PDF association

Application Note

Including custom metadata structures in PDF

2025-10

PDF TWG

PDF Association
Friedenstr. 2A · 16321 Bernau bei Berlin · Germany

E-mail: copyright@pdfa.org
Web: pdfa.org
Published in Germany and the United States of America

Table of Contents

Introduction

Many industries and organizations have defined various metadata schemes beyond those specified in ISO standards for PDF technology. In PDF, all such third-party schemes describe “custom metadata”.

Such custom metadata often exists as manifest files in various serialization formats supporting existing complex workflows and systems. Organizations leveraging these metadata in the PDF context can benefit from guidance regarding appropriate method(s) for including their custom metadata in PDF documents, enabling smoother and more efficient workflows and avoiding “sidecar files” or less user-friendly ZIP archive-based solutions.

This Application Note focuses on methods for incorporating custom metadata structures into PDF files and provides recommendations and guidance to assist organizations in selecting the most suitable mechanism(s) for storing their custom metadata manifests in PDF documents, consistent with ISO standards for PDF. A basic understanding of PDF is assumed.

NOTE: This Application Note focuses on larger custom metadata schemes and thus does not provide detailed advice on other mechanisms more appropriate for lightweight or small custom metadata data points.

PDF's metadata mechanisms

Custom metadata manifests can be stored in PDF documents in several ways, including:

- In XMP Metadata streams;
- As embedded Associated Files;
- Utilizing PDF Collections;
- As private PieceInfo data.

All these methods may be used to support custom metadata at the document level, with some methods also supporting custom metadata at an object level (e.g., associated with a page, an image XObject, an annotation, or other PDF objects).

This document also provides basic guidance on compatibility with PDF/A (ISO 19005 series of standards), as well as viewer application and PDF JavaScript support for these methods. Note that not all PDF applications support JavaScript.

XMP metadata

XMP metadata is the preferred method for storing metadata in modern PDF files. Defined by ISO 16684-1:2019, the XMP standardizes a data model, a serialization format based on RDFa/XML, and core properties for defining and processing extensible metadata. It is supported in both PDF 1.x and PDF 2.0, as well as in all ISO-standardized PDF subsets. XMP can be associated with an entire document or any specific PDF object, such as a page, image XObject, structure element, or a marked content sequence in a content stream. All modern PDF files include document-level XMP metadata, which provides standard document metadata such as author, title, producer, and creator (see ISO 16684-1, §8.3 and existing XMP namespaces), as well as ISO-standardized PDF conformance declarations.

Many XMP metadata schemas are already defined and widely adopted across various industries; these should be leveraged whenever possible, before defining new custom metadata. Industry-standardized PDF Declarations also leverage XMP to indicate the PDF document's conformance to third-party standards or specifications.

All XMP metadata in PDF is an XMP packet as the value of the **Metadata** key in a PDF stream object. Although **Metadata** entries can be associated with any PDF object, they are most commonly included as document-level metadata in the stream referenced by the document catalog's **Metadata** entry (ISO 32000-2:2020, Table 29). Like any PDF stream, it may be compressed or uncompressed, but document-level XMP metadata is typically left uncompressed. XMP metadata always uses UTF-8 encoding.

XMP is the most appropriate method for small custom metadata payloads, payloads with an existing XMP or RDFa serialization, or relatively simple XML-based payloads. When utilizing XMP for custom metadata, it is recommended to define a single preferred serialization by providing a RELAX-NG schema (and, if PDF/A-1, PDF/A-2, or PDF/A-3 support is relevant, a corresponding PDF/A Extension Schema). In addition to helping developers to validate their custom metadata support, the custom metadata schemas are needed by PDF/A (see below) to meet archival needs.

Encrypted PDF documents may also indicate whether the document-level XMP metadata is encrypted or unencrypted via the **EncryptMetadata** entry in the encryption dictionary (ISO

32000-2:2020, Tables 21 and 27). In encrypted PDF documents, all object-level XMP metadata streams will always be encrypted.

```
100 0 obj
<< % an unencrypted and uncompressed XMP metadata stream
  /Type /Metadata
  /Subtype /XML
  /Length …
>>
stream
<?xpacket begin="?" id="W5M0MpCehiHzreSzNTczkc7e"?>
  <x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="ACME XMP Toolkit 2.34">
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      …
    </rdf:RDF>
  </x:xmpmeta>
<?xpacket end="w"?>
endstream
endobj
```

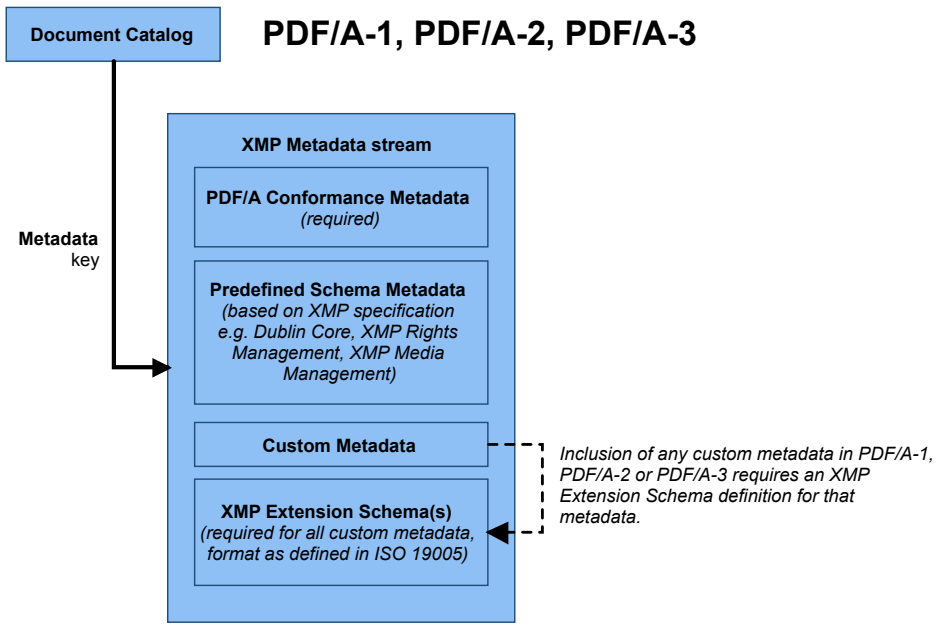
PDF/A requirements

Long-term preservation implies the inclusion of schema information, allowing custom metadata to be reliably understood in the future.

PDF/A-1, PDF/A-2, PDF/A-3

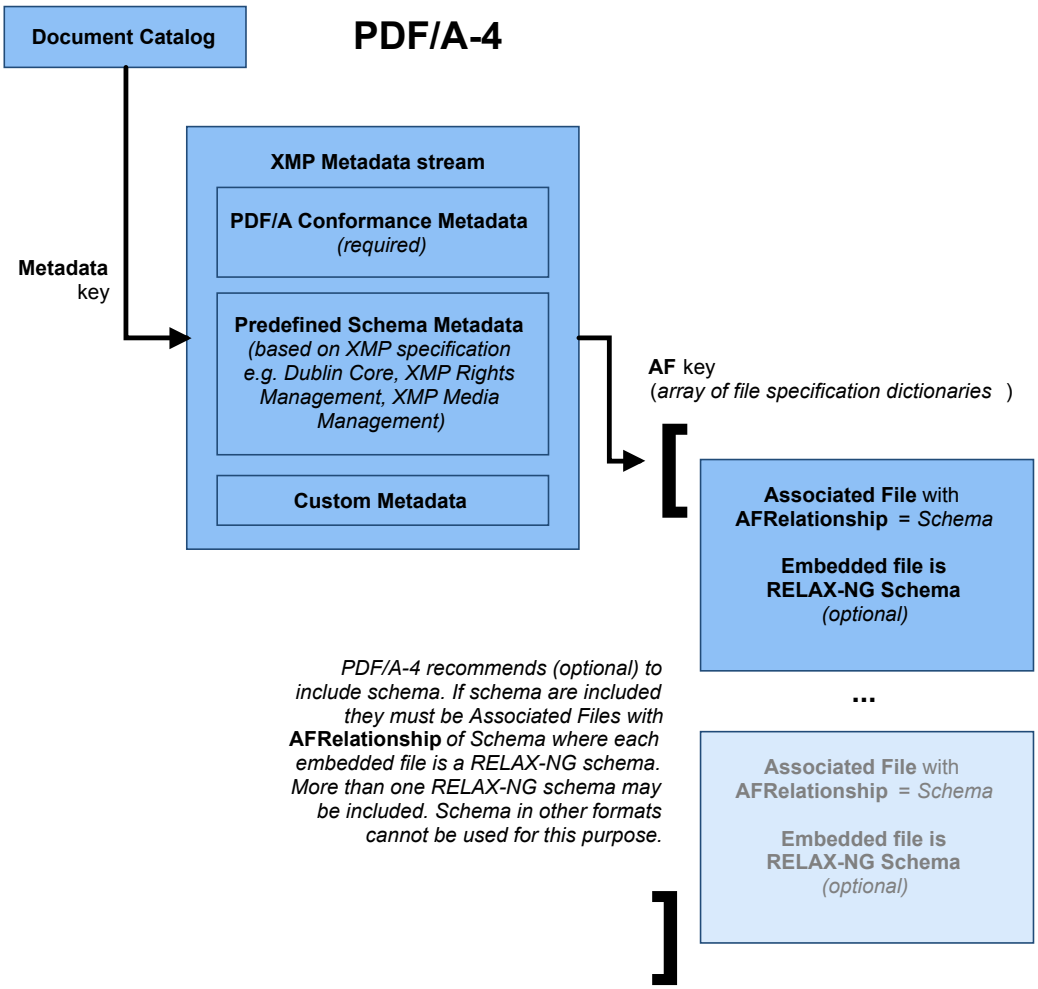
If a PDF/A-1, PDF/A-2, or PDF/A-3 file includes custom metadata in XMP, the standard requires that the respective PDF/A Extension Schema is included in the PDF/A file. Refer to the appropriate part of ISO 19005 for specific information.

The PDF Association provides a set of free XMP Extension Schema templates for ISO-standardized PDF subsets, PDF Declarations, and other common XMP data used with PDF/A-1, PDF/A-2, and PDF/A-3 documents.



PDF/A-4

If a PDF/A-4 file includes XMP custom metadata, ISO 19005-4 recommends that a RELAX-NG schema (based on ISO 16684-2:2014) also be included as a separate Associated File with an **AFRelationship** value of Schema (ISO 19005-4).



Application Support

Support for document-level XMP metadata is required when ascertaining whether a file claims conformance with ISO-standardized subsets of PDF. Although some specialized PDF applications can also display document- and/or object-level metadata, the majority of simple PDF viewing applications do not provide end-user-friendly visualization of XMP metadata, and often only display a small set of document metadata from the document information dictionary (ISO 32000-2:2020, §14.3.3).

Thus, end-user access to custom metadata within XMP metadata is currently limited to PDF applications that provide technical views of the document-level XMP. Some applications may support JavaScript that can process XMP, or might offer support (as Adobe Acrobat does) for custom file information panels, which can provide a limited, tailored presentation of XMP metadata.

JavaScript Support

ISO 21757-1 Document management — ECMAScript for PDF — Part 1: Use of ISO 32000-2 (PDF 2.0) standardizes the use of JavaScript with PDF 2.0. This standard was based on the JavaScript support that Adobe defined for PDF 1.7 in its "PDF JavaScript API".

Using the mechanisms defined in these APIs, PDF JavaScript can access the full document XMP metadata via the `doc` object's `metadata` property (see the `Doc` object in ISO 21757-1, §10.13 and Adobe Acrobat SDK documentation). This API returns a string containing all metadata as XML, which can then be further processed to extract custom metadata.

> NOTE: The XMLData static object defined in the Adobe Acrobat SDK documentation is not standardized in ISO 21757-1.

Associated Files

Embedding custom metadata as PDF Associated Files (ISO 32000-2:2020, §14.13) is appropriate for complex metadata or custom metadata that already has well-defined and widely supported serializations, such as JSON or YAML.

Associated Files can be associated with any PDF object by the use of an **AF** entry, including the document via the document catalog, pages, individual PDF objects, or marked-content sequences. It is recommended that all Associated Files are listed in the document catalog **EmbeddedFiles** name-tree to maximise their discoverability. The PDF Association's "PDF 2.0 Application Note 002: Associated Files" provides more information.

Each Associated File includes an AFRelationship entry, which defines a basic semantic relationship with the associated PDF object. Although PDF specifications define several possible values for AFRelationship (ISO 32000-2:2020, Table 43 and ISO 19005-3:2012, Annex E), these values are insufficient for ensuring the reliable discoverability and interoperability of custom metadata.

It is also recommended that custom metadata manifests:

- Use a dedicated 2nd-class name to clearly identify its role. For example, C2PA mandates that the **AFRelationship** value be C2PA_Manifest. This assists end-users using file attachment panes in PDF viewers to easily identify the custom metadata manifest file from other embedded files.
- Use an easily identifiable filename convention to support end users, JavaScript, and legacy applications. This can be either a fixed filename (such as "`factur-x.xml`" as specified by ZUGFeRD for e-invoices) or rely on matching prefixes (e.g., the use of an identifiable prefix, such as the registered 2nd-class name prefix) and known file extensions.
- Have appropriate IANA media type(s) (without parameters) specified for the **Subtype** entry of the embedded file stream dictionary to maximize interoperability. Ensuring that manifest formats are compatible with JavaScript can enable application or document functionality via scripting.

Document-level custom metadata manifests are included in the document catalog **AF** array entry, which may also include other associated files with other **AFRelationship** values.

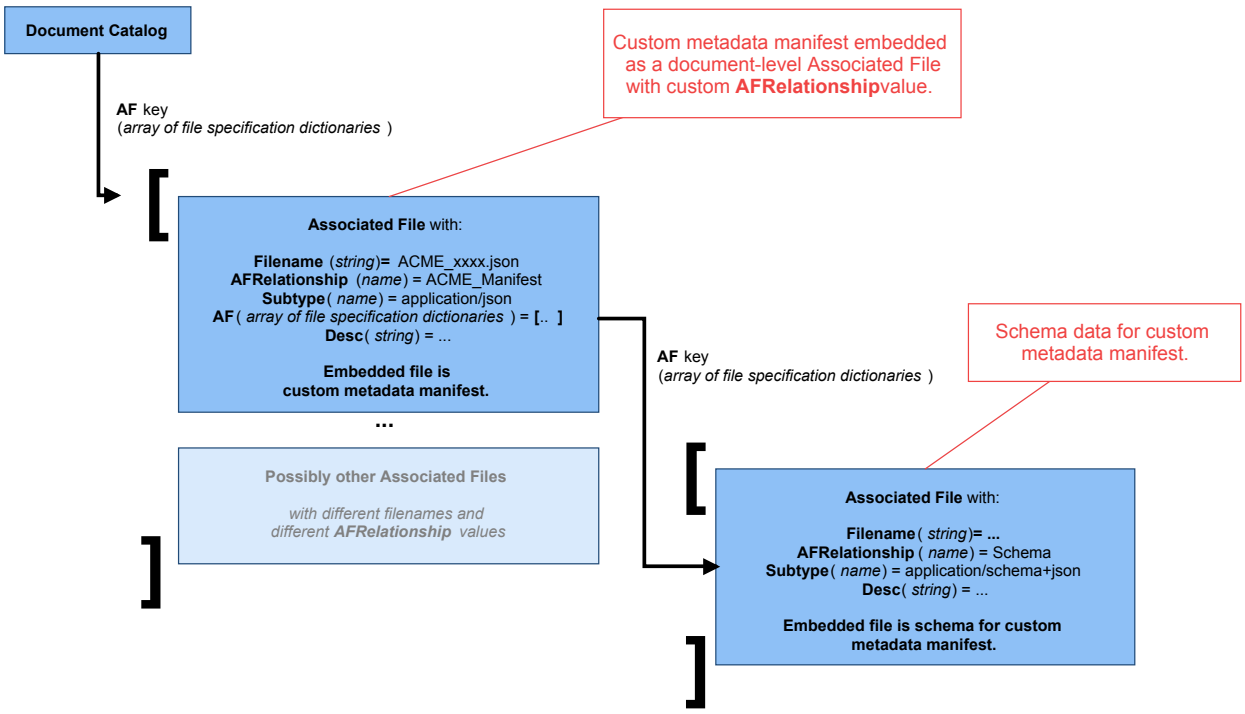
Like any PDF stream, custom metadata serializations and related schema files can be compressed or uncompressed when the PDF is written. However, in encrypted PDF documents, all strings and streams must be encrypted; accordingly, all custom metadata serializations and related schema files will be encrypted (i.e., there is no equivalent mechanism to the XMP **EncryptMetadata** entry for Associated Files).

Associated Files are similar to, but more expressive than, File attachment annotations. File attachment annotations can only be associated with pages, typically with an on-page visualization graphic of a paperclip or pushpin that enables user interaction. The embedded file streams associated with file attachment annotations do *not* have to be listed in the document's **EmbeddedFiles** name tree and thus may not display in some PDF viewers, or may not be visible until/unless the page with the annotation is viewed. There is also no semantic relationship defined between the embedded file and the PDF, so the use of Associated Files for custom metadata manifests is the preferred method.

PDF/A requirements

For the ISO-standardized archival subset of PDF, the use of Associated Files is limited to PDF/A-3 (for PDF 1.7) and PDF/A-4f or PDF/A-4e (for PDF 2.0).

ISO 19005-4 recommends that, in addition to embedding the custom metadata serialization as an Associated File, a corresponding schema be included as a separate Associated File with an **AFRelationship** value of "Schema" as an array element of the **AF** entry in the custom metadata file specification dictionary.



Application Support

Many PDF viewers provide interactive user functionality to list and extract embedded files, typically in a dedicated file navigation pane. This functionality typically uses the document catalog's **EmbeddedFiles** name tree to list embedded files; however, implementations vary in which filename entry (**F** or **UF**) is presented to the user. Although most of these interfaces provide at least the embedded file's filename, description, size, and type, today, few applications display the **AFRelationship** entry or provide any information specific to Associated Files.

For this reason, ensuring that embedded custom metadata is included in the document catalog's **EmbeddedFiles** name tree with an appropriate filename and description (file specification dictionary **Desc** entry or file annotation **Contents** entry) will help users identify and access specific custom metadata payloads across the majority of PDF applications.

Developers of PDF viewers can process Associated Files for custom metadata serializations by traversing the Document Catalog **AF** array, looking for appropriately named entries with appropriate **AFRelationship** values. Once extracted, the custom metadata files can be validated and processed using other software.

JavaScript Support

ISO 21757-1 *Document management — ECMAScript for PDF — Part 1: Use of ISO 32000-2 (PDF 2.0)* formally standardized JavaScript for PDF 2.0. This standard is based on the JavaScript support that Adobe defined in its "PDF JavaScript API" for PDF 1.7.

Using the mechanisms defined in the API, PDF JavaScript can access the custom metadata files via the document `dataObjects` property, which represents the set of embedded files specified in the document catalog **EmbeddedFiles** name tree (see the **Data** object in ISO 21757-1, §10.11 and Adobe Acrobat SDK documentation). The PDF JavaScript API supports access to the embedded file's filename, media type, **ModDate**, and size; however, PDF JavaScript does not provide a specialized API for Associated Files, so the value of the **AFRelationship** entry cannot be accessed. The PDF JavaScript API also supports reading the embedded file content into a string (see `the util.streamFromString method` in ISO 21767-1, §10.32.3.10, and the Adobe Acrobat SDK documentation).

Accordingly, in order to enable JavaScript functionality, it is recommended to ensure that custom metadata embedded files are included in the document catalog's **EmbeddedFiles** name tree with a known filename, media type, and description (file specification dictionary **Desc** entry or file annotation **Contents** entry). Elementary security and robustness precautions imply that the code should verify that the custom metadata manifest data is as expected before attempting to process it.

Example PDF JavaScript

This example PDF JavaScript snippet is compatible with PDF 1.7 and 2.0, according to the Adobe Acrobat SDK documentation and ISO 21757-1 specifications, respectively. It illustrates how a standardized filename prefix and extension can be used to locate a custom metadata manifest file by iterating through the documents' **EmbeddedFiles** name tree (note that the **AFRelationship** entry is not accessible via standardized PDF JavaScript APIs).

```
var embeddedfiles = this.dataObjects;
var count = 0;
var msg = "";
for (var i = 0; i < embeddedfiles.length; i++) {
    if ((embeddedfiles[i].name.startsWith('ACME_')) &&
        (embeddedfiles[i].name.endsWith('.json'))) {
      msg = msg + '"' + embeddedfiles[i].name + '": "' +
            embeddedfiles[i].description + '"\n';
      count += 1;
      // Load the embedded file stream containing ACME JSON
      acme_manifest = getDataObjectContents(embeddedfiles[i].name);

      // Convert stream to a UTF-8 string
      acme_json_utf8 = util.stringFromStream(acme_manifest, "utf-8");

    // Process the JSON…
      …
     }
}
if (count > 0) {
    msg = 'Found ' + count + ' ACME manifests:\r\n' + msg;
    app.alert({ cMsg: msg, cTitle: "ACME Detection", nIcon: 3 });
}
```

PDF portable collections

A PDF portable collection (or, simply, a "collection") is a special container-like PDF file that packages multiple embedded files. Terminology varies by application, but common terms for PDF portable collections include "portfolios", "packages", or "binders".

Collections provide numerous advantages compared with ZIP or similar archives of files, as the author can:

- Define a cover document with custom content.
- Define a preferred layout and initial view for optimal user navigation.
- Define additional custom attributes of each file in the collection.
- Define the sort order of files.

The files contained in a PDF collection can be of any format – they do not have to be other PDF files, although this is a common scenario. The collection feature is designed in a backward-compatible manner so that if a given PDF viewer doesn't support collections, the author can have some assurance that a file list will nonetheless be presented to end users.

Thus, a PDF collection (ISO 32000-2:2020, §12.3.5) can encapsulate PDF documents, custom metadata manifests, schemas for custom metadata, or any other file. The pages in the container PDF may contain instructions or an explanation for the entire collection package.

The use of a PDF collection can be helpful when each PDF in the collection is small compared to the custom metadata manifest; thus, repeatedly embedding the large custom metadata into each small PDF becomes inefficient. The packaging of multiple files into a PDF collection offers benefits for complex or large custom metadata manifests, enabling them to be loosely associated (via the collection PDF package) with multiple documents across diverse file formats, or where the workflow is primarily driven by custom metadata rather than the PDF.

PDF/A Requirements

PDF collections that include custom metadata manifest files or custom metadata schema files are thereby limited to PDF/A-3 (for PDF 1.7) and PDF/A-4f (for PDF 2.0).

Application and JavaScript support

PDF application support for PDF collections is currently limited. However, fallback support in most PDF viewers via embedded file attachment navigation panes ensures that basic functionality, such as listing or extracting files, is available to end-users.

JavaScript support for PDF Collections is the same as for Associated Files (see above) - by iterating through the list of files in the container PDF **EmbeddedFiles** name tree and processing for a standardized filename and extension, custom metadata manifest files in a collection can be located, extracted, and processed.

Private PieceInfo Data

The storage of metadata as private data in PDF is achieved using **PieceInfo** dictionaries (ISO 32000-2:2020, §14.5).

> NOTE: The term “page-piece dictionaries”, the title of clause 14.5, no longer reflects the full capabilities, as **PieceInfo** dictionaries can be associated with an entire document via the document catalog or associated with a page(s).

The use of **PieceInfo** is appropriate only for private metadata used within closed workflows.

PieceInfo dictionaries use the standard PDF objects expressed in PDF syntax to store their private data under a second-class name (see ISO 32000-2:2020, Annex E). The use of registered prefixes with second-class names avoids naming collisions between different stakeholders. As a consequence, all string and stream objects in private **PieceInfo** data will also be encrypted in encrypted PDF files.

The required **LastModified** date entry in each **PieceInfo** data dictionary can be used to detect when a PDF document has been modified, and the private metadata may thus be invalidated.

```
<<
   /ACME_PrivateData <<
      /LastModified (D:202506…)
      /Private <<
         /SomeKey (some private data as a string object)
         …
      >>
   >>
   /WXYZ_Corporation <<
      /LastModified (D:202412…)
      /Private <<
         /SomeKey /DifferentDataAsNameObject
         …
      >>
   >>
   …
>>
```

PDF/A requirements

Although **PieceInfo** entries may be present in PDF/A files, due to their private nature, they are not recommended for intentional preservation purposes. XMP metadata or Associated Files are more appropriate as they clearly expose the presence of custom metadata to preservationists.

Application and JavaScript support

PieceInfo data is not exposed via the PDF JavaScript APIs and thus cannot be accessed by either document-level or application-level JavaScript. Due to its intended use as *private* data, conventional viewing applications also do not provide access to this information.

Other methods

PDF supports other means of including custom metadata, which may be more appropriate for a small or isolated number of custom data points. All these methods are permitted in PDF/A conforming documents.

Many existing data formats that can occur in PDF streams, such as font programs, images, and ICC color profiles, may also contain their own custom embedded metadata. Refer to the appropriate specification(s) (as listed in ISO 32000-2:2020 clause 2, Normative References) for limitations and technical details for each specific format.

Second-class names

PDF is an extensible page description language that allows publishers to include additional information, such as custom metadata. This includes the use of 2nd-class names as dictionary keys that are prefixed with registered developer-specific entries, as described in ISO 32000-2:2020, Annex E. Since these keys are developer-specific, conventional viewing applications and PDF JavaScript APIs do not provide access to this information.

The PDF specification also defines various objects that support customization through the use of second-class names as the value of certain keys, for example, as the value of **AFRelationship** for Associated Files (see above) or the O entry for custom structure attributes (see below).

Document information dictionaries

The document information dictionary (ISO 32000-2:2020, §14.3.3) was introduced with PDF 1.1 and is largely deprecated in PDF 2.0, with XMP metadata streams now being the preferred mechanism for all metadata.

Document information dictionaries, however, can contain custom entries so long as these entries are text strings. Such custom entries are best kept as short, human-understandable text and may be visible with some legacy PDF software.

Document Parts and Document Part Metadata (DPM)

Document Parts are a new feature in PDF 2.0 (ISO 32000-2:2020, §14.12) that was adopted from PDF/VT-2 (ISO 16612-2:2010, based on PDF 1.6). It enables custom data to be associated with specific page ranges expressed using PDF objects. Although typically associated with job ticketing for commercial variable data printing workflows, the Document Part Metadata

dictionary (ISO 3200-2:2020, §14.12.4.2) can express custom metadata within certain constraints as detailed in ISO 32000-2.

Specialized PDF viewing applications may provide support for Document Parts and DPM; however, there is no standardized support via the PDF JavaScript APIs.

Custom structure attributes

Within the logical structure tree of a document, structure elements may contain structure attributes with an owner (value of the **O** entry in ISO 32000-2:2020, Table 360) of either *NSO (new in PDF 2.0)* or a second-class name. This capability enables the association of custom attribute data in tagged PDF documents.

Advanced PDF viewing applications may support custom structure attributes in their presentation of the structure tree; however, there is no standardized support via the PDF JavaScript APIs.

Structure attributes user properties

Within the logical structure tree of a document, structure elements may contain structure attributes with an array of custom user properties (see ISO 32000-2:2020, §14.7.6.4). User properties are restricted to text strings, numbers (integers or reals), and boolean values expressed as PDF objects. Such documents also need to indicate the presence of user properties by setting the UserProperties entry to true within the document's mark information dictionary.

Advanced PDF viewing applications may support structure attribute user properties in their presentation of the structure tree; however, there is no standardized support via the PDF JavaScript APIs.

Bibliography

Archival PDF,
https://pdfa.org/archival-pdf/

“Understanding Private Data in PDF/A” whitepaper,
https://pdfa.org/resource/understanding-private-data-in-pdf-a/

“Files inside PDF”,
https://pdfa.org/files-inside-pdf/

ISO 16684 series of XMP standards,
https://pdfa.org/resource/iso-16684-xmp/

PDF Declarations,
https://pdfa.org/resource/pdf-declarations/

“PDF 2.0 Application Note 002: Associated Files”,
https://pdfa.org/resource/pdf-2-0-application-note-002-associated-files/

XMP namespace definitions,
https://developer.adobe.com/xmp/docs/XMPNamespaces/

Third-party XMP standards,
https://www.adobe.com/products/xmp/standards.html