First Edition
2008-7-1

Document management — Portable document format — Part 1: PDF 1.7

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing.

Copyright Notice

Contents

Page

Foreword

On January 29, 2007, Adobe Systems Incorporated announced it's intention to release the full Portable Document Format (PDF) 1.7 specification to the American National Standard Institute (ANSI) and the Enterprise Content Management Association (AIIM), for the purpose of publication by the International Organization for Standardization (ISO).

PDF has become a de facto global standard for more secure and dependable information exchange since Adobe published the complete PDF specification in 1993. Both government and private industry have come to rely on PDF for the volumes of electronic records that need to be more securely and reliably shared, managed, and in some cases preserved for generations. Since 1995 Adobe has participated in various working groups that develop technical specifications for publication by ISO and worked within the ISO process to deliver specialized subsets of PDF as standards for specific industries and functions. Today, PDF for Archive (PDF/A) and PDF for Exchange (PDF/X) are ISO standards, and PDF for Engineering (PDF/E) and PDF for Universal Access (PDF/UA) are proposed standards. Additionally, PDF for Healthcare (PDF/H) is an AIIM proposed Best Practice Guide. AIIM serves as the administrator for PDF/A, PDF/E, PDF/UA and PDF/H.

In the spring of 2008 the ISO 32000 document was prepared by Adobe Systems Incorporated (based upon PDF Reference, sixth edition, Adobe Portable Document Format version 1.7, November 2006) and was reviewed, edited and adopted, under a special "fast-track procedure", by Technical Committee ISO/TC 171, *Document management application*, Subcommittee SC 2, *Application issues*, in parallel with its approval by the ISO member bodies.

In January 2008, this ISO technical committee approved the final revised documentation for PDF 1.7 as the international standard ISO 32000-1. In July 2008 the ISO document was placed for sale on the ISO web site (http://www.iso.org).

This document you are now reading is a copy of the ISO 32000-1 standard. By agreement with ISO, Adobe Systems is allowed to offer this version of the ISO standard as a free PDF file on it's web site. It is not an official ISO document but the technical content is identical including the section numbering and page numbering.

Introduction

ISO 32000 specifies a digital form for representing documents called the Portable Document Format or usually referred to as PDF. PDF was developed and specified by Adobe Systems Incorporated beginning in 1993 and continuing until 2007 when this ISO standard was prepared. The Adobe Systems version PDF 1.7 is the basis for this ISO 32000 edition. The specifications for PDF are backward inclusive, meaning that PDF 1.7 includes all of the functionality previously documented in the Adobe PDF Specifications for versions 1.0 through 1.6. It should be noted that where Adobe removed certain features of PDF from their standard, they too are not contained herein.

The goal of PDF is to enable users to exchange and view electronic documents easily and reliably, independent of the environment in which they were created or the environment in which they are viewed or printed. At the core of PDF is an advanced imaging model derived from the PostScript® page description language. This PDF Imaging Model enables the description of text and graphics in a device-independent and resolution-independent manner. To improve performance for interactive viewing, PDF defines a more structured format than that used by most PostScript language programs. Unlike Postscript, which is a programming language, PDF is based on a structured binary file format that is optimized for high performance in interactive viewing. PDF also includes objects, such as annotations and hypertext links, that are not part of the page content itself but are useful for interactive viewing and document interchange.

PDF files may be created natively in PDF form, converted from other electronic formats or digitized from paper, microform, or other hard copy format. Businesses, governments, libraries, archives and other institutions and individuals around the world use PDF to represent considerable bodies of important information.

Over the past fourteen years, aided by the explosive growth of the Internet, PDF has become widely used for the electronic exchange of documents. There are several specific applications of PDF that have evolved where limiting the use of some features of PDF and requiring the use of others, enhances the usefulness of PDF. ISO 32000 is an ISO standard for the full function PDF; the following standards are for more specialized uses. PDF/X (ISO 15930) is now the industry standard for the intermediate representation of printed material in electronic prepress systems for conventional printing applications. PDF/A (ISO 19005) is now the industry standard for the archiving of digital documents. PDF/E (ISO 24517) provides a mechanism for representing engineering documents and exchange of engineering data. As major corporations, government agencies, and educational institutions streamline their operations by replacing paper-based workflow with electronic exchange of information, the impact and opportunity for the application of PDF will continue to grow at a rapid pace.

PDF, together with software for creating, viewing, printing and processing PDF files in a variety of ways, fulfils a set of requirements for electronic documents including:

- preservation of document fidelity independent of the device, platform, and software,
- merging of content from diverse sources—Web sites, word processing and spreadsheet programs, scanned documents, photos, and graphics—into one self-contained document while maintaining the integrity of all original source documents,
- collaborative editing of documents from multiple locations or platforms,
- digital signatures to certify authenticity,
- security and permissions to allow the creator to retain control of the document and associated rights,
- accessibility of content to those with disabilities,
- extraction and reuse of content for use with other file formats and applications, and
- electronic forms to gather data and integrate it with business systems.

The International Organization for Standardization draws attention to the fact that it is claimed that compliance with this document may involve the use of patents concerning the creation, modification, display and processing of PDF files which are owned by the following parties:

- Adobe Systems Incorporated, 345 Park Avenue, San Jose, California,95110-2704, USA

ISO takes no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights has assured the ISO that they are willing to negotiate licenses under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with ISO. Information may be obtained from those parties listed above.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. ISO shall not be held responsible for identifying any or all such patent rights.

A repository of referenced documents has been established by AIIM (http://www.aiim.org/pdfrefdocs). Not all referenced documents can be found there because of copyright restrictions.

Document management — Portable document format —

Part 1:
PDF 1.7

IMPORTANT — The electronic file of this document contains colours which are considered to be useful for the correct understanding of the document. Users should therefore consider printing this document using a colour printer.

1 Scope

This International Standard specifies a digital form for representing electronic documents to enable users to exchange and view electronic documents independent of the environment in which they were created or the environment in which they are viewed or printed. It is intended for the developer of software that creates PDF files (conforming writers), software that reads existing PDF files and interprets their contents for display and interaction (conforming readers) and PDF products that read and/or write PDF files for a variety of other purposes (conforming products).

This standard does not specify the following:

- specific processes for converting paper or electronic documents to the PDF format;
- specific technical design, user interface or implementation or operational details of rendering;
- specific physical methods of storing these documents such as media and storage conditions;
- methods for validating the conformance of PDF files or readers;
- required computer hardware and/or operating system.

2 Conformance

2.1 General

Conforming PDF files shall adhere to all requirements of the ISO 32000-1 specification and a conforming file is not obligated to use any feature other than those explicitly required by ISO 32000-1.

NOTE 1 The proper mechanism by which a file can presumptively identify itself as being a PDF file of a given version level is described in 7.5.2, "File Header".

2.2 Conforming readers

A conforming reader shall comply with all requirements regarding reader functional behaviour specified in ISO 32000-1. The requirements of ISO 32000-1 with respect to reader behaviour are stated in terms of general functional requirements applicable to all conforming readers. ISO 32000-1 does not prescribe any specific technical design, user interface or implementation details of conforming readers. The rendering of conforming files shall be performed as defined by ISO 32000-1.

2.3 Conforming writers

A conforming writer shall comply with all requirements regarding the creation of PDF files as specified in ISO 32000-1. The requirements of ISO 32000-1 with respect to writer behaviour are stated in terms of general functional requirements applicable to all conforming writers and focus on the creation of conforming files. ISO 32000-1 does not prescribe any specific technical design, user interface or implementation details of conforming writers.

2.4 Conforming products

A conforming product shall comply with all requirements regarding the creation of PDF files as specified in ISO 32000-1 as well as comply with all requirements regarding reader functional behavior specified in ISO 32000-1.

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 639-1:2002, *Codes for the representation of names of languages -- Part 1: Alpha-2 code.*

ISO 639-2:1998, *Codes for the representation of names of languages -- Part 2: Alpha-3 code.*

ISO 3166-1:2006, *Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes.*

ISO 3166-2:1998, *Codes for the representation of names of countries and their subdivisions -- Part 2: Country subdivision code.*

ISO/IEC 8824-1:2002, *Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

ISO/IEC 10918-1:1994, *Digital Compression and Coding of Continuous-Tone Still Images* (informally known as the JPEG standard, for the Joint Photographic Experts Group, the ISO group that developed the standard).

ISO/IEC 15444-2:2004, *Information Technology—JPEG 2000 Image Coding System: Extensions.*

ISO/IEC 11544:1993/Cor 2:2001, *Information technology—Coded representation of picture and audio information—Progressive bi-level image compression* (JBIG2).

IEC/3WD 61966-2.1:1999, *Colour Measurement and Management in Multimedia Systems and Equipment, Part 2.1: Default RGB Colour Space—sRGB.*

ISO 15076-1:2005, *Image technology colour management - Architecture, profile format and data structure - Part 1:Based on ICC.1:2004-10.*

ISO 10646:2003, Information technology -- Universal Multiple-Octet Coded Character Set (UCS).

ISO/IEC 9541-1:1991, Information technology -- Font information interchange -- Part 1: Architecture.

ANSI X3.4-1986, Information Systems - Coded Sets 7-Bit American National Standard Code for Information Interchange (7-bit ASCII).

NOTE 1 The following documents can be found at AIIM at http://www.aiim.org/pdfrefdocs as well as at the Adobe Systems Incorporated Web Site http://www.adobe.com/go/pdf_ref_bibliography.

PDF Reference, Version 1.7, – 5th ed., (ISBN 0-321-30474-8), Adobe Systems Incorporated.

JavaScript for Acrobat API Reference, Version 8.0, (April 2007), Adobe Systems Incorporated.

Acrobat 3D JavaScript Reference, (April 2007), Adobe Systems Incorporated.

Adobe Glyph List, Version 2.0, (September 2002), Adobe Systems Incorporated.

OPI: Open Prepress Interface Specification 1.3, (September 1993), Adobe Systems Incorporated.

PDF Signature Build Dictionary Specification v.1.4, (March 2008), Adobe Systems Incorporated.

Adobe XML Architecture, Forms Architecture (XFA) Specification, version 2.5, (June 2007), Adobe Systems Incorporated.

Adobe XML Architecture, Forms Architecture (XFA) Specification, version 2.4, (September 2006), Adobe Systems Incorporated.

Adobe XML Architecture, Forms Architecture (XFA) Specification, version 2.2, (June 2005), Adobe Systems Incorporated.

Adobe XML Architecture, Forms Architecture (XFA) Specification, version 2.0, (October 2003), Adobe Systems Incorporated.

NOTE 2 Beginning with XFA 2.2, the XFA specification includes the Template Specification, the Config Specification, the XDP Specification, and all other XML specifications unique to the XML Forms Architecture (XFA).

Adobe XML Architecture, XML Data Package (XDP) Specification, version 2.0, (October 2003), Adobe Systems Incorporated.

Adobe XML Architecture, Template Specification, version 2.0, (October 2003), Adobe Systems Incorporated.

XML Forms Data Format Specification, version 2.0, (September 2007), Adobe Systems Incorporated.

XMP: Extensible Metadata Platform, (September 2005), Adobe Systems Incorporated.

TIFF Revision 6.0, Final, (June 1992), Adobe Systems Incorporated.

NOTE 3 The following Adobe Technical Notes can be found at the AIIM website at http://www.aiim.org/pdfnotes as well as at the Adobe Systems Incorporated Web Site (http://www.adobe.com) using the general search facility, entering the Technical Note number.

Technical Note #5004, Adobe Font Metrics File Format Specification, Version 4.1, (October 1998), Adobe Systems Incorporated.

NOTE 4 Adobe font metrics (AFM) files are available through the Type section of the ASN Web site.

Technical Note #5014, Adobe CMap and CID Font Files Specification, Version 1.0, (June 1993), Adobe Systems Incorporated.

Technical Note #5015, Type 1 Font Format Supplement, (May 1994), Adobe Systems Incorporated.

Technical Note #5078, Adobe-Japan1-4 Character Collection for CID-Keyed Fonts, (June 2004), Adobe Systems Incorporated.

Technical Note #5079, Adobe-GB1-4 Character Collection for CID-Keyed Fonts, (November 2000), Adobe Systems Incorporated.

Technical Note #5080, Adobe-CNS1-4 Character Collection for CID-Keyed Fonts, (May 2003), Adobe Systems Incorporated.

Technical Note #5087, Multiple Master Font Programs for the Macintosh, (February 1992), Adobe Systems Incorporated.

Technical Note #5088, Font Naming Issues, (April 1993), Adobe Systems Incorporated.

Technical Note #5092, CID-Keyed Font Technology Overview, (September 1994), Adobe Systems Incorporated.

Technical Note #5093, Adobe-Korea1-2 Character Collection for CID-Keyed Fonts, (May 2003), Adobe Systems Incorporated.

Technical Note #5094, Adobe CJKV Character Collections and CMaps for CID-Keyed Fonts, (June 2004), Adobe Systems Incorporated.

Technical Note #5097, Adobe-Japan2-0 Character Collection for CID-Keyed Fonts, (May 2003), Adobe Systems Incorporated.

Technical Note #5116, Supporting the DCT Filters in PostScript Level 2, (November 1992), Adobe Systems Incorporated.

Technical Note #5176, The Compact Font Format Specification, version 1.0, (December 2003), Adobe Systems Incorporated.

Technical Note #5177, The Type 2 Charstring Format, (December 2003), Adobe Systems Incorporated.

Technical Note #5411, ToUnicode Mapping File Tutorial, (May 2003), Adobe Systems Incorporated.

Technical Note #5620, Portable Job Ticket Format, Version 1.1, (April 1999), Adobe Systems Incorporated.

Technical Note #5660, Open Prepress Interface (OPI) Specification, Version 2.0, (January 2000), Adobe Systems Incorporated.

NOTE 5 The following documents are available as Federal Information Processing Standards Publications.

FIPS PUB 186-2, Digital Signature Standard, describes DSA signatures, (January 2000), Federal Information Processing Standards.

FIPS PUB 197, Advanced Encryption Standard (AES), (November 2001), Federal Information Processing Standards.

NOTE 6 The following documents are available as Internet Engineering Task Force RFCs.

RFC 1321, The MD5 Message-Digest Algorithm, (April 1992), Internet Engineering Task Force (IETF).

RFC 1738, Uniform Resource Locators, (December 1994), Internet Engineering Task Force (IETF).

RFC 1808, Relative Uniform Resource Locators, (June 1995), Internet Engineering Task Force (IETF).

RFC 1950, ZLIB Compressed Data Format Specification, Version 3.3, (May 1996), Internet Engineering Task Force (IETF).

RFC 1951, DEFLATE Compressed Data Format Specification, Version 1.3, (May 1996), Internet Engineering Task Force (IETF).

RFC 2045, Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, (November 1996), Internet Engineering Task Force (IETF).

RFC 2046, Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, (November 1996), Internet Engineering Task Force (IETF).

RFC 2083, PNG (Portable Network Graphics) Specification, Version 1.0, (March 1997), Internet Engineering Task Force (IETF).

RFC 2315, PKCS #7: Cryptographic Message Syntax, Version 1.5, (March 1998), Internet Engineering Task Force (IETF).

RFC 2396, Uniform Resource Identifiers (URI): Generic Syntax, (August 1998), Internet Engineering Task Force (IETF).

RFC 2560, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol—OCSP, (June 1999), Internet Engineering Task Force (IETF).

RFC 2616, Hypertext Transfer Protocol—HTTP/1.1, (June 1999), Internet Engineering Task Force (IETF).

RFC 2898, PKCS #5: Password-Based Cryptography Specification Version 2.0, (September 2000), Internet Engineering Task Force (IETF).

RFC 3066, Tags for the Identification of Languages, (January 2001), Internet Engineering Task Force (IETF).

RFC 3161, Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP), (August 2001), Internet Engineering Task Force (IETF).

RFC 3174, US Secure Hash Algorithm 1 (SHA1), (September 2001), Internet Engineering Task Force (IETF).

RFC 3280, Internet X.509 Public Key Infrastructure, Certificate and Certificate Revocation List (CRL) Profile, (April 2002), Internet Engineering Task Force (IETF).

NOTE 7 The following documents are available from other sources.

Adobe Type 1 Font Format., Version 1.1, (February 1993), Addison-Wesley, ISBN 0-201-57044-0.

OpenType Font Specification 1.4, December 2004, Microsoft.

*TrueType Reference Manual, (*December 2002), Apple Computer, Inc.

Standard ECMA-363, Universal 3D File Format, 1st Edition (U3D), (December 2004), Ecma International.

PANOSE Classification Metrics Guide, (February 1997), Hewlett-Packard Corporation.

ICC Characterization Data Registry, International Color Consortium (ICC).

Recommendations T.4 and T.6, Group 3 and Group 4 facsimile encoding, International Telecommunication Union (ITU).

TrueType 1.0 Font Files Technical Specification, Microsoft Corporation.

Client-Side JavaScript Reference, (May 1999), Mozilla Foundation.

The Unicode Standard, Version 4.0, Addison-Wesley, Boston, MA, 2003, Unicode Consortium.

Unicode Standard Annex #9, The Bidirectional Algorithm, Version 4.0.0, (April 2003), Unicode Consortium.

Unicode Standard Annex #14, Line Breaking Properties, Version 4.0.0, (April 2003), Unicode Consortium.

Unicode Standard Annex #29, Text Boundaries, Version 4.0.0, (March 2005), Unicode Consortium.

Extensible Markup Language (XML) 1.1, World Wide Web Consortium (W3C).

4 Terms and definitions

For the purposes of this document, these terms and definitions apply.

4.1
… (ellipsis)
An ellipsis is used within PDF examples to indicate omitted detail. Pairs of ellipses are also used to bracket comments, in italic, about such omitted detail.

4.2
8-bit value
(see byte)

4.3
array object
a one-dimensional collection of objects arranged sequentially and implicitly numbered starting at 0

4.4
ASCII
the American Standard Code for Information Interchange, a widely used convention for encoding a specific set of 128 characters as binary numbers defined in ANSI X3.4-1986

4.5
binary data
an ordered sequence of bytes

4.6
boolean objects
either the keyword **true** or the keyword **false**

4.7
byte
a group of 8 binary digits which collectively can be configured to represent one of 256 different values and various realizations of the 8 binary digits are widely used in today's electronic equipment

4.8
catalog
the primary dictionary object containing references directly or indirectly to all other objects in the document with the exception that there may be objects in the **trailer** that are not referred to by the **catalog**

4.9
character
numeric code representing an abstract symbol according to some defined character encoding rule

NOTE 1 There are three manifestations of characters in PDF, depending on context:

- A PDF file is represented as a sequence of 8-bit bytes, some of which are interpreted as character codes in the ASCII character set and some of which are treated as arbitrary binary data depending upon the context.
- The contents (data) of a string or stream object in some contexts are interpreted as character codes in the PDFDocEncoding or UTF-16 character set.
- The contents of a string within a PDF content stream in some situations are interpreted as character codes that select glyphs to be drawn on the page according to a character encoding that is associated with the text font.

4.10
character set
a defined set of symbols each assigned a unique character value

4.11
conforming reader
software application that is able to read and process PDF files that have been made in conformance with this specification and that itself conforms to requirements of conforming readers specified here [ISO 32000-1]

4.12
conforming product
software application that is both a conforming reader and a conforming writer

4.13
conforming writer
software application that is able to write PDF files that conform to this specification [ISO 32000-1]

4.14
content stream
stream object whose data consists of a sequence of instructions describing the graphical elements to be painted on a page

4.15
cross reference table
data structure that contains the byte offset start for each of the indirect objects within the file

4.16
developer
Any entity, including individuals, companies, non-profits, standards bodies, open source groups, etc., who are developing standards or software to use and extend ISO 32000-1.

4.17
dictionary object
an associative table containing pairs of objects, the first object being a name object serving as the key and the second object serving as the value and may be any kind of object including another dictionary

4.18
direct object
any object that has not been made into an indirect object

4.19
electronic document
electronic representation of a page-oriented aggregation of text, image and graphic data, and metadata useful to identify, understand and render that data, that can be reproduced on paper or displayed without significant loss of its information content

4.20
end-of-line marker (EOL marker)
one or two character sequence marking the end of a line of text, consisting of a CARRIAGE RETURN character (0Dh) or a LINE FEED character (0Ah) or a CARRIAGE RETURN followed immediately by a LINE FEED

4.21
FDF file
File conforming to the Forms Data Format containing form data or annotations that may be imported into a PDF file (see 12.7.7, “Forms Data Format”)

4.22
filter
an optional part of the specification of a stream object, indicating how the data in the stream should be decoded before it is used

4.23
font
identified collection of graphics that may be glyphs or other graphic elements [ISO 15930-4]

4.24
function
a special type of object that represents parameterized classes, including mathematical formulas and sampled representations with arbitrary resolution

4.25
glyph
recognizable abstract graphic symbol that is independent of any specific design [ISO/IEC 9541-1]

4.26
graphic state
the top of a push down stack of the graphics control parameters that define the current global framework within which the graphics operators execute

4.27
ICC profile
colour profile conforming to the ICC specification [ISO 15076-1:2005]

4.28
indirect object
an object that is labeled with a positive integer object number followed by a non-negative integer generation number followed by **obj** and having **endobj** after it

4.29
integer object
mathematical integers with an implementation specified interval centered at 0 and written as one or more decimal digits optionally preceded by a sign

4.30
name object
an atomic symbol uniquely defined by a sequence of characters introduced by a SOLIDUS (/), (2Fh) but the SOLIDUS is not considered to be part of the name

4.31
name tree
similar to a dictionary that associates keys and values but the keys in a name tree are strings and are ordered

4.32
null object
a single object of type null, denoted by the keyword **null**, and having a type and value that are unequal to those of any other object

4.33
number tree
similar to a dictionary that associates keys and values but the keys in a number tree are integers and are ordered

4.34
numeric object
either an integer object or a real object

4.35
object
a basic data structure from which PDF files are constructed and includes these types: array, Boolean, dictionary, integer, name, null, real, stream and string

4.36
object reference
an object value used to allow one object to refer to another; that has the form “<n> <m> R” where <n> is an indirect object number, <m> is its version number and R is the uppercase letter R

4.37
object stream
a stream that contains a sequence of PDF objects

4.38
PDF
Portable Document Format file format defined by this specification [ISO 32000-1]

4.39
real object
approximate mathematical real numbers, but with limited range and precision and written as one or more decimal digits with an optional sign and a leading, trailing, or embedded PERIOD (2Eh) (decimal point)

4.40
rectangle
a specific array object used to describe locations on a page and bounding boxes for a variety of objects and written as an array of four numbers giving the coordinates of a pair of diagonally opposite corners, typically in the form [ll_x ll_y ur_x ur_y] specifying the lower-left x, lower-left y, upper-right x, and upper-right y coordinates of the rectangle, in that order

4.41
resource dictionary
associates resource names, used in content streams, with the resource objects themselves and organized into various categories (e.g., Font, ColorSpace, Pattern)

4.42
space character
text string character used to represent orthographic white space in text strings

NOTE 2 space characters include HORIZONTAL TAB (U+0009), LINE FEED (U+000A), VERTICAL TAB (U+000B), FORM FEED (U+000C), CARRIAGE RETURN (U+000D), SPACE (U+0020), NOBREAK SPACE (U+00A0), EN SPACE (U+2002), EM SPACE (U+2003), FIGURE SPACE (U+2007), PUNCTUATION SPACE (U+2008), THIN SPACE (U+2009), HAIR SPACE (U+200A), ZERO WIDTH SPACE (U+200B), and IDEOGRAPHIC SPACE (U+3000)

4.43
stream object
consists of a dictionary followed by zero or more bytes bracketed between the keywords stream and endstream

4.44
string object
consists of a series of bytes (unsigned integer values in the range 0 to 255) and the bytes are not integer objects, but are stored in a more compact form

4.45
web capture
refers to the process of creating PDF content by importing and possibly converting internet-based or locally-resident files. The files being imported may be any arbitrary format, such as HTML, GIF, JPEG, text, and PDF

4.46
white-space character
characters that separate PDF syntactic constructs such as names and numbers from each other; white space characters are HORIZONTAL TAB (09h), LINE FEED (0Ah), FORM FEED (0Ch), CARRIAGE RETURN (0Dh), SPACE (20h); (see Table 1 in 7.2.2, “Character Set”)

4.47
XFDF file
file conforming to the XML Forms Data Format 2.0 specification, which is an XML transliteration of Forms Data Format (FDF)

4.48
XMP packet
structured wrapper for serialized XML metadata that can be embedded in a wide variety of file formats

5 Notation

PDF operators, PDF keywords, the names of keys in PDF dictionaries, and other predefined names are written in bold sans serif font; words that denote operands of PDF operators or values of dictionary keys are written in italic sans serif font.

Token characters used to delimit objects and describe the structure of PDF files, as defined in 7.2, "Lexical Conventions", may be identified by their ANSI X3.4-1986 (ASCII 7-bit USA codes) character name written in upper case in bold sans serif font followed by a parenthetic two digit hexadecimal character value with the suffix “h”.

Characters in text streams, as defined by 7.9.2, "String Object Types", may be identified by their ANSI X3.4-1986 (ASCII 7-bit USA codes) character name written in uppercase in sans serif font followed by a parenthetic four digit hexadecimal character code value with the prefix “U+” as shown in EXAMPLE 1 in this clause.

EXAMPLE 1 **EN SPACE** (U+2002).

6 Version Designations

For the convenience of the reader, the PDF versions in which various features were introduced are provided informatively within this document. The first version of PDF was designated PDF 1.0 and was specified by Adobe Systems Incorporated in the PDF Reference 1.0 document published by Adobe and Addison Wesley. Since then, PDF has gone through seven revisions designated as: PDF 1.1, PDF 1.2, PDF 1.3, PDF 1.4, PDF 1.5, PDF 1.6 and PDF 1.7. All non-deprecated features defined in a previous PDF version were also included in the subsequent PDF version. Since ISO 32000-1 is a PDF version matching PDF 1.7, it is also suitable for interpretation of files made to conform with any of the PDF specifications 1.0 through 1.7. Throughout this specification in order to indicate at which point in the sequence of versions a feature was introduced, a notation with a PDF version number in parenthesis (e.g., *(PDF 1.3)*) is used. Thus if a feature is labelled with *(PDF 1.3)* it means that PDF 1.0, PDF 1.1 and PDF 1.2 were not specified to support this feature whereas all versions of PDF 1.3 and greater were defined to support it.

7 Syntax

7.1 General

This clause covers everything about the syntax of PDF at the object, file, and document level. It sets the stage for subsequent clauses, which describe how the contents of a PDF file are interpreted as page descriptions, interactive navigational aids, and application-level logical structure.

PDF syntax is best understood by considering it as four parts, as shown in Figure 1:

- *Objects*. A PDF document is a data structure composed from a small set of basic types of data objects. Sub-clause 7.2, "Lexical Conventions," describes the character set used to write objects and other syntactic elements. Sub-clause 7.3, "Objects," describes the syntax and essential properties of the objects. Sub-clause 7.3.8, "Stream Objects," provides complete details of the most complex data type, the stream object.

- *File structure*. The PDF file structure determines how objects are stored in a PDF file, how they are accessed, and how they are updated. This structure is independent of the semantics of the objects. Sub-clause 7.5, "File Structure," describes the file structure. Sub-clause 7.6, "Encryption," describes a file-level mechanism for protecting a document's contents from unauthorized access.

- *Document structure*. The PDF document structure specifies how the basic object types are used to represent components of a PDF document: pages, fonts, annotations, and so forth. Sub-clause 7.7, "Document Structure," describes the overall document structure; later clauses address the detailed semantics of the components.

- *Content streams*. A PDF *content stream* contains a sequence of instructions describing the appearance of a page or other graphical entity. These instructions, while also represented as objects, are conceptually distinct from the objects that represent the document structure and are described separately. Sub-clause 7.8, "Content Streams and Resources," discusses PDF content streams and their associated resources.

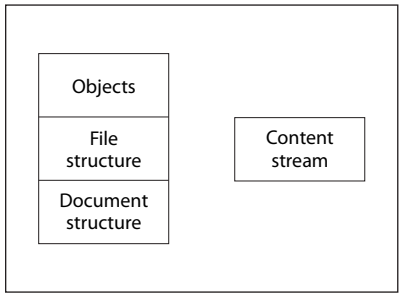

Figure 1 – PDF Components

In addition, this clause describes some data structures, built from basic objects, that are so widely used that they can almost be considered basic object types in their own right. These objects are covered in: 7.9, "Common Data Structures"; 7.10, "Functions"; and 7.11, "File Specifications."

NOTE Variants of PDF's object and file syntax are also used as the basis for other file formats. These include the Forms Data Format (FDF), described in 12.7.7, "Forms Data Format", and the Portable Job Ticket Format (PJTF), described in Adobe Technical Note #5620, *Portable Job Ticket Format*.

7.2 Lexical Conventions

7.2.1 General

At the most fundamental level, a PDF file is a sequence of bytes. These bytes can be grouped into *tokens* according to the syntax rules described in this sub-clause. One or more tokens are assembled to form higher-

level syntactic entities, principally *objects*, which are the basic data values from which a PDF document is constructed.

A non-encrypted PDF can be entirely represented using byte values corresponding to the visible printable subset of the character set defined in ANSI X3.4-1986, plus white space characters. However, a PDF file is not restricted to the ASCII character set; it may contain arbitrary bytes, subject to the following considerations:

- The tokens that delimit objects and that describe the structure of a PDF file shall use the ASCII character set. In addition all the reserved words and the names used as keys in PDF standard dictionaries and certain types of arrays shall be defined using the ASCII character set.

- The data values of strings and streams objects may be written either entirely using the ASCII character set or entirely in binary data. In actual practice, data that is naturally binary, such as sampled images, is usually represented in binary for compactness and efficiency.

- A PDF file containing binary data shall be transported as a binary file rather than as a text file to insure that all bytes of the file are faithfully preserved.

NOTE 1 A binary file is not portable to environments that impose reserved character codes, maximum line lengths, end-of-line conventions, or other restrictions

NOTE 2 In this clause, the usage of the term character is entirely independent of any logical meaning that the value may have when it is treated as data in specific contexts, such as representing human-readable text or selecting a glyph from a font.

7.2.2 Character Set

The PDF character set is divided into three classes, called *regular*, *delimiter*, and *white-space* characters. This classification determines the grouping of characters into tokens. The rules defined in this sub-clause apply to all characters in the file except within strings, streams, and comments.

The *White-space characters* shown in Table 1 separate syntactic constructs such as names and numbers from each other. All white-space characters are equivalent, except in comments, strings, and streams. In all other contexts, PDF treats any sequence of consecutive white-space characters as one character.

Table 1 – White-space characters

Decimal	Hexadecimal	Octal	Name
0	00	000	Null (NUL)
9	09	011	HORIZONTAL TAB (HT)
10	0A	012	LINE FEED (LF)
12	0C	014	FORM FEED (FF)
13	0D	015	CARRIAGE RETURN (CR)
32	20	040	SPACE (SP)

The CARRIAGE RETURN (0Dh) and LINE FEED (0Ah) characters, also called *newline characters*, shall be treated as *end-of-line* (EOL) markers. The combination of a CARRIAGE RETURN followed immediately by a LINE FEED shall be treated as one EOL marker. EOL markers may be treated the same as any other white-space characters. However, sometimes an EOL marker is required or recommended—that is, preceding a token that must appear at the beginning of a line.

NOTE The examples in this standard use a convention that arranges tokens into lines. However, the examples' use of white space for indentation is purely for clarity of exposition and need not be included in practical use.

The *delimiter characters* (,), <, >, [,], {, }, /, and % are special (LEFT PARENTHESIS (28h), RIGHT PARENTHESIS (29h), LESS-THAN SIGN (3Ch), GREATER-THAN SIGN (3Eh), LEFT SQUARE BRACKET (5Bh), RIGHT SQUARE BRACKET (5Dh), LEFT CURLY BRACE (7Bh), RIGHT CURLY BRACE (07Dh), SOLIDUS (2Fh) and PERCENT SIGN (25h), respectively). They delimit syntactic entities such as arrays, names, and comments. Any of these characters terminates the entity preceding it and is not included in the entity. Delimiter characters are allowed within the scope of a string when following the rules for composing strings; see 7.3.4.2, “Literal Strings”. The leading (of a string does delimit a preceding entity and the closing) of a string delimits the string’s end.

Table 2 – Delimiter characters

Glyph	Decimal	Hexadecimal	Octal	Name
(	40	28	50	LEFT PARENTHESIS
)	41	29	51	RIGHT PARENTHESIS
<	60	3C	60	LESS-THAN SIGN
>	62	3E	62	GREATER-THAN SIGN
[	91	5B	133	LEFT SQUARE BRACKET
]	93	5D	135	RIGHT SQUARE BRACKET
{	123	7B	173	LEFT CURLY BRACKET
}	125	7D	175	RIGHT CURLY BRACKET
/	47	2F	57	SOLIDUS
%	37	25	45	PERCENT SIGN

All characters except the white-space characters and delimiters are referred to as *regular characters*. These characters include bytes that are outside the ASCII character set. A sequence of consecutive regular characters comprises a single token. PDF is case-sensitive; corresponding uppercase and lowercase letters shall be considered distinct.

7.2.3 Comments

Any occurrence of the PERCENT SIGN (25h) outside a string or stream introduces a *comment*. The comment consists of all characters after the PERCENT SIGN and up to but not including the end of the line, including regular, delimiter, SPACE (20h), and HORZONTAL TAB characters (09h). A conforming reader shall ignore comments, and treat them as single white-space characters. That is, a comment separates the token preceding it from the one following it.

EXAMPLE The PDF fragment in this example is syntactically equivalent to just the tokens abc and 123.

```
abc% comment ( /%) blah blah blah
123
```

Comments (other than the %PDF–*n.m* and %%EOF comments described in 7.5, "File Structure") have no semantics. They are not necessarily preserved by applications that edit PDF files.

7.3 Objects

7.3.1 General

PDF includes eight basic types of objects: Boolean values, Integer and Real numbers, Strings, Names, Arrays, Dictionaries, Streams, and the null object.

Objects may be labelled so that they can be referred to by other objects. A labelled object is called an indirect object (see 7.3.10, "Indirect Objects").

Each object type, their method of creation and their proper referencing as indirect objects is described in 7.3.2, "Boolean Objects" through 7.3.10, "Indirect Objects."

7.3.2 Boolean Objects

Boolean objects represent the logical values of true and false. They appear in PDF files using the keywords **true** and **false**.

7.3.3 Numeric Objects

PDF provides two types of numeric objects: integer and real. *Integer objects* represent mathematical integers. *Real objects* represent mathematical real numbers. The range and precision of numbers may be limited by the internal representations used in the computer on which the conforming reader is running; Annex C gives these limits for typical implementations.

An integer shall be written as one or more decimal digits optionally preceded by a sign. The value shall be interpreted as a signed decimal integer and shall be converted to an integer object.

EXAMPLE 1 Integer objects

123 43445 +17 -98 0

A real value shall be written as one or more decimal digits with an optional sign and a leading, trailing, or embedded PERIOD (2Eh) (decimal point). The value shall be interpreted as a real number and shall be converted to a real object.

EXAMPLE 2 Real objects

34.5 -3.62 +123.6 4. -.002 0.0

NOTE 1 A conforming writer shall not use the PostScript syntax for numbers with non-decimal radices (such as 16#FFFE) or in exponential format (such as 6.02E23).

NOTE 2 Throughout this standard, the term *number* refers to an object whose type may be either integer or real. Wherever a real number is expected, an integer may be used instead. For example, it is not necessary to write the number 1.0 in real format; the integer 1 is sufficient.

7.3.4 String Objects

7.3.4.1 General

A *string object* shall consist of a series of zero or more bytes. String objects are not integer objects, but are stored in a more compact format. The length of a string may be subject to implementation limits; see Annex C.

String objects shall be written in one of the following two ways:

- As a sequence of literal characters enclosed in parentheses ( ) (using LEFT PARENTHESIS (28h) and RIGHT PARENThESIS (29h)); see 7.3.4.2, "Literal Strings."
- As hexadecimal data enclosed in angle brackets < > (using LESS-THAN SIGN (3Ch) and GREATER-THAN SIGN (3Eh)); see 7.3.4.3, "Hexadecimal Strings."

NOTE In many contexts, conventions exist for the interpretation of the contents of a string value. This sub-clause defines only the basic syntax for writing a string as a sequence of bytes; conventions or rules governing the contents of strings in particular contexts are described with the definition of those particular contexts.

7.9.2, "String Object Types," describes the encoding schemes used for the contents of string objects.

7.3.4.2 Literal Strings

A *literal string* shall be written as an arbitrary number of characters enclosed in parentheses. Any characters may appear in a string except unbalanced parentheses (LEFT PARENHESIS (28h) and RIGHT PARENTHESIS (29h)) and the backslash (REVERSE SOLIDUS (5Ch)), which shall be treated specially as described in this sub-clause. Balanced pairs of parentheses within a string require no special treatment.

EXAMPLE 1 The following are valid literal strings:

```
(This is a string)
(Strings may contain newlines
and such.)
(Strings may contain balanced parentheses ( ) and
special characters (*!&}^% and so on).)
(The following is an empty string.)
()
(It has zero (0) length.)
```

Within a literal string, the REVERSE SOLIDUS is used as an escape character. The character immediately following the REVERSE SOLIDUS determines its precise interpretation as shown in Table 3. If the character following the REVERSE SOLIDUS is not one of those shown in Table 3, the REVERSE SOLIDUS shall be ignored.

Table 3 – Escape sequences in literal strings

Sequence	Meaning
\n	LINE FEED (0Ah) (LF)
\r	CARRIAGE RETURN (0Dh) (CR)
\t	HORIZONTAL TAB (09h) (HT)
\b	BACKSPACE (08h) (BS)
\f	FORM FEED (FF)
\(	LEFT PARENTHESIS (28h)
\)	RIGHT PARENTHESIS (29h)
\\	REVERSE SOLIDUS (5Ch) (Backslash)
ddd	Character code *ddd* (octal)

A conforming writer may split a literal string across multiple lines. The REVERSE SOLIDUS (5Ch) (backslash character) at the end of a line shall be used to indicate that the string continues on the following line. A conforming reader shall disregard the REVERSE SOLIDUS and the end-of-line marker following it when reading the string; the resulting string value shall be identical to that which would be read if the string were not split.

EXAMPLE 2 (These \

```
two strings \
are the same.)
(These two strings are the same.)
```

An end-of-line marker appearing within a literal string without a preceding REVERSE SOLIDUS shall be treated as a byte value of (0Ah), irrespective of whether the end-of-line marker was a CARRIAGE RETURN (0Dh), a LINE FEED (0Ah), or both.

EXAMPLE 3
```
(This string has an end-of-line at the end of it.
)
(So does this one.\n)
```

The *ddd* escape sequence provides a way to represent characters outside the printable ASCII character set.

EXAMPLE 4
```
(This string contains \245two octal characters\307.)
```

The number *ddd* may consist of one, two, or three octal digits; high-order overflow shall be ignored. Three octal digits shall be used, with leading zeros as needed, if the next character of the string is also a digit.

EXAMPLE 5
```
the literal
    (\0053)
denotes a string containing two characters, \005 (Control-E) followed by the digit 3, whereas both
    (\053)
and
    (\53)
denote strings containing the single character \053, a plus sign (+).
```

Since any 8-bit value may appear in a string (with proper escaping for REVERSE SOLIDUS (backslash) and unbalanced PARENTHESES) this \ddd notation provides a way to specify characters outside the ASCII character set by using ASCII characters only. However, any 8-bit value may appear in a string, represented either as itself or with the \ddd notation described.

When a document is encrypted (see 7.6, “Encryption”), all of its strings are encrypted; the encrypted string values contain arbitrary 8-bit values. When writing encrypted strings using the literal string form, the conforming writer shall follow the rules described. That is, the REVERSE SOLIDUS character shall be used as an escape to specify unbalanced PARENTHESES or the REVERSE SOLIDUS character itself. The REVERSE SOLIDUS may, but is not required, to be used to specify other, arbitrary 8-bit values.

7.3.4.3 Hexadecimal Strings

Strings may also be written in hexadecimal form, which is useful for including arbitrary binary data in a PDF file. A hexadecimal string shall be written as a sequence of hexadecimal digits (0–9 and either A–F or a–f) encoded as ASCII characters and enclosed within angle brackets (using LESS-THAN SIGN (3Ch) and GREATER-THAN SIGN (3Eh)).

EXAMPLE 1
```
<4E6F762073686D6F7A206B6120706F702E>
```

Each pair of hexadecimal digits defines one byte of the string. White-space characters (such as SPACE (20h), HORIZONTAL TAB (09h), CARRIAGE RETURN (0Dh), LINE FEED (0Ah), and FORM FEED (0Ch)) shall be ignored.

If the final digit of a hexadecimal string is missing—that is, if there is an odd number of digits—the final digit shall be assumed to be 0.

EXAMPLE 2
```
<901FA3>
is a 3-byte string consisting of the characters whose hexadecimal codes are 90, 1F, and A3, but
<901FA>
is a 3-byte string containing the characters whose hexadecimal codes are 90, 1F, and A0.
```

7.3.5 Name Objects

Beginning with PDF 1.2 a *name object* is an atomic symbol uniquely defined by a sequence of any characters (8-bit values) except null (character code 0). *Uniquely defined* means that any two name objects made up of the same sequence of characters denote the same object. *Atomic* means that a name has no internal structure; although it is defined by a sequence of characters, those characters are not considered elements of the name.

