Embedding XMP Metadata in Application Files

DRAFT

September 14, 2001

ADOBE SYSTEMS INCORPORATED
Corporate Headquarters
345 Park Avenue
San Jose, CA 95110-2704
(408) 536-6000
http://www.adobe.com

This publication and the information herein is furnished AS IS, is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies, makes no warranty of any kind (express, implied, or statutory) with respect to this publication, and expressly disclaims any and all warranties of merchantability, fitness for particular purposes, and noninfringement of third party rights.

Contents

1 Embedding XMP Metadata in Application Files

1.1 Introduction

This document describes how XMP metadata can be embedded in the following file formats: TIFF, JPEG, GIF, PNG, HTML, PDF, SVG/XML, AI, and EPS. This information is intended to help application developers understand how to embed XMP metadata in a standardized form to help ensure better document interchange.

This document is currently a stand-alone draft; when approved and finalized, it will become an appendix to the *XMP – Extensible Metadata Platform* document.

> **NOTE:** This document assumes that the reader has a basic knowledge of the referenced file formats, and does not attempt to fully describe each format or the terms used.

1.2 TIFF

In TIFF files, the XML Packet containing XMP metadata is pointed to by an entry in the Image File Directory (IFD). That entry has a Tag value of 700, as shown in Table 1.1, "TIFF IFD Directory Entry for XML Packets."

TABLE 1.1 TIFF IFD Directory Entry for XML Packets

Byte Offset	Field value	Field Tame	Comments
0, 1	700	TAG	Tag that identifies the field (decimal value).
2,3	1	Field type	The field type is "BTYE", which is represented as a value of "1".
4–7		Count	The number of values of the indicated type, or count of the indicated Type. For metadata, this is the total byte count of the XML Packet.
8–11		Value or Offset	The value offset, in bytes, to the location of the XML Packet.

References:

TIFF 6.0 Specification:

http://partners.adobe.com/asn/developer/pdfs/tn/TIFF6.pdf

1.3 JPEG

XML Packets embedded in JPEG files use an APP1 marker to designate the location of the XML packet containing the XMP metadata. Table 1.2 shows the entry format.

TABLE 1.2 Content of XMP JPEG APP1

Byte Offset	Field value	Field name	Length	Comments
0	0xFFE1	APP1	2	XMP application data marker
2	2 + length of namespace + length of XMP data	Lp		
4	Null-terminated ASCII string without quotation marks.	namespace		
	XMP data			

The namespace used for XMP metadata in a JPEG file is:

http://ns.adobe.com/xap/1.0/

The Header plus the following data must be less than 64 KB bytes. While there can be multiple markers, the XMP data cannot be split across the multiple markers, so the metadata must be less than 64 KB minus the length of the header.

References:

(1) JPEG File Interchange Format Version 1.02

(2) ISO/IEC 10918-1 Information technology - Digital Compression and Coding of continuous-tone still images: requirements and guidelines.

(3) ISO/IEC 10918-4 Information technology — Digital compression and coding of continuous-tone still images: Registration of JPEG profiles, SPIFF profiles, SPIFF tags, SPIFF colour spaces, APPn markers, SPIFF compression types and Registration Authorities (REGAUT)

NOTE: Reference (3) specifies the format of APPn markers and the file interchange format.

1.4 GIF

An XML packet in a GIF89a file should be encoded in an Application Extension block. The application identifier of that block should be ‘XMP Data’ and the Application Authenticator should be ‘XMP’. The XML packet follows that block, encoded in UTF-8. That is followed by a 258 byte trailer, where the byte values are as shown in Table 1.3, beginning with 0x01, 0xFF, and then descending values until the final bytes with the values 0x01, 0x00, and the final byte (258) is the 0x00 Block Terminator.

TABLE 1.3 XMP in GIF File Format

	7 6 5 4 3 2 1 0	Field Name	Type
0	0x21	Extension Introducer	Byte
1	0xFF	Extension Label	Byte
0	0x0B	Block Size	Byte
1	‘X’ 0x58	Application Identifier	8 Bytes
2	‘M’ 0x4D		
3	‘P’ 0x50		
4	‘ ’ 0x20		
5	‘d’ 0x64		
6	‘a’ 0x61		
7	‘t’ 0x74		
8	‘a’ 0x61		
9	‘X’ 0x58	Application Authentication Code	3 Bytes
10	‘M’ 0x4D		
11	‘P’ 0x50		
	[XML Data]	XML Packet, UTF-8 encoded	Byte
	0x01	XML Packet Trailer	258 Bytes
	0xFF		
	0xFE		
	⋮		
	0x01		
	0x00		
	0x00	Block Terminator	Byte

1.5 PNG

An XML packet can be embedded in a PNG graphic file by adding a chunk of type “tXMP”. The data of the chunk should be a UTF-8 serialized XML packet.

There should be no more than one ‘tXMP’ chunk present in each PNG file. Encoders are encouraged to place the chunk at the beginning of the file, but this is not a requirement.

The specification for the PNG file format can be found at:

http://www.w3.org/TR/REC-png.html

1.6 HTML

Adobe recommends the use of W3C recommendations for embedding XML in HTML. This was the subject of a meeting held by W3C in May 1998; the meeting report may be found at

http://www.w3.org/TR/NOTE-xh

Briefly, the use of the HTML *SCRIPT* element is the recommended method.

1.7 PDF

For PDF files, the XML packet is embedded in a metadata stream (PDF 1.4) contained in a PDF object. Information on the use of metadata in PDF files is available in *XMP – Extensible Metadata Platform* document (see Appendix A, “PDF and Dublin Core Schema”).

The following is an example of XMP metadata embedded as an XML packet, stored as a metadata stream:

Example 1.1 Example of XMP Metadata in a PDF File

```
1152 0 obj
<< /Type /Metadata /Subtype /XML /Length 1706 >>
stream
<?xpacket begin='' id='W5M0MpCehiHzreSzNTczkc9d' bytes='1706'?>

rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
 xmlns:iX='http://ns.adobe.com/iX/1.0/'>

<rdf:Description about=''
  xmlns='http://ns.adobe.com/pdf/1.3/'
  xmlns:pdf='http://ns.adobe.com/pdf/1.3/'>
  <pdf:CreationDate>2001-08-09T11:50:18Z</pdf:CreationDate>
  <pdf:ModDate>2001-08-09T12:24:47-07:00</pdf:ModDate>
  <pdf:Producer>Acrobat Distiller 5.0 for Macintosh</pdf:Producer>
  <pdf:Author>Adobe Systems, Inc.</pdf:Author>
  <pdf:Creator>FrameMaker 6.0</pdf:Creator>
  <pdf:Title>The XMP Toolkit</pdf:Title>
  <pdf:Subject>Reference document for the XAP Toolkit</pdf:Subject>
  <pdf:Keywords>XMP Toolkit metadata specification</pdf:Keywords>
 </rdf:Description>

<rdf:Description about=''
  xmlns='http://ns.adobe.com/xap/1.0/'
  xmlns:xap='http://ns.adobe.com/xap/1.0/'>
  <xap:CreateDate>2001-08-09T11:50:18Z</xap:CreateDate>
  <xap:ModifyDate>2001-08-09T12:24:47-07:00</xap:ModifyDate>
  <xap:Author>Adobe Systems, Inc.</xap:Author>
  <xap:MetadataDate>2001-08-09T12:24:47-07:00</xap:MetadataDate>
  <xap:Title>
   <rdf:Alt>
    <rdf:li xml:lang='x-default'>The XMP Toolkit</rdf:li>
   </rdf:Alt>
  </xap:Title>
  <xap:Description>
   <rdf:Alt>
    <rdf:li xml:lang='x-default'>Reference document for the XMP
        Toolkit</rdf:li>
   </rdf:Alt>
  </xap:Description>
 </rdf:Description>

<rdf:Description about=''
  xmlns='http://purl.org/dc/elements/1.1/'
  xmlns:dc='http://purl.org/dc/elements/1.1/'>
  <dc:creator>Adobe Systems, Inc.</dc:creator>
  <dc:title>The XMP Toolkit</dc:title>
  <dc:description>Reference document for the XMP
     Toolkit</dc:description>
 </rdf:Description>

/rdf:RDF>
<?xpacket end='r'?>
endstream
endobj
```

The values of the two date fields are required to be in the date format used by XMP, not the date format used by PDF.

Users of PDF are permitted to define their own metadata items in the PDF Info dictionary. There is another schema whose namespace URI is ‘`http://ns.adobe.com/pdfx/1.3/`’, usually given the namespace prefix ‘`pdfx`’, which is used to store any such user-defined keys. They are given the same name as the Info dictionary item.

Acrobat 5.0 has the limitation that user-defined keys must be valid XML names as defined in the XML 1.0 Recommendation <http://www.w3.org/TR/REC-xml> so that this correspondence can be put into effect.

1.7.1 XMP in PDF Metadata Dictionaries and Streams

The *PDF Reference*, Version 1.3, for PDF 1.4, specifies two methods for specifying metadata in PDF files: as a dictionary, and as a metadata stream. Because that specification has not been published as of the data of this draft document, that information is duplicated here for convenience. The PDF Reference will soon (late 2001) be available at:

http://partners.adobe.com/asn/developer/technotes/main.html

NOTE: The remainder of this section 1.7.X consists of text and tables excerpted from the forthcoming *PDF Reference, Third Edition (for PDF language version 1.4). The table and example numbering, as well as page and chapter references, in the text below, refers to tables and examples in that document. This section will be removed from this document when the new PDF Reference has been published.*

Metadata can be stored in a PDF document in either of the following ways:

- In a *document information dictionary* associated with the document
- In a *metadata steam (PDF 1.4)* associated with the document or a component of the document

Document Information Dictionary

The optional **Info** entry in the trailer of a PDF file (see Section 3.4.4, “File Trailer”) can hold a *document information dictionary* containing metadata for the document; Table 1.4 shows its contents. Any entry whose value is not known should be omitted from the dictionary, rather than included with an empty string as its value.

Some extensions may choose to permit searches on the contents of the document information dictionary. To facilitate browsing and editing, all keys in the dictionary are fully spelled out, not abbreviated. New keys should be chosen with care so that they make sense to users.

Note: Although viewer applications can store custom metadata in the document information dictionary, it is inappropriate to store private content or structural information there; such information should be stored in the document catalog instead (see Section 3.6.1, “Document Catalog”).

Table 1.4 Entries in a document information dictionary

KEY	TYPE	VALUE
Title	text string	*(Optional; PDF 1.1)* The document’s title.
Author	text string	*(Optional)* The name of the person who created the document.
Subject	text string	*(Optional; PDF 1.1)* The subject of the document.
Keywords	text string	*(Optional; PDF 1.1)* Keywords associated with the document.
Creator	text string	*(Optional)* If the document was converted to PDF from another format, the name of the application (for example, Adobe FrameMaker®) that created the original document from which it was converted.
Producer	text string	*(Optional)* If the document was converted to PDF from another format, the name of the application (for example, Acrobat Distiller) that converted it to PDF.
CreationDate	date	*(Optional)* The date the document was created, in human-readable form (see Section 3.8.2, “Dates”).
ModDate	date	*(Optional; PDF 1.1)* The date the document was last modified, in human-readable form (see Section 3.8.2, “Dates”).
Trapped	name	*(Optional; PDF 1.3)* A name object indicating whether the document has been modified to include trapping information (see Section 9.10.5, “Trapping Support”): True The document has been fully trapped; no further trapping is needed. (Note that this is the name True, not the boolean value **true**.) False The document has not yet been trapped; any desired trapping must still be done. (Note that this is the name False, not the boolean value **false**.) Unknown Either it is unknown whether the document has been trapped or it has been partly but not yet fully trapped; some additional trapping may still be needed. Default value: Unknown. The value of this entry may be set automatically by the software creating the document’s trapping information or may only be known to a human operator and entered manually.

Example 1.2 shows a typical document information dictionary.

Example 1.2

```
1 0 obj
    << /Title (PostScript Language Reference, Third Edition)
       /Author (Adobe Systems Incorporated)
       /Creator (Adobe® FrameMaker® 5.5.3 for Power Macintosh)
       /Producer (Acrobat® Distiller™ 3.01 for Power Macintosh)
       /CreationDate (D:19970915110347-08'00')
       /ModDate (D:19990209153925-08'00')
    >>
endobj
```

Metadata Streams

Metadata, both for an entire document and for components within a document, can be stored in PDF streams called *metadata streams (PDF 1.4)*. The advantages of metadata streams over the document information dictionary include the following:

- PDF-based workflows often embed metadata-bearing artwork as components within larger documents. Metadata streams provide a standard way of preserving the metadata of these components for examination downstream. PDF-aware applications should be able to derive a list of all metadata-bearing document components from the PDF document itself.
- PDF documents are often made available on the World Wide Web or in other environments, where many tools routinely examine, catalog, and classify documents. These tools should be able to understand the self-contained description of the document even if they do not understand PDF.

Besides the usual entries common to all stream dictionaries (see Table 3.4 on page 40), the metadata stream dictionary contains the additional entries listed in Table 1.5.

The contents of a metadata stream is the metadata represented in Extensible Markup Language (XML). This information will be visible as plain text to tools that are not PDF-aware only if the metadata stream is both unfiltered and unencrypted.

Table 1.5 Additional entries in a metadata stream dictionary

KEY	TYPE	VALUE
Type	name	*(Required)* The type of PDF object that this dictionary describes; must be **Metadata** for a metadata stream.
Subtype	name	*(Required)* The type of metadata stream that this dictionary describes; must be **XML**.

The format of the XML representing the metadata is defined as part of an XML metadata framework specified by Adobe (in a document that was not available at the time of publication of this book but that will eventually be available on the ASN Developer Program Web site; see the Bibliography). This framework provides a way to use XML to represent metadata describing documents and their components, and is intended to be adopted by a wider class of applications than just those that process PDF. It includes a method to embed XML data within non-XML data files in a platform-independent format that can be easily located and accessed by simple scanning rather than requiring the document file to be parsed.

A metadata stream can be attached to a document through the **Metadata** entry in the document catalog (see Chapter 3.6.1, “Document Catalog,” and also see implementation note 103 in Appendix H). In addition, most PDF document components represented as a dictionary or stream can have a **Metadata** entry (see Table 1.6).

Table 1.6 Additional dictionary entry for components having metadata

KEY	TYPE	VALUE
Metadata	stream	*(Optional; PDF 1.4)* A metadata stream containing metadata for the component.

In general, a PDF stream or dictionary can have metadata attached to it as long as the stream or dictionary represents an actual information resource, as opposed to serving as an implementation artifact. Table 9.5 lists the PDF constructs that are considered implementational (and hence cannot have associated metadata), and indicates where further information about each one can be found.

References:

PDF Reference, Version 1.3 (for PDF Language Version 1.4); not released at this time, but soon to be posted at:

http://partners.adobe.com/asn/developer/technotes/acrobatpdf.html

1.8 SVG/XML

Since XMP uses the standard XML representation of RDF, the serialized XMP metadata can be directly embedded within an XML document. This can be with or without the XML packet wrapper, which is a pair of XML processing instructions. Using the XML packet wrapper is recommended, doing so will allow the metadata to be located by other software in a uniform manner.

The XMP metadata might have an outermost <x:xapmeta> or <x:xmpmeta> element, where x is the namespace “adobe:ns:meta/”. This is a serialization option. Next there will be an <rdf:RDF> element, inside of which are the <rdf:Description> elements containing the properties.

There is no XML declaration, the XMP metadata is not intended to be a complete standalone XML document. The XMP may be placed anywhere within the XML document. If the toolkit is asked to parse the entire XML document it is capable of finding the <rdf:RDF> element.

If a DTD is required, be aware that RDF provides many equivalent ways to express the same model. Thus the safest approach is to use a DTD describing any possible RDF. We don’t know of a standard DTD fragment to do this.

It is recommended that the file be encoded as Unicode using UTF-8 or UTF-16. This provides compatibility for software that scans for XML packets and parses just their content.

1.9 Adobe Illustrator .ai Format

An Adobe Illustrator generated “.ai” file uses the PDF file format. Hence, the format for embedding XMP metadata is the same as for PDF files. See Section 1.7, “PDF” for information on PDF metadata embedding.

1.10 EPS

XMP metadata packets are embedded in EPS files as a stream. The following pseudo-code illustrates how it is included in an EPS file:

```
% some PostScript
% some PostScript
currentfile 0 (%% XML ENDS HERE) /SubFileDecode filter
..............xml..........................
..............xml..........................
..............xml..........................
%% XML ENDS HERE
% PostScript continues here
% PostScript continues here
```

Note: the "%% XML ENDS HERE" comment is not a literal requirement.