Datalogics

A Developer's Introduction to the PDF File Format

Matt Kuznicki
Chief Technical Officer
Datalogics

Overview

- A Brief History of PDF
- PDF File Format Overview
- Use Cases for PDF Files
- PDF File Problem Areas
- PDF Tools and Toolkits
- Summary

A Brief History of PDF

But First… What IS a PDF File?

A PDF file is a portable representation of a visible series of pages, and information about the pages

- Intended to represent documents that will be, at some time, viewed by one or more people
- Consist primarily of one or more pages – each page has a sequence of ordered operators that place marks on the page or modify how subsequent operators will be interpreted
- Information about how to interpret these operators is included for reliable display and processing of the PDF pages

Key Features of PDF Files

Advantages of PDF over other image formats:

- Contents can be a mix of raster and vector formats
- Multiple color spaces, resolutions and compression formats are supported for different objects
- Multiple uses of the same graphic data at different resolutions, scaling and rotation factors allowed
- Support for masking and transparency

Key Features of PDF Files

Advantages of PDF over other document formats:

- Format is open, completely documented and freely implementable by anyone
- PDF format can capture both how a document should appear, and the semantic content of the document
- Capable of containing revisions and several versions of contents
- Several levels of document password protection
- Digital signatures and change detection built-in
- Suitable for archiving, user presentation as well as in workflow steps

A Brief History of PDF Files

PDF – Portable Document Format – files were created by Adobe Systems

- 1993: first PDF file viewer released by Adobe Systems
- 1996: Adobe Reader released for free for viewing PDFs
- 2005: PDF Association vendor association is formed by interested PDF tool vendors and users
- 2008: PDF revision 1.7 released as international (ISO) standard – ISO 32000

PDF as defined in ISO 32000 may be freely implemented by anyone without need for patent license.

PDF File Format Overview

PDF File Format: Overview

PDF files are object-oriented – they consist of a series of numbered and versioned objects that define the elements in the PDF and references to these elements.

Additional information is used to specify:

- PDF format identifier
- The byte offset of every PDF object that can be referenced by another PDF object
- Byte offsets for each different version of every PDF object that has multiple versions
- Locations of the byte offset tables in the PDF file

PDF File Format: Non-Object Data

The non-object data in a PDF file provides the necessary glue to interpret the PDF as a series of objects:

- Object cross-reference (**xref**) information exists as a table of object number & byte position pairs
  - This can be in a simple table or in a compressed stream
- Multiple cross-reference tables or streams may exist within a PDF file
  - This supports different PDF file object versions (*incremental updates*) as well as rapid retrieval of the first visible page in the PDF (*linearization)*

PDF Object Basics

Objects in PDF files can be **direct** or **indirect** objects:

- Direct objects are defined entirely within the scope of an enclosing object
- Indirect objects are defined independently and outside of an enclosing object
- Both direct and indirect objects can reference other indirect objects
- Indirect objects may be versioned – multiple versions of the same object can exist in a PDF file

PDF Object Basics

There are eight types of objects in a PDF file:

- Boolean objects
- Integer and real numbers
- Strings
- Names
- Arrays
- Dictionaries
- Streams
- The *null* object

PDF Object Basics: Boolean Objects

A boolean object represents a value that is true or false. It is represented by the keywords **true** and **false** in the PDF file.

Boolean Object Examples	
Direct Object as Dict Value	Indirect Object

`/KeyName` **`true`**	`10 0 obj`
	`true`
	`endobj`

PDF Object Basics: Integer and Real Numbers

A number in a PDF file is a sequence of digits (0 – 9), potentially with a preceding sign (+/-) and a decimal (0x2E, "**.**") separating the whole-number and fractional portions

- Both integer and real numbers may be represented as signed or as unsigned values
- Some numeric values in PDF are specifically required to be integers, others may be real numbers
- Integer numbers are valid for any value that is defined as a real number, but not vice-versa
- Real number representation is not specified as any specific machine or programming language format
  - Different implementations may use different representations, e.g. 16.16 fixed-point integer or IEEE 754. This is an implementation decision.

PDF Object Basics: Integer and Real Numbers

Number Object Examples	
Direct Object as Dict Value	Indirect Object

`/IntegerReference 1000`
 	
`/RealReference -1234.5678`
	`10 0 obj`
	`1000`
	`endobj`
	
	`11 0 obj`
	`-1234.5678`
	`endobj`

PDF Object Basics: String Objects

Strings in PDF files may be represented in two forms:

- Literal byte values, delineated by "(" and ")"
- Hexadecimal digits 0-9 and A-F, delineated by "<" and ">"

Important differences in strings from other formats:

- Strings in PDF files may contain binary and arbitrary data, including line-feeds
- "Encoding" or interpretation of a PDF string's bytes depends on its context within the PDF file
- Strings may escape certain values with the backslash ("\") characters to denote specific sequences, e.g. to include "(" or ")" in a string

PDF Object Basics: String Objects

String Object Examples	
Direct Object as Dict Value	Indirect Object
/StringReference **(This is a string delimited by parentheses.)** /HexStringReference % "Hello World" **<48656C6C6F20576F726C64>**	10 0 obj **(This is a string delimited by parentheses.)** endobj 11 0 obj % "Hello World" **<48656C6C6F20576F726C64>** endobj

PDF Object Basics: Name Objects

Name objects are string-like sequences of characters that form named identifiers in PDF files

- Specified with a leading forward slash (“/”) and following characters
  - The number sign (#) is used to escape two-digit hexadecimal codes that cannot be represented by normal characters
- Usually interpreted in “PDF document encoding” (**PDFDocEncoding**)
  - In certain situations, interpreted as sequences of characters in UTF-8 byte encoding

PDF Object Basics: Name Objects

Name Object Examples	
Name References to Direct Objects	Name References to Indirect Objects
/NameExample (NameExample refers to this string in a dictionary) **/Hex#20Name#20Example** **/ReferencedName** % Both the name and its referent are direct name objects here	**/NameExample** 10 0 obj **/NameExample_v2** 10 1 obj

PDF Object Basics: Array Objects

Arrays in PDF files are ordered collections of other PDF objects

- Any combination of object types is allowed – not restricted to containing only one object type
- Direct objects and indirect object references are allowed
- Only one-dimensional arrays are supported
  - But a PDF array can itself contain a series of other arrays to simulate multi-dimensionality

PDF Object Basics: Array Objects

Array Object Examples	
Direct Object as Dict Value	Indirect Object
/ArrayReference **[150 true (this is an array with three direct objects)]** /ArrayExample2 **[150 true 1 0 obj]** % This is a three element array, where the 3rd element is a reference to indirect object #1 in the PDF file	10 0 obj **[150 true (this is an array with three direct objects)]** endobj 11 0 obj % An array of arrays **[[1 0 0] [0 1 0] [0 0 1]]** endobj

PDF Object Basics: Dictionary Objects

Dictionaries in PDF files are associative containers in **key** – **value** format

- Delineated by double angle brackets (<< and >>)
- Keys are always direct PDF name objects
- Multiple instances of the same key are not allowed in a PDF dictionary
- Values may be any object type, including other dictionaries
- Values may be direct objects or references to indirect objects

PDF Object Basics: Dictionary Objects

Dictionary Object Examples	
Direct Object as Dict Value	Indirect Object

/DictReference **<<**
  /Type /ExampleType
  /NumberExample 1.0
  /BooleanExample false
  /NestedDictExample <<
  /IsNested true
  >>
>>
	10 0 obj **<<**
	  /IsIndirectDict true
	  /ArrayExample [0 1 2]
	  /IndirectReference 11 0 obj
	>> endobj

PDF Object Basics: Stream Objects

A PDF file stream is a binary data stream and an accompanying dictionary that gives information about the data stream

- The dictionary defining characteristics of the data stream comes before the binary data stream
- Binary data stream is delimited by the **stream** and **endstream** keywords, along with line feed or CR/LF characters after the keywords
  - Alternately, data bytes may be stored in an external file
- Binary data may be encoded, for compression or other effects, and in multiple ways

Stream objects are always indirect objects

PDF Object Basics: Stream Objects

Stream Object Example

```
% The stream object is defined by the dictionary and the data
10 0 obj <<
  /Type /ExampleStreamType
  /Length 12 % Length key is required
>>
stream
Example Data
endstream
```

PDF Object Basics: the *null* object

There is only one *null* object in a PDF file; all references to the *null* object are equivalent

- References to the *null* object are made using the **null** keyword
- References to the *null* object denote an absence of value – they do not specify any actual value
- Null object references in value entries in dictionaries are equivalent to not specifying the key/value pair at all

References to indirect objects that don't exist are equivalent to references to the *null* object

PDF Object Basics

Dictionary and stream objects form the main building blocks of PDF files

- Elements in PDF files are built up from dictionaries or streams and the references that are in these dictionaries
- Many dictionary and stream objects can contain **Type** entries; the specific type a dictionary or stream represents can always be derived from the dictionary that is referencing it

Comments in PDF File Data

Comments are allowed at any time in a PDF file except within a string or a stream object

- Each comment is delimited via the percent sign (%) and continues to the end of the line (until a CR or LF) the comments starts on
- PDF processors are not required to preserve comments. Different implementations may choose to preserve or remove comments as desired

PDF Basic Building Blocks

Basic PDF objects are used to build the visible elements of a PDF:

- Content streams: represent ordered sequences of marking operators
- XObjects: reusable elements that can be referenced in different content streams. Include raster images and reusable PDF marking sequences
- Page objects: sequences of content streams that make the pages of a PDF file that a viewer will see and interact with

PDF Basic Building Blocks: Content Streams

Content streams are PDF stream objects containing marking operators:

- Positional operators change the current drawing point, scaling and direction
- Text blocks display text via font glyph references
- Vector elements can be used to draw lines, shapes and filled polygons
- Raster images may be specified directly in the content stream, or by reference to **Image** XObjects

PDF Basic Building Blocks: Content Streams

The *graphic state* contains the current color, drawing matrix, color space and other parameters

- Graphic state is reset for every new PDF page and for every annotation
- The graphic state is a stack. It's state can be saved (pushed) and that state restored (popped) to undo changes
- Graphic state changes are cumulative and inherited between different content streams on the same page
- Graphic state is inherited by reusable *form XObjects*. Any changes made and not popped off the stack are inherited by the calling content stream!

PDF Basic Building Blocks: XObjects

XObjects represent content that may be used by any content stream in a PDF file:

- Raster images represented as **Image** XObjects for efficiency and reusability across different content streams
- PDF content streams represented as **Form** XObjects for reusability in other content streams

Image and Form object references can modify the scaling, rotation and placement of these objects on a per-reference basis.

PDF Basic Building Blocks: PDF Pages

A **Page** object in a PDF file is a dictionary that contains information about the page to display:

- Page dimensions
- Actions to be performed when the page is shown or the display exits the page
- One or more content streams that are drawn, in order, to show the PDF page
- Named resources – fonts, XObjects, color spaces, etc. – required by the content streams
- Annotations that are drawn over the content streams

Other information is also present – this is just a basic overview

A Very Simple Example

Annotations and Markup

Annotations are content that are drawn on top of a given page. These allow others to easily add some content:

- Document editing marks (strikeouts, replacements, sticky notes)
- Additional content (text, shapes, highlights)
- Stamps (approval, rejection, etc.)
- Digital signatures

Information can be stored and extracted from annotations to drive business processes

User Forms and Workflows

PDF forms allow sending and retrieving information from users:

- Acrobat Forms (“AcroForm”) technology allows users to add information in a fixed-layout format and save this information in a PDF or send back to the sender
- XML forms architecture (“XFA”) technology uses PDF as a thin wrapper for dynamic document layout
- Both allow users to provide information in familiar interfaces that can drive business value chains

Acrobat form technology has much better support across multiple tools and platforms than XFA.

User Interactivity

PDF documents can include interactive elements:

- Audio, video, 3D drawings and other inclusions
- Specified actions on document open, page turns and other pre-specified triggers
- Creator-defined JavaScript to be executed on pre-specified triggers
- Links to external documents and resources

Flexibility comes with a price: security is an important consideration. Most tools restrict interactivity for safety.

Semantic Structuring and Metadata

PDF files can also contain information about what the contents actually mean:

- Document-level metadata: author, title, keywords, etc.
- Object-level metadata: EXIF information, author, copyright, etc.
- Application-specific data for transfer to/from authoring tools (e.g. Adobe Illustrator, InDesign)
- Semantic meaning for text blocks and alternate text for objects (Tagged PDF)

Wait! Doesn't a PDF Say What It Means?

There's can by differences between what a PDF *looks like* and the *actual content:*

- The letters that form text – words, sentences, paragraphs, lists, etc. – may be spread all across content streams
  - Same goes for other page content: the reading order may be quite different from the content stream marking operators
- The glyphs that text are in may be accidentally or intentionally obfuscated and hinder text extraction
- Text and other content may be invisible: drawn transparently or hidden by other content above
  - This is popular with PDFs created by OCR processes, to put invisible text behind the scanned image to make "searchable"

Use Cases for PDF Files

Best Use Cases for PDF Files

The PDF format is ideal for representing documents and packages of documents:

- Various standards (PDF/A, PDF/X, PDF/UA) define PDF syntax refinements for more reliable archiving, transmission and universal accessibility
- PDF format can easily accommodate graphics of different types, color spaces and resolutions; and text in various and different languages
- PDF format can be self-contained and stored without reliance on external files or resources

Documents vs. Publications

Documents are data designed to appear consistently across space, time and viewers

- Files where one does not want the contents to change in meaning because of layout or data source changes

Publications are data designed with the potential for change after a user receives these

- Website links may change in content, or may change significantly in visual presentation, depending on when retrieved and on what device

Page Oriented Visual Layout Fidelity

Reliable and predictable visual presentation is a PDF strength:

- Page orientation allows precise, intended placement of text and graphics with respect to other elements
- Ability to precisely state color space calibration allows for reliable interpretation of color
- Ability to store and preserve transparency information means elements do not need to be "flattened"

However, not all developer tools properly handle all PDF file format features!

Reliable Interchange

PDF is a strong format for reliable interchange of information between people and in workflows, across boundaries of space and time:

- Open standard allows for anyone to know how to interpret PDF data properly
- Unambiguous presentation allows for reliability and fidelity across viewers now and in the future
- PDF/A further refines these ideas and is specially intended for archival purposes for decades or centuries

Digitally Signed Documents

PDF format has native support for digital signatures:

- Public-key and digital certificate based. Several standards in certificates and hashing supported.
- Author signatures allow for the creator of a document to attest to their authorship; certifying signatures allow for a 3rd party (e.g. a corporation) to attest to the validity of a document
- Digital signatures allow for reliably detecting changes in a PDF from when it was signed (invalid document) and verifying the authority issuing the signing certificate
- Specific changes can be allowed to PDFs that do not invalidate the digital signatures

PDF File Problem Areas

Editing PDF Files

PDF files can be edited, but this was not a primary design consideration for PDF:

- PDF files without semantic information may need to have this reconstructed
- PDF element reflow (for changing contents or page sizes) is a difficult process and involves guesswork
- Some edits may cause some contents to change pages, causing widespread updates
- Tool and program support for editing is limited

Usually it's best to perform substantial edits in the original program of file creation, not in a PDF tool.

PDF In The World of Phones and Tablets

Many of the difficulties with editing affect viewing on smaller-format devices also:

- Fixed location specification in content streams means that most PDF pages need re-interpretation of content
  - For files missing tagging information, this needs to be done heuristically and can have problems
- Fixed size specification in content streams means that most viewers scale all PDF page content the same
  - This can turn text unreadably small or large vs. images and other content
- Fixed page areas make flowing two pages together difficult and requires guesswork

Tagged PDF can help some viewers greatly – but not all PDFs contain tagged content.

PDF Tools and Toolkits

PDF Tools and Toolkits

Many vendors and projects have been working on various aspects of PDF support since the first PDF specification was published in 1993

- Different tools and toolkits have different strengths and weaknesses
- Many different PDF file creation tools and toolkits exist, from many different formats and workflows
- PDF manipulation and rendering tools are fewer in number and more complicated to write for modern PDF format versions

Hundreds of different PDF tools are out there – the next slides only cover a few different options for application developers

PDF Programmer Toolkits: Open Source

Open Source tools and toolkits include:

- Apache PDFBox: low-level PDF creation, PDF manipulation and PDF rendering
- Apache FOP: XSL-FO processor for conversion of XML to PDF via XSL stylesheets
- pdftk: assemble, manipulate, split and join PDF files
- ReportLab open-source: PDF creation via low-level objects or specialized markup language

PDF Programmer Toolkits: Dual License

Tools and toolkits with Dual Licensing schemes include:

- ghostscript: PDF file creation, PDF rasterization
  - released as Affero GPL licensed software. Commercial use requires paid commercial licenses
  - Older versions available under GPL license without further restriction
- iText: PDF file creation, interpretation and manipulation
  - released as Affero GPL licensed software. Commercial use requires paid commercial licenses
  - Older versions available under GPL license without further restriction

PDF Programmer Toolkits: Commercial

Tools and toolkits that require commercial licensing:

- Adobe Acrobat: programmers can write plugins with the Acrobat SDK to process PDF files
  - Plugins can be written against a restricted version of the Acrobat SDK for Adobe Reader as well
- ActivePDF: suite of toolkits for PDF creation, display, manipulation and print
- PDFLib: suite of toolkits for PDF creation, manipulation and content extraction
- PDFTron: toolkit for PDF creation, display, manipulation and print

Adobe / Datalogics PDF Toolkits

Datalogics specializes in bringing Adobe PDF technologies to the developer market:

- Adobe PDF Library: the core of Adobe Acrobat and the Acrobat SDK, in SDK format
  - Includes Java and .NET interfaces to the C/C++ language SDK
- Adobe PDF Java Toolkit: the core of Adobe LiveCycle PDF processing, as an SDK for Java programs
- Datalogics PDF WebAPI: web APIs for adding popular PDF operations, such as PDF to image conversion and form filling, to internet applications
  - Powered by the Adobe PDF Library and PDF Java Toolkit

Adobe / Datalogics PDF Toolkits

Datalogics specializes in bringing Adobe PDF technologies to the developer market:

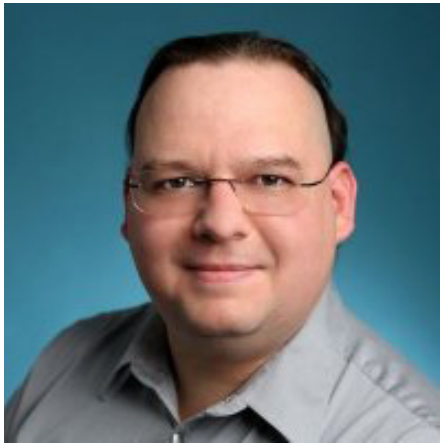
- Leading PDF display, rendering, image conversion and printing technology based on Adobe Acrobat
- Leading color handling based on Adobe Color Engine
- Over twenty continuous years of PDF development represented in the Adobe PDF Library
- Supported by PDF experts with deep industry and format experience

Come talk to any of us for more information!

Summary

Summary

- PDF is a flexible, reliable file format openly specified and documented for over twenty years
- PDF is built up from a set of objects, references and relationships between these objects
- PDF files may be self-contained for reliability, or may reference external data
- PDF files are ideal for representing documents for reliable interchange and for gathering information from people
- Many different tools exist to bring PDF support to applications, in many different languages, with many different strengths



Contact Me!

mattk@datalogics.com

http://www.datalogics.com
http://blogs.datalogics.com