

7.4 Color operators

Device-independent color spaces can be used within page descriptions as well as in image resources.

Replace the first paragraph with the following three paragraphs:

Colors may be specified in any device or device-independent color space, but not in an indexed color space. The operators that set a device color also set the color space that is used. The default color space is **DeviceGray** and the default fill and stroke colors are both black.

Implementation note

For compatibility with PDF 1.0 viewers, it is strongly recommended that device-dependent colors be specified using the 1.0 operators and that device-independent colors be specified using the color space substitution method defined in Section 6.8.5, “Color space resources.”

Add the following operator descriptions to the end of the list in Section 7.4.

c0 c1 c2 c3 **sc** **setcolor** (fill)

Sets the color to use for filling paths. The number of operands required and their interpretation is based on the current fill color space. For **DeviceGray** and **CalGray**, one operand is required. For **DeviceRGB**, **CalRGB**, and **Lab**, three operands are required. For **DeviceCMYK** and **CalCMYK**, four operands are required.

c0 c1 c2 c3 **SC** **setcolor** (stroke)

Same as **sc**, but for stroking paths.

colorspace **cs** **setcolorspace** (fill)

Sets the fill color space in the graphics state. **colorspace** is a name of a **ColorSpace** resource defined in the **Resources** dictionary of the current page. In-line specification of the color space is not permitted. The **ColorSpace** resource may be any of the device or device independent color spaces, but it may not be an indexed space.

colorspace **CS** **setcolorspace** (stroke)
 Same as **cs**, but for strokes.

intent **ri**
 Sets the color rendering intent in the graphics state. intent is a name of a color rendering intent which indicates the style of color rendering that should occur, as described in Section 6.8.6, “XObject resources.” The default rendering intent is **RelativeColorimetric**.

Implementation note *If an Acrobat 1.0 viewer reads a page containing any of the setcolor, setcolorspace, or intent operators, it will report an error. Errors can be ignored by the user and objects will be displayed, but colors will most likely be black (the default).*

7.7.4 Text string operators

Replace the note in the third paragraph that begins, “If a current point has not been established..,” with the following clarification:

Note *The default current point is at the page origin. Therefore, unless some prior operation in the same text object changes the current point, the text will appear at the origin. It is suggested that a **Tm** operation be used to establish the initial current point in a text object at the position in text space where initial text is to appear. Subsequent text operations may change the current point.*

7.9 In-line image operators

In-line images can also specify device-independent color spaces using the same keys as image resources. Add the following entries to Table 7.3:

Table 7.3 Abbreviations for in-line image names(extended)

Name	Abbreviated name
CalGray	CG
CalRGB	CR
CalCMYK	CC
Intent	no abbreviation
Lab	no abbreviation

7.11 In-line pass-through PostScript fragments

Add this new section to Chapter 7:

PDF 1.1 enables a document to include PostScript language fragments in a page description. These fragments are printer-dependent and only take effect when printing on a PostScript printer. They have no effect when viewing the file or when printing to a non-PostScript printer. In addition, any other applications which understand PDF are unlikely to be able to interpret PostScript language fragments. Hence, this capability should only be used if there is no other way to achieve the same result. See the “Pass-through PostScript language resources” subsection of Section 6.8.6 for additional information.

Implementation note *If an Acrobat 1.0 viewer reads a page containing this operator, it will report an error. The operator is otherwise ignored.*

string **PS**

The pass-through PostScript operator **PS** provides an in-line equivalent to a PostScript language resource. The **PS** operator has one argument, a string. When a **PS** operator is encountered while a document is being printed to a PostScript printer, the contents of the string are placed into the PostScript output as the argument of an instance of the PostScript operator **exec**. This string is copied without interpretation and may include PostScript comments. In any other case, the **PS** operator consumes its argument and has no other effect.

7.12 Compatibility operators

Add this new section to Chapter 7:

PDF does not specify a viewer’s behavior when it encounters an undefined page description operator. However, Appendix G.2.7 does describe the behavior of the Adobe Acrobat 1.0 and 2.0 viewers. An Acrobat viewer usually alerts the user when it encounters an undefined page description operator. The operators below modify this behavior.

Implementation note *If an Acrobat 1.0 viewer reads a page containing these operators, it will report an error. The operators are otherwise ignored.*

— **BX**

This operator directs a viewer to not report any undefined operators until a matching **EX** is encountered. (**BX**–**EX** pairs may nest.)

— **EX**

This operator ends a section of page description in which undefined operators should not be reported.

Appendix F PDF name registry

This new appendix is not part of the PDF specification:

With the introduction of Adobe Acrobat 2.0, it has become easy for third parties to add private data to PDF documents and to change viewer behavior based on this data. This capability greatly increases the potential functionality of extensions. However, Acrobat users have certain expectations when opening a PDF document, no matter what extensions are available. PDF enforces certain restrictions on private data in order to meet these expectations.

A PDF producer or Acrobat plug-in may define new action, destination, annotation, and encryption types. If a user opens a PDF document and the extension that implements the new type of object is unavailable, the viewers will behave as described in Appendix G.2.

A PDF producer or Acrobat plug-in may also add keys to any PDF object that is implemented as a dictionary except the trailer dictionary.

To avoid conflicts with third-party names and with future versions of PDF, Adobe maintains a registry of names, similar to the registry it maintains for Document Structuring Conventions. Third-party developers should only add private data that conforms to the registry rules.

The registry includes three classes of names:

- First-class names have widespread applicability. All the names defined in PDF 1.0 and 1.1 are first-class names. Plug-ins that are publicly available should use first-class names for all private data.
- Second-class names are applicable to a limited set of users. If a developer writes a plug-in for an in-house application, second-class names are usually appropriate. Developers can either register a prefix or an entire name. Registering a prefix allows a developer to define as many names as necessary without individually registering each one. Registering second-class names guarantees that no conforming PDF file will ever have names that conflict with the registered names.
- Third-class names begin with a special prefix reserved by Adobe for private plug-ins. These names are intended for use in files that will never be seen by other third parties since these names may conflict with third class names defined by others. Names beginning with “XX” are third-class names.

New keys for the Info dictionary in the Catalog and in Threads need not be registered.

Appendix G

Compatibility

This new appendix is not part of the PDF specification:

The goal of the Adobe Acrobat family of products is to enable people to easily and reliably exchange and view electronic documents. Ideally, “easily and reliably” means that any Acrobat viewer should be able to display the contents of any PDF file even if the PDF file was created long before or long after the viewer. Of course, new versions of viewers are introduced to provide additional capabilities not present before. Furthermore, beginning with Acrobat 2.0, viewers may accept plug-in extensions, making some Acrobat 2.0 viewers more capable than others depending on what extensions are present. Both the viewers and PDF itself have been designed to enable users to view everything in the document that the viewer understands and to ignore or inform the user about objects not understood. The decision whether to ignore or inform the user is made on a feature-by-feature basis.

The original PDF specification did not specify how a viewer should behave when it reads a file that does not conform to the specification. This addendum provides this information. The PDF version number associated with a file determines how it should be treated when a viewer encounters a problem.

G.1 Version numbers

The PDF version number consists of a major and minor version. The version number is part of the PDF header, the first line of the file. This header takes the form:

`%PDF-M.m`

where *M* is the major number and *m* is the minor number.

If PDF changes in a way that current viewers will be unlikely to read a document without a serious error, the major version number will be incremented. A serious error is an error that prevents pages from being viewed. Adding a new filter type for page contents would require a change in the major version number. Adding a new page description operator would not.

If PDF changes in a way that a viewer will display an error message but continue its work, the minor version number will change. Adding new page description operators would require a change in the minor version number.

If PDF changes in a way that current viewers are unlikely to detect, the version number need not change. This includes the addition of private data that can be gracefully ignored by consumers that do not understand that data. An example is adding a key to a dictionary object such as the Catalog.

An Acrobat viewer will try to read any file with a valid PDF header, even if the version number is newer than the viewer itself. It will read without errors any file that does not require a plug-in, even if the version number is older than the viewer. Some documents may require a plug-in to display an

annotation or execute a link or bookmark action. Viewer behavior in this situation is described below. However, a plug-in is never required to display the contents of a page.

If a viewer opens a document with a newer major version number than it expects, it warns the user that it is unlikely to be able to successfully read the document and that the user will not be able to change or save the document. At the first error related to document processing, the viewer will notify the user that an error has occurred but that no further errors will be reported.¹ Processing will continue if possible. Acrobat Exchange will not permit a document with a newer major version number to be inserted into another document.

If a viewer opens a document with a newer minor version number than it expects, it silently remembers the version number. Only if it encounters an error does it alert the user. At this point it notifies the user that the document is newer than expected, that an error has occurred, and that no further errors will be reported. The document may not be incrementally saved but can be saved to a new file. The saved file will continue to have the new version number. A user may insert a document with a newer minor version into another document. The resulting document can be saved. Its version number will be the maximum of the version number of the original document and the documents inserted into the original.

When opening a file, the Acrobat viewers are very liberal in their check for a valid PDF header. All viewers allow the header to appear anywhere in the first 1,000 bytes of the file. The 1.0 viewers require only that 'PDF-' appear in the header, but ignore the rest of the header. The 2.0 viewers search for a header of the form described above. However, they also accept a header of the form:

%!PS-Adobe-N.n PDF-M.m

where N.n is an Adobe Document Structuring Conventions version number and M.m is a PDF version number. (The *PostScript Language Reference Manual* describes the Document Structuring Conventions).

G.2 Viewer compatibility behavior

This section describes how the Acrobat 1.0 and 2.0 viewers behave when encountering items that do not conform to the PDF 1.0 specification. It is planned that future Acrobat viewers will behave the same as Acrobat 2.0 viewers.

1. Some errors will always be reported, including file I/O errors, extension loading errors, out-of-memory errors, and notification that a command failed.

G.2.1 Dictionary keys

Adding key-value pairs not described in the PDF specification to dictionary objects usually does not affect the behavior of 1.0 viewers and never affects the behavior of Acrobat 2.0 viewers. These keys are ignored. If a dictionary object such as an annotation is copied into another document during a page insertion (or in Acrobat 2.0 viewers during a page extraction), all key-value pairs are copied. If a value is an indirect reference to another object, that object may be copied as well, depending on the key.

In some cases a 1.0 viewer will display an error if it finds an unknown key in a dictionary. These cases are keys in image dictionaries (both XObjects and in-line images) and keys in DecodeParms dictionaries for filters.

See Appendix F for information on how to choose key names that are compatible with future versions of PDF.

G.2.2 Annotations

An annotation is a dictionary element of a page's **Annots** array. Its **Subtype** specifies the kind of annotation it is. Only **Text** and **Link** are defined by PDF 1.0. If a 1.0 viewer reads a page with an annotation whose **Subtype** is not **Text** or **Link**, it displays an error. It displays one error per page no matter how many annotations are present.

An Acrobat 2.0 viewer displays unknown annotations in a closed form similar to text annotations, with an icon containing a question mark. If the user opens the annotation, an alert appears with a message giving the annotation type and explaining that an unavailable plug-in is required to open it. An unknown annotation can be selected, moved, and deleted. Every annotation type must specify its position and size using the **Rect** key.

G.2.3 Destinations and actions

A link or a bookmark in PDF 1.0 is a dictionary that contains a **Dest** key that specifies a new view of the document that should be displayed when the link or bookmark is activated. A destination is an array whose first element is a name that serves as destination type. It determines the interpretation of subsequent array elements. If a 1.0 viewer encounters an unknown destination type, no action is performed and no error is reported when the user activates the link or bookmark. An Acrobat 2.0 viewer will display a message when it finds an unknown destination type.

PDF 1.1 adds several new destination types described in Section 6.6.3, "Destinations and actions." This section also describes actions, which have superseded destinations in PDF 1.1. A 1.0 viewer ignores actions. It does nothing if it does not find a **Dest** key in a link or bookmark.

G.2.4 XObjects

An XObject is a stream or dictionary that is referred to by name from a page description by the **Do** operator. The effect of the operator is determined by the type of the XObject. PDF 1.0 supports Image and Form XObjects. A 1.0 viewer displays an error for each XObject of a different type, no matter how many are on a page.

Plug-ins may not add XObject types since they are considered part of the page, and a viewer without plug-ins should be able to display a page. If an Acrobat 2.0 viewer encounters an unknown XObject type, it will be in a document with a PDF version number greater than 1.1. The viewer will display an error specifying the type of XObject but not report any further errors.

To avoid the 1.0 viewers' error behavior, new XObject types in PDF 1.1 can be specified as Forms, providing the required Form keys but having no content. The required keys are **Name**, **BBox**, **FormType**, and **Matrix**. **Subtype2** can specify the actual type, and additional keys can specify additional information. See Section 6.8.6, "XObject resources," for a description of the one new XObject type added in PDF 1.1.

A 1.0 viewer checks the **FormType** and displays an error once per form if the **FormType** is not 1. It also displays an error that it cannot find the form each the time a page references the form. An Acrobat 2.0 viewer checks that the **FormType** is 1 and puts up an error once per document and then ignores the form if its **FormType** is not 1.

G.2.5 Color spaces

An image has a **ColorSpace** key. A 1.0 viewer displays an error each time it finds an image with a color space that is not one of the PDF 1.0 color spaces. Like XObjects, color spaces may not be added by plug-ins. If an Acrobat 2.0 viewer encounters an unknown **ColorSpace** type, it will be in a document with a PDF version number greater than 1.1. The viewer will display an error specifying the type of **ColorSpace** but not report any further errors.

PDF 1.1 defines three additional color spaces: **CalGray**, **CalRGB**, and **Lab**. To be more compatible with 1.0 viewers, PDF 1.1 allows an image color space to be specified indirectly through the page resources. When an Acrobat 2.0 viewer processes an image and the image's **ColorSpace** key specifies **DeviceRGB**, the viewer looks in the page's resources for a **ColorSpace** called **DefaultRGB**. If this key is present, the color space associated with it is used instead of **DeviceRGB**. Similarly, if an image's **ColorSpace** key specifies **DeviceGray**, the viewer looks for **DefaultGray**. The 1.0 viewer ignores **DefaultRGB** and **DefaultGray**.¹

1. A page's resources are specified by the **Resources** key. The **Resources** may be inherited from a Pages object that is an ancestor of the page.

See Section 7.4, “Color operators,” for an explanation of the use of color spaces in page descriptions. The presence of **DefaultRGB** or **DefaultGray** change the interpretation of some color operators.

G.2.6 Filters

PDF uses stream objects to encapsulate image, indexed color space, thumbnail, and embedded font data and page, form, and Type 3 character descriptions. These streams usually use filters to compress their data. The legal PDF 1.0 filters are the same as those available in PostScript Level 2. The 1.0 viewer behavior when encountering an unknown filter depends on its context:

- Image resource. The image does not appear but no error is reported.
- In-line image. (An in-line image is specified directly in a page description, while an image resource is specified outside of a page and referenced from the page.) An error is reported, and page processing stops.
- Indexed color space. An error is reported, but page processing continues.
- Thumbnail. An error is reported, no more thumbnails are displayed, but the thumbnails can be deleted and created again.
- Embedded font. An error is reported, and the viewer behaves as if the font is not embedded.
- Page description. An error is reported, and page processing stops.
- Form description. An error is reported, and page processing stops.
- Type 3 character description. An error is reported, and page processing stops.

The Acrobat 2.0 viewers do not allow plug-ins to provide additional filters. If an unrecognized filter is encountered, an Acrobat 2.0 viewer will specify the context in which the filter was found. If an error occurs while displaying a page, only the first error is reported. Subsequent behavior depends on the context:

- Image resource. The image does not appear but page processing continues.
- In-line image. Page processing stops.
- Indexed color space. The image does not appear but page processing continues.
- Thumbnail. An error is reported, no more thumbnails are displayed, but the thumbnails can be deleted and created again.
- Embedded font. The viewer behaves as if the font had not been embedded.

- Page description. Page processing stops.
- Form description. The form does not appear but page processing continues.
- Type 3 character description. The character does not appear but page processing continues. The current point is adjusted based on the character's width.

Operations that process pages, such as Find and Create Thumbnails, stop as soon as an error occurs.

G.2.7 Page description operators

A 1.0 viewer reports an error the first time it finds an unknown operator or an operator with too few operands, but it continues processing the page. If it finds ten errors on a page, it reports back to the user and asks whether to continue processing. No further errors are reported. Each time an error occurs, the operand stack is cleared. Acrobat 2.0 viewers behave the same, although there is no additional warning if ten errors are encountered.

PDF 1.1 provides new page description operators for specifying device-independent color and pass-through PostScript fragments. Since these operators are incompatible with 1.0 viewers, PDF 1.1 provides alternative compatible methods as well.

G.2.8 Procedure sets

Each page includes a ProcSet resource that describe the PostScript procedure sets required to print the page. A 1.0 viewer ignores requests for unknown procedure sets. An Acrobat 2.0 viewer warns the user that a procedure set is unavailable and cancels printing.

Copyrights and permissions to use PDF

The general idea of utilizing an interchange format for final-form documents is in the public domain. Anyone is free to devise his or her own set of unique commands and data structures that define an interchange format for final-form documents. Adobe owns the copyright in the data structures, operators and the written specification for the particular interchange format called the Portable Document Format. These elements may not be copied without Adobe's permission.

Adobe will enforce its copyright. Adobe's intention is to maintain the integrity of the Portable Document Form as a standard. This enables the public to distinguish between the Portable Document Format and other interchange formats for final-form documents.

However, Adobe desires to promote the use of the Portable Document Format for information interchange among diverse products and applications. Accordingly, Adobe gives permission to anyone to:

- Prepare files in which the file content conforms to the Portable Document Format.
- Write drivers and applications that produce output represented in the Portable Document Format.
- Write software that accepts input in the form of the Portable Document Format and displays the results, prints the results or otherwise interprets a file represented in the Portable Document Format.
- Copy Adobe's copyrighted list of operators and data structures to the extent necessary to use the Portable Document Format for the above purposes.

The only condition on such permission is that anyone who uses the copyrighted list of operators and data structures in this way must include an appropriate copyright notice.

This limited right to use the copyright list of operators and data structures does not include the right to copy the *Portable Document Format Reference Manual*, other copyrighted material from Adobe, or the software in any of Adobe's products which use the Portable Document Format, in whole or in part.

Bibliography

Adobe Systems Incorporated, *Portable Document Format Reference Manual*, Addison-Wesley, 1993, ISBN 0-201-62628-4.

Adobe Systems Incorporated, *PostScript Language Reference Manual*, Addison-Wesley, 1990, ISBN 0-201-18127-4.

CCITT, *Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1)*, 1988.

Microsoft Corp., *TrueType 1.0 Font Files*, Revision 1.00, May 1992.

R. Rivest, *RFC 1321: The MD5 Message-Digest Algorithm*, April 1992.

Errata for the Portable Document Format Reference Manual

- Table 6.10, p.62 — The value of the **Name** key must be a name, not a string.
- Section 4.5, p. 27 — Names may not contain whitespace characters, *e.g.* space or tab, in addition to the characters listed.
- Example 6.10, p. 65 — Object 21 begins `/Widths [`. The object should not contain `/Widths`, but should simply begin with `[`.
- Example 6.11, p. 66 — Object 19 begins `/Widths [`. The object should not contain `/Widths`, but should simply begin with `[`.
- Chapter 11 — An additional image optimization suggestion: don't use images that have already been halftoned.
- Section 6.8, p.61 — Correction to the wording of the last sentence in the second paragraph (the substantive portion of the change is the addition of several words, which are shown in italics here):
Only *the list of* ProcSet resources is represented as an array...; all other resource *lists* are represented as dictionaries...
- Section 7.4, p.95 — Color operators are also graphics state operators, but this is not mentioned in the text.

