

# On the Evolution of PDF

James C. King (mailto: [jking@adobe.com](mailto:jking@adobe.com))

Adobe Systems Incorporated

August 19, 2009

## 1.0 Introduction

The Portable Document Format (PDF) was developed by Adobe Systems Incorporated and the first PDF specification was published in June 1993 at the same time that Adobe announced its Acrobat product line. PDF has flourished since then and Adobe updated the PDF specification 7 times, moving it from PDF 1.0 to PDF 1.7. Each time more features were added but seldom was a feature removed. The specification went from 230 pages for PDF 1.0 to 1310 pages for PDF 1.7.

This document summarizes some of the objectives that Adobe had while updating the PDF specification and also describes the various techniques that had been used to meet those objectives.

During 2007-2008, Adobe handed the PDF 1.7 specification over to AIIM and ISO to become an International standard and in July 2008 ISO published the ISO 32000-1 standard which corresponds to Adobe's PDF 1.7. It is hoped that ISO will be as careful in the evolution of PDF as Adobe had been.

This document attempts to spell out what that level of care actually involves.

## 2.0 Objectives for Evolving PDF

Under Adobe's ownership, versions of the PDF specification and versions of Adobe's Acrobat software were synchronized such that Acrobat 1.0 followed the PDF 1.0 specification, Acrobat 2 followed the PDF 1.1 specification, and so on until Acrobat 8 was created to follow PDF 1.7 or what has become ISO 32000-1.

The challenge has been, with each new version, to add features to the PDF language while at the same time preserving the ability of newer software to process old files and older software to process new files to whatever extent possible. Of course, older and newer are defined in terms of having PDF version numbers increasing over time.

We identify three kinds of compatibility between versions of PDF files and versions of products that process PDF files.

1. Backward-compatibility is the ability of "newer" products to process "older" PDF files.
2. Forward-compatibility is the ability of older products to adequately process newer PDF files. We will qualify the word "adequately" in our subsequent discussion.

3. Feature-compatible is the ability of products that purposefully lack support for certain features, to adequately process PDF files that contain those features

We use the word “adequately” to mean that either the file is processed completely as intended, or it is processed in some acceptable but lesser way, or the user is provided with an informative message and helped with corrective action (e.g., a URL for downloading a new application). Of course, adequately is a spectrum and we always favored the most adequate treatment: processing the complete file successfully. We settled only for “adequate messages to the user” when we could devise no better solution.

We most often think of browsing software that renders the contents of PDF documents, but there are many other reasons software may process PDF files. For example, software may be written to extract the fill-in data that was collected into a PDF form. There may be no requirement for this software to render any of the pages, or any other content from the PDF file. So *adequate* processing also depends upon of the objective of processing software.

We also note that, to some extent, what is *adequate* is ultimately defined by the author of the document. One file may be quite adequately processed even though all its video clips are only presented as fixed posters whereas for yet another file its whole meaning may depend heavily upon the video clips it contains. Adobe architects have talked about having a “document requirements” feature for authors or authoring software to include in PDF files to address this, but it has not become part of PDF.

## 2.1 Backward-compatibility

Each new PDF-consuming product was required to adequately process all PDF files that have ever been created. “Once a PDF always a PDF.” Except for bugs and a few deprecated features this form of compatibility was maintained. Due to this objective, PDF, as a file format, has the advantage where any file ever created can be successfully processed by today’s software, over 16 years after PDF was first introduced.

## 2.2 Forward-compatibility

A more difficult compatibility is “forward-compatibility.” PDF was enhanced with new features with each release of the specification. What happens when older software attempts to process newer files? The incompatibility of newer PDF’s with older software has two channels for resolution:

- creating PDF’s today that we can be viewed by and interacted with using older viewers, and
- creating viewers that will adequately allow viewing and interacting with PDFs made meeting some future enhanced specification — i.e., documents from the future.

## 2.3 Feature-compatibility

Sometimes a consuming software application doesn’t care about some parts of a document and won’t be affected if a new feature is used in those parts. For example, it is not

expected to be able to print a video. A simple viewer may not be expected to allow forms data to be entered (although it might reasonably be expected to display already filled in data). These are compromises that must be made to support wider processing choices for PDFs.

## 2.4 Issues with Version Numbers in PDF Files

Software that processes PDF files is obliged to document which particular version of the PDF specification is followed by that software. Likewise, PDF files contain a version number indicating to which version of the PDF specification the file conforms.

However the use of versions numbers within PDF files has become confused. We have seen the following problems:

1. Some applications that create or modify a document just label it with the latest PDF version number, rather than with the version corresponding to the set of features actually used in the document. This causes gratuitous forward incompatibility problems.
2. Compounding this problem, some applications are needlessly restrictive about the PDF versions that they will consume.

The objective should be for applications that write PDF files to use the lowest version number that covers all the features used within that particular file and for consuming software to attempt to process as much of the file as possible.

A further complication with respect to version numbers is that ISO 32000-2 will become PDF 2.0 and it is anticipated that ISO 32000-3 will become PDF 3.0. See Section 4.0 on page 6 for further explanation of the troubles that this may cause.

## 3.0 Standard Techniques

Over the years, Adobe PDF architects used quite a variety of mechanisms to support forward and feature compatible behavior when new PDF features were introduced. These mechanisms were developed piecemeal, without any unifying principles; it is useful to have a catalog of them. Here are the ones gathered from a survey of the PDF Reference manuals.

1. Ignorable dictionary entries

By far the most common approach is to introduce new entries into existing dictionaries. Unknown dictionary entries are typically just not processed by software unaware of those entries. The entries will be maintained, however; that is a PDF rule. When a PDF file using such a new feature is encountered by consuming software unaware of that feature, the effect is as if those features were not present. All other aspects of the PDF file function normally. This works fine for self-contained features that don't alter the behavior of existing features and if ignoring the feature can be considered "adequate" processing.

On several occasions, this approach has been carried to excess. For instance, transparency was introduced as an extension to existing objects (principally the form XObject and the graphics state parameter dictionary). A PDF consumer that does not support the transparency extensions will process a transparency-containing PDF without an error message. However, the resulting appearance will be incorrect – everything is drawn opaquely and the transparency effects are lost. This is an example of an extension that alters the behavior of existing features. These compatibility consequences may be troublesome, but given the strong desire to introduce transparency, this fall-back behavior was considered “adequate,” although in most cases, at the low end of the spectrum.

## 2. Ignorable content

Two mechanisms exist to provide forward and feature compatibility for extensions appearing in PDF content streams:

- If a new content operator is introduced, it can be enclosed in a BX/EX bracketing construct. If a PDF consumer encounters an operator that it does not recognize, but the operator is bracketed with BX/EX, the operator is simply ignored, by definition. (This mechanism is no longer of much use because new content operators haven't been introduced for a long time.)
- Portions of content can be turned off using the optional content feature. Optional content is normally used to implement “layers” that a user can turn on and off. However, it can also be used to hide content that a user doesn't have access to for various reasons (such as being encrypted with a key that the user doesn't have). Optional content can be exploited in forward and feature compatibility situations to provide alternate content that an older or limited consumer will understand.

## 3. Alternate representations

In some cases, a PDF file can specify alternate representations of a feature, with the consumer free to choose which one to use. This enables the PDF producer to provide forward and feature compatible representations for older or limited consumers. Note that, by itself, the alternate representation mechanism doesn't provide compatibility; the PDF-producing application needs to use the mechanism in an intelligent way.

Features for which this has been done include:

- Annotation appearances. If a PDF consumer does not recognize an annotation type, it simply displays the annotation's static appearance stream. The appearance can contain, for instance, a poster image for a movie annotation.
- Alternate images.
- Alternate color spaces. If an ICCBased color space specifies an ICC profile in a format that the PDF consumer doesn't understand, it can fall back to an alternate color space that (hopefully) it does understand.
- Multimedia viability (see Section 5 on page 5).

## 4. Well-defined fall-back behavior

In certain cases, PDF prescribes the behavior that is to occur if a consumer encounters a feature that it recognizes but is unable to satisfy. The most common situation is a feature that is specified by an enumeration of alternatives. If the enumeration is subsequently extended with new alternatives that an old consumer doesn't recognize, PDF prescribes a fall-back default value that is to be used, instead of giving an error.

## 5. Multimedia viability

The PDF multimedia framework provides the most comprehensive forward and feature compatibility mechanism, based on the concept of “viability.”

Each multimedia object has a collection of parameters, which the PDF consumer evaluates to determine whether it can satisfy the object's requirements. If it can, the object is considered viable. If it can't, the consumer will not process the object but will consider an alternate object (if one has been specified).

The parameters for an object are divided into two groups: the “must honor” (MH) group and the “best efforts” (BE) group. The entries specified in the MH group must be honored, or else the object is not viable. The entries specified in the BE group should be honored if possible, but if they can't be, the object is still viable and the PDF consumer will take some default action.

The set of conditions determining viability can be quite diverse, including such things as:

- availability of a player for a specific media type,
- ability to display multimedia in a window of specified dimensions and position, and
- ability to resolve a URL reference to an external file.

It is important to understand how introduction of new features affects viability. If a new entry is placed in the MH group and is not understood by an old consumer, the object is considered non-viable. But if a new entry is placed in the BE group and is not understood by an old consumer, it is simply ignored and does not influence viability. This gives the PDF producer some control over what happens if the PDF file is encountered by an old or limited consumer.

## 6. Compatibility-checking scripts

PDF producers can include JavaScripts that execute when a document is opened and those scripts can check the version of the reader or XFA engine. If the engine version is too low to support a feature that the PDF file depends on, the script can take some action. Usually, the action is to notify the user that certain features of the document will not work and to recommend that the user upgrade to a newer software version. Generally, there is little that the script can do to mitigate the forward incompatibilities.

Today there are thousands of applications created by hundreds of organizations that process PDF files in some way. This means that in order to preserve the utility PDF documents, as we update the ISO 32000 specification, we need to pay particular attention to the three kinds of compatibility outlined above.

## 4.0 PDF Version and Extension Numbering

### 4.1 Version Numbering

As noted earlier Adobe defined PDF versions 1.0 through PDF 1.7 and ISO 32000-1 is the same as PDF 1.7. The ISO PDF Committee has decided to use versions corresponding to the ISO 32000 part number so that ISO 32000-2 will use the PDF 2.0 designation within the PDF files. One reason to do this was to skip over the pending problem when PDF 1.10 could possibly be confused with PDF 1.1. It should also be convenient to associate part numbers and version numbers.

The use of two integers separated by a period (e.g, 1.0, 1.7 or 2.0) was claimed in Adobe's documentation to be of the form of a "major version number" followed by a "minor version number" with the major incompatible changes requiring an advance of the major version number. Adobe never change the major number (from 1) even though, arguably some of the changes should have prompted an increment. With the revised ISO numbering scheme, to date, no definition of what the number to the right of the period should signify and until that happens zero will be used.

This is now a complete change in the version numbering strategy which will have to be recognized by PDF processing software.

### 4.2 Extension Numbering

ISO 32000-1 introduced an extension denotation mechanism that allows organizations to define and publish extensions to PDF versions and to include information within the PDF file to denote their usage in the file. A new key, Extensions, has been defined for the document's catalog which refers to one or more developer's extensions denotations which include the version of the PDF specification, BaseVersion, from which the developer's extension was developed and a developers version number, ExtensionLevel.

The Extension level is an integer defined by the developer to denote the extension being used. If the developer introduces more than one extension to a given BaseVersion the extension level numbers assigned by that developer shall increase over time. The developer is obliged to publish the specification of the extensions and register them with the ISO PDF Registry as described in Annex E of the ISO 32000-1 standard. Here is an example of what we might find in a PDF file that uses an Adobe extension 3 and a GLGR extension 1002 both defined against the PDF 1.7 (ISO 32000-1) specification.

```
%PDF-1.7
<</Type /Catalog/Extensions
  <</ADBE
    <</BaseVersion /1.7
      /ExtensionLevel 3
    >>
  /GLGR
  <</BaseVersion /1.7
    /ExtensionLevel 1002
```

>>  
>>  
>>

### ***Example 1 -- The developer extension denotation mechanism***

#### **4.3 Scope of a PDF Version**

The purpose of a PDF specification is to define what can be put into a PDF file and what is the meaning of that material. The PDF specifications, both those from Adobe and now the one from ISO make references to a large number of other technical specifications for things, such as ICC Profiles, Open Type Fonts and JPEG images, which can be included within a PDF file. So if one wishes to thoroughly understand everything that can be included in a PDF file, one has to read and understand a large tree structure of technical specification only the root of which is the ISO 32000-1 document. The tree is estimated to contain over 50 different standards or documents.

In order to avoid confusion as to what may or may not be included within a PDF file conforming to a certain PDF version, we must fix the versions of the subordinate documents associated with that fixed version of PDF. So the complete PDF specification not only means the ISO 32000 part number but exact versions of all subordinate documents as well. If the ISO committee decides that a revised version of a subordinate document should become the official definition for that feature in PDF, then a new ISO 32000 part must be created that references the new subordinate standard.

This is a rather strong position but it is necessary to allow, say, PDF 2.0, to have a well defined meaning and to know exactly what documentation defines proper file contents. Perhaps this is a use for the minor version number: to denote sets of changes to subordinate documents.

#### **5.0 Summary of Compatibility Recommendations**

Here is a summary of techniques which can be used to mitigate compatibility issues.

- Consume any version level of PDF including past and future version levels.
- Plan a complete feature even though some of it may not be immediately implemented in products. Introducing a feature piecemeal aggravates the version compatibility issues.
- Make it as easy as possible for customers to upgrade to new software.
- Avoid tying clients and servers together around version levels of PDF. Both clients and servers should strive to process all levels as adequately as possible.
- Use the existing extensibility features of our languages when designing new features.

#### **6.0 PDF Profiles (i.e., subsets)**

Many proper profiles of PDF have become ISO standards. They allow support for various vertical markets more effectively than open-ended PDF might. These standards efforts

also bring together a group of knowledgeable and experienced people in those vertical markets.

Here is the current list of ISO standards that are subsets of PDF:

#### **6.1 PDF/X (ISO 15930) – targeted for prepress workflows.**

- ISO 15930-1:2001: PDF/X-1a:2001, Blind exchange in CMYK + Spot Colors, based on PDF 1.3
- ISO 15930-3:2002: PDF/X-3:2002, Allows CMYK, Spot, Calibrated (managed) RGB, CIELAB, with ICC Profile, based on PDF 1.3.
- ISO 15930-4:2003: PDF/X-1a:2003, revision of PDF/X-1a:2001 based on PDF 1.4
- ISO 15930-5: PDF/X-2, An extension of PDF/X-3 which allows for OPI-like (external linked) data to be included
- ISO 15930-6:2003: PDF/X-3:2003, revision of PDF/X-3:2002 based on PDF 1.4
- ISO 15930-7:2008: PDF/X-4, Colour-managed, CMYK, gray, RGB or spot colour data are supported, as are PDF transparency and optional content. A second conformance level named PDF/X-4p may be used when the ICC Profile in the output intent is externally supplied.
- ISO 15930-8:2008 PDF/X-5, a collection of three conformance levels:
  - PDF/X-5g: An extension of PDF/X-4 that enables the use of OPI-like workflows.
  - PDF/X-5pg: An extension of PDF/X-4p that enables the use of OPI-like workflows in conjunction with a reference to an external ICC Profile for the output intent.
  - PDF/X-5n: An extension of PDF/X-4p that allows the externally supplied ICC Profile for the output intent to use a color space other than Grayscale, RGB and CMYK.

#### **6.2 PDF/A (ISO 19005) – targeted for document archiving**

- PDF/A-1a - Level A compliance based on PDF 1.4
- PDF/A-1b - Level B compliance based on PDF 1.4
- PDF/A-2 (ISO 19005, Part-2) in progress based upon ISO 32000-1.

#### **6.3 PDF/E (ISO 24517) – targeted for engineering especially engineering drawings**

- PDF/E (ISO 24517-1:2008) based on PDF 1.6.

#### **6.4 PDF/UA (ISO/AWI 14289) – for universal access.**

- This one is not yet a standards but a committee has been formed to make it so.

#### **6.5 Questions and concerns about the proliferation of profiles (subsets)**

1. Is it possible to make a PDF file that obeys more than one profile specification?
2. Are they all true subsets?
3. Will ISO be able to stay abreast of these activities and guide them in a direction that will not corrupt the PDF standard and compatibility?

If the proliferation of profiles is allowed to grow unbounded a rather chaotic and counter productive environment will be created. As much as possible we should re-use those profiles that have already been defined.

Note of Credit: This document was derived from an earlier Adobe document that had been produced by a study group consisting of Nabeel Al-Shamma, Jim Donahue, Macduff Hughes, Mark Manca, Roberto Perelman, Ed Taft, Phil Ydens, Jeff Young and Jim King. Thanks to them for their contributions.

§§§§§§§§§§

8/19/2009 – Jim King <jking@adobe.com>