

Application Note

Understanding Private Data in PDF/A

2024-06



PDF/A Technical
Working Group

© 2024 PDF Association – pdfa.org

This work is licensed under CC-BY-4.0 © ⓘ

Copyright © 2024 PDF Association or its licensors.

This work is licensed under the Creative Commons Attribution 4.0 International License.

To view a copy of this license, visit
<http://creativecommons.org/licenses/by/4.0/>

or send a letter to
Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

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

E-mail: copyright@pdfa.org
Web: www.pdfa.org

Published in Germany and the United States of America

Vendors own their respective copyrights wherever they are mentioned. The information contained in this document is provided only as general information, which may be incomplete or outdated. Users of this document are responsible for independently verifying any and all information. Any mention of companies or products does not imply endorsement or support of any of the mentioned information, services, products, or providers.

Table of Contents

1. Background	1
1.1. Private data is everywhere, and always has been	1
1.2. Private data in PDF	2
1.3. ...and within other formats nested within a PDF	2
1.4. Static vs. Interactive PDF	3
1.5. Errors are not “private data”	3
2. PDF is extensible by design	4
2.1. Page-piece dictionaries	4
2.2. Custom encryption	5
2.3. Custom annotations	5
2.4. Compatibility operators	5
2.5. Private digital signature handlers	5
2.6. Private tags in marked content	6
2.7. Private hinting for linearization	6
3. Extending PDF	6
3.1. 1 st class keys – “the specification”	6
3.2. 2 nd class keys – “developer-specific”	7
3.3. 3 rd class keys – “private”	7
3.4. Extensions dictionaries	7
3.5. What not to do: vendor-specific rendering	8
4. PDF processors	9
4.1. Processor requirements	9
4.2. Validation, why it matters	10
5. Questions and answers about private data in PDF/A	11
5.1. Q & A: Does “containment” imply “use”?	11
5.2. Q & A: Private data in PDF/A-4	12
5.3. Q & A: Specific examples of private data	13
6. Conclusion	16
Appendix A: PDF/A conformance levels	17
Appendix B: ISO extensions, clarifications and errata	18
Acknowledgements	19

1. Background

PDF/A-1 is based on the Adobe PDF 1.4 Reference, PDF/A-2 and PDF/A-3 are based on ISO 32000-1:2008 (PDF 1.7), while PDF/A-4 is based on the latest PDF 2.0 standard, ISO 32000-2:2020. All parts of PDF/A have the same goals and follow the same principles, with the main differences arising due to PDF's evolution.

PDF/A defines various conformance levels with distinct technical requirements to suit a variety of cases. PDF/A-3 introduced a new file-format feature while PDF/A-4 modernized the model for PDF 2.0 and added support for engineering content. See the appendix, "History of PDF/A versions".

This article makes reference to the latest PDF 2.0 standard (ISO 32000-2:2020) for all PDF features as it contains, in many cases, improved or updated descriptions over previous PDF specifications, however the guidance in this article will generally apply to all parts of PDF/A.

1.1. Private data is everywhere, and always has been

From comments to internal workflow metadata, attachments, retired features, proprietary extensions, and more, almost all file formats allow for the inclusion of some forms of information that aren't fully described by the format specification and which may not be obvious or even visible to end users. Although unspecified data is routine, a software application can only attend to data that it was specifically written to support, and has little choice but to either ignore information that it doesn't recognize or generate an error message.

Some common examples of private data:

- The TIFF and JPEG image formats, as well as ICC profiles and many other formats, use tags to provide support for extensions and private data. The [US Library of Congress Sustainability of Digital Formats "TIFF Tags" resource](#) illustrates that baseline tags (i.e., what is formally specified), extended tags, private tags, proprietary tags and private IFDs are all commonplace in TIFF image files. The popular [exiftool](#) utility [lists various extensions to JPEG tags](#) from both private organizations and government.
- Any SGML-based file format, such as HTML and XML, can have comments (`<!-- ... -->`), custom XML tags, and custom XML attributes that are only meaningful to specific implementations.
- The WhatWG "Living HTML" specification acknowledges the problematic scenario of [custom elements](#) (custom HTML tags) as follows: "*Although authors could always use non-standard elements in their documents, with application-specific behavior added after the fact by scripting or similar, such elements have historically been non-conforming and not very functional.*" and have now standardized support for [custom data attributes](#) that "*.. are intended to store custom data, state, annotations, and similar, private to the page or application, for which there are no more appropriate attributes or elements*".
- Even the ubiquitous Unicode defines [Private Use Areas \(PUA\)](#) for use by third parties.

As seen from these examples, unspecified data is commonly referred to via terminology such as "private data", "custom data", "proprietary data", extensions, etc. It is therefore not surprising that such data also occurs in the PDF context.

1.2. Private data in PDF

“Private data” is not formally defined in any PDF specification or standard. However, in the PDF community, “private data” is a generic term commonly used to describe data contained in a PDF file that is unexpected (i.e., in an undocumented location) or not fully described in the ISO standards for PDF, regardless of origin or intent. Private data in PDF includes anything that is not publicly defined, such as unknown keys and/or values in dictionaries, unknown operators, unknown compression formats, or comments. Private data may be used by a given vendor’s software – with no expectation of use beyond that software – or may simply be present informatively. Known data types, including JavaScript, that are stored in unknown or private locations, are considered private data. Although private data is permitted in PDF/A subsets, PDF/A requires that private data is ignored when rendering static content (see “[Static vs. Interactive PDF](#)”, below).

In this article, we'll define 'private data' as any data, whether in binary or text form, that is allowed to exist within a file format but lacks a specific definition or specification within the file format's documentation. Thus a file used by processor A in a private workflow might include “private data” that is not understood by processor B.

This definition specifically excludes data in embedded file streams (ISO 32000-2, 7.11.4), commonly known as “attachments”. Attachments are specified using multiple PDF objects and data structures, meaning that PDF software understands how the attachment payload is embedded within the encapsulating file format, as well as meta-information such as a filename, a MIME type, etc. PDF software expects that attachments may be present, but may or may not support presenting such attachments to users. So, from the perspective of the encapsulating PDF file format and software, the attachment may be *arbitrary data*, but it is not *private data*.

PDF was designed to accommodate private data since the beginning:

- page-piece dictionaries (introduced in PDF 1.3 via the **PieceInfo** key and documented in ISO 32000-2:2020, 14.5);
- the PDF second and third-class name system defined in Annex E “*Extending PDF*” of ISO 32000-2;
- custom annotation handlers (ISO 32000-2:2020, 12.5.1);
- custom crypt filter handlers (e.g. as enabled by PDF 2.0’s “Unencrypted wrapper document” feature, ISO 32000-2:2020, 7.6.7);
- custom compress filters (e.g., this [proposal](#) for a private “BZIP2” filter);
- extensions formally declared in PDF files via Extensions dictionaries (introduced with PDF 1.7 / ISO 32000-1:2008), or
- new operators occurring between the **BX/EX** compatibility operators in PDF content streams (see ISO 32000-2:2020, Table 33 and further discussion below).

1.3. ...and within other formats nested within a PDF

PDF is a container format. The PDF data model incorporates both data encoded in various other formats as part of the basic syntax (e.g., certain Unicode encodings are allowed in PDF text string objects), and as directly embedded formats (e.g., [JPEG](#) compressed Image XObject using **DCTDecode** data, or embedded [ICC profiles](#) as **ICCBased** color data). In many cases the

embedded data can be directly extracted, saved to a separate file, and subsequently processed using non-PDF software without further modification.

Embedded data can also include its own internal private data according to the format's own specifications, and of course, recursive nesting is possible. Encountering custom JPEG and ICC tags is a common scenario, so it is logical to conclude that private data in nested encapsulated formats in PDF is also common. For example, a PDF file might contain a JPEG image with an embedded ICC color profile, where both the JPEG and ICC could each contain their own private data tags according to their own specifications.

1.4. Static vs. Interactive PDF

When considering the role of private data in PDF, and especially with respect to PDF/A, it's vital to understand what's meant by "static PDF" and "interactive PDF", as PDF/A's requirements are specific to the former and – in general – don't address the latter.

1.4.1. Examples of "static" PDF

- PDF files without annotations
- PDF with annotations in their default state (e.g., an unpressed button, the default view of a unactivated 3D annotation)

1.4.2. Examples of "interactive" PDF:

- PDF with annotations that are no longer in the default state due to user interaction (e.g., a widget annotation with non-default options selected or text entered, an edited Text annotation, a played movie annotation, an activated 3D annotation).
- PDF files including Actions, such as GoTo actions

1.5. Errors are not "private data"

Although general purpose PDF software often successfully opens files with certain types of errors (generally without informing the user, or by altering appearance or behavior), those files are nonetheless invalid. Processing and rendering of PDF/A files with such errors is beyond the scope of both PDF and PDF/A standards, as it requires implementation-dependent processing.

Examples of errors include:

- file structural errors (e.g. bad or missing startxref data, bad cross-reference table data, bad trailers);
- syntax errors (e.g. missing or misspelled PDF keywords (e.g. keyword "true" cannot be misspelled as "True" or "endstream" as "eendstream"); lexical or whitespace errors);
- additional parsing behaviors beyond what is defined in the ISO standards for PDF (e.g. UTF-32 support; alternate key spellings; missing required keys; support for alternate key value data types (such as a PDF string as well as a name); alternate date string formats); or
- other error recovery or software behavior that violate ISO standards for PDF.

For error-free PDF/A files, the behavior of PDF/A conforming software is completely determined by PDF and PDF/A standards.

2. PDF is extensible by design

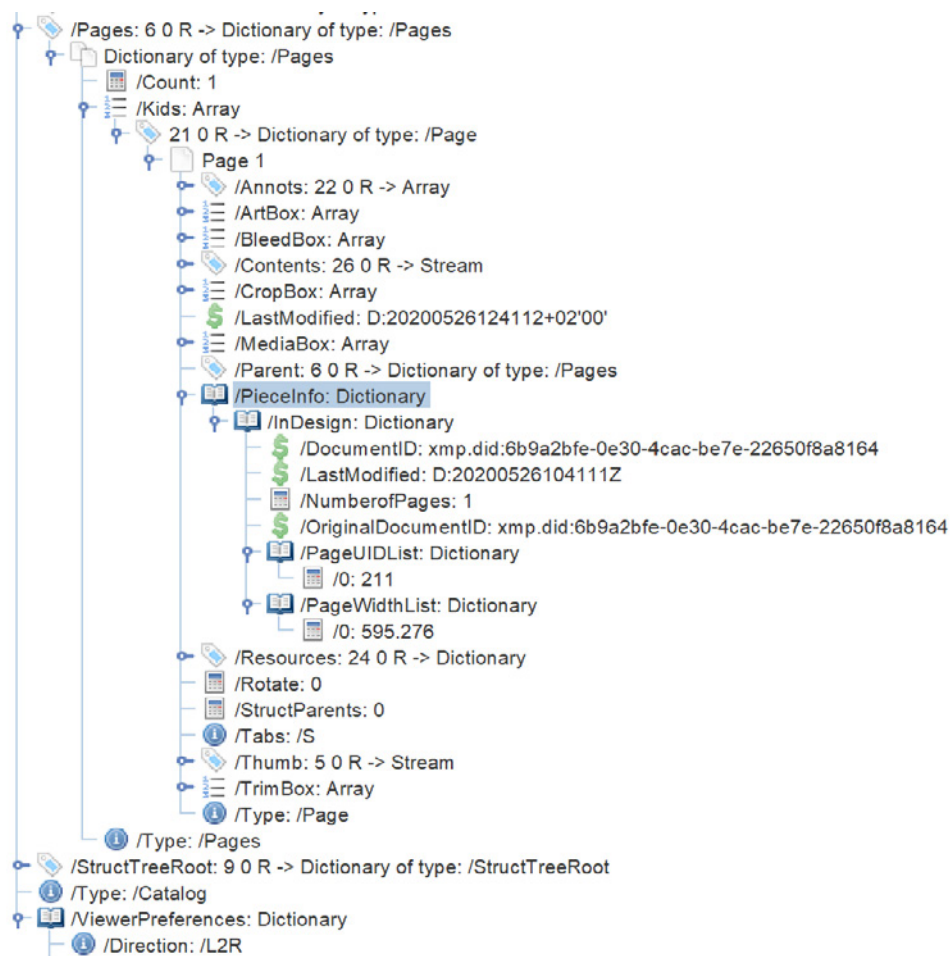
PDF, like TIFF, JPEG, and many other file formats, has always been extensible by design. Many PDF features are specifically defined to be extensible by vendors (such as custom annotations or proprietary crypt filters), or allow various unconstrained data (e.g. choice of multimedia assets or embedded file attachment formats), often by simply permitting key values that are not explicitly constrained in the PDF specification, along with some high-level guidance on how software might handle unexpected values.

PDF standardizes a few mechanisms that explicitly support the concept of private data.

2.1. Page-piece dictionaries

Also known as **PieceInfo** data, page-piece dictionaries are defined by clause 14.5 in ISO 32000-2:2020 and are the documented method “... to hold private PDF processor data”. Note that this data structure is historically named and is no longer constrained to be associated only with pages.

Note that the example shown below does not use the **Private** key as specified in the PDF specifications.



In his 2018 keynote presentation “PDF State of the Union”, Adobe’s Leonard Rosenthol reported that 10% of all ~1.5 billion PDF files opened during the month of April 2016 by the ~1 billion installations of Adobe Acrobat and Reader included “Page Piece Information” indicating widespread usage of that particular class of private data in PDF at that time. More recently, the SafeDocs “PDF Observatory” identifies that 4.4% of PDFs in a corpus of over 1M PDFs from CommonCrawl had **PieceInfo** data.

2.2. Custom encryption

PDF’s crypt filters were designed to specifically permit extension by vendors, including addition or encryption of related private data:

“... the decode parameters dictionary may include entries that are private to the security handler” (ISO 32000-2:2020, 7.4.10) and “... a security handler may choose to encrypt any objects that are private to itself.” (ISO 32000-2:2020, 7.6.2).

2.3. Custom annotations

PDF’s annotations model allows for new or unknown types of annotations to be rendered on a page by unaware PDF software:

“A PDF processor shall provide annotation handlers for all of the conforming annotation types. The set of annotation types is extensible. An interactive PDF processor shall provide certain expected behaviour for all annotation types that it does not recognise ...” (ISO 32000-2:2020, 12.5.1).

2.4. Compatibility operators

The **BX/EX** compatibility operators introduced with PDF 1.1 are specifically defined to encapsulate non-standardized operators:

“Ordinarily, when a PDF reader encounters an operator in a content stream that it does not recognise, an error shall occur. A pair of compatibility operators, BX and EX (PDF 1.1), shall modify this behaviour (see “Table 33 — Compatibility operators”). ... They bracket a compatibility section, a portion of a content stream within which unrecognised operators shall be ignored without error. This mechanism enables a PDF processor to use operators defined in later versions of PDF without sacrificing compatibility with older applications.” (ISO 32000-2:2020, 7.8.2).

2.5. Private digital signature handlers

Digital “signature handlers may add private entries of their own” in a PDF signature dictionary (ISO 32000-2:2020, 12.8.1).

“To avoid name duplication, the keys for all such private entries shall be prefixed with the registered handler name followed by a PERIOD (2Eh)” (ISO 32000-2:2020, 12.8.1). Digital signature “seed value dictionar[ies] may also include seed values for private entries belonging to multiple handlers. A given handler shall use only those private entries that are pertinent to itself and ignore any other private entries.” (ISO 32000-2:2020, 12.7.5.5).

2.6. Private tags in marked content

Property lists of the marked content operators **DP** and **BDC** are “*dictionary[ies] containing information (either private or of types defined in this document) meaningful to the PDF processor*” (ISO 32000-2:2020, 14.6.2). The example below shows a non-standardized **PlacedPDF** tag as an operand to the **BDC** operator:

```
/PlacedPDF /MC0 BDC
q
609.448 0 581.103 841.89 re
W
n
q
/GS1 gs
597.4631 0 0 844.2156 608.42065 -1.1631773 cm
/Im0 Do
Q
Q
q
1020.524 45.685 133.886 31.18 re
W
```

2.7. Private hinting for linearization

Linearization supports private hint tables:

“A PDF writer may add additional hint tables for PDF processor-specific data. A generic format for such hint tables is defined; see F.4.5, “Generic hint tables” Alternatively, the format of a hint table may be private to the PDF processor; see Annex E, “Extending PDF” for further information.” (ISO 32000-2:2020, Annex F.4.1).

3. Extending PDF

Annex E “*Extending PDF*” of ISO 32000-2 describes the system of PDF name classes. By following the rules in Annex E, different developers’ extensions, data for private workflows, and future versions of PDF can co-exist without colliding. This concept is similar to how [HTML custom-data attributes](#) are specified to start with “**data-**” and must not contain any uppercase ASCII characters, thus avoiding collisions with attributes defined in future HTML specifications.

Annex E formally defines three classes of keys to ensure that dictionary keys do not collide.

3.1. 1st class keys – “the specification”

1st class keys are defined as “*All names defined in ISO 32000 specifications are first-class names (that is, they are not preceded by a second-class name prefix). First-class names and data formats are fully defined in official ISO publications and are available for all developers to use*”. Keys introduced by ISO TS extensions are thus all 1st class keys.

3.2. 2nd class keys – “developer-specific”

2nd class keys begin with a [registered](#) 4 character unique, publically available developer-specific prefix. Use of this prefix allows developers to avoid name collisions and allows extensions from all parties (ISO and vendors) to happily co-exist. For example, all TS extensions defined by ISO working groups use the registered developer prefix “[ISO_](#)” whereas Microsoft has registered “[MSFT](#)” as its prefix. Specific exceptions are permitted for additional metadata keys stored in document information or thread information dictionaries.

3.3. 3rd class keys – “private”

3rd class keys begin with “[XX](#)”. The specification requires that they “... *may be used only in PDF files that are part of an internal process between writer and processor in order to avoid conflicts with third-class names defined by others*”. Not unsurprisingly for private (internal use only) PDFs, there were no files with a 3rd class key in the 1M PDF file corpus of public documents in the SafeDocs “PDF Observatory”.

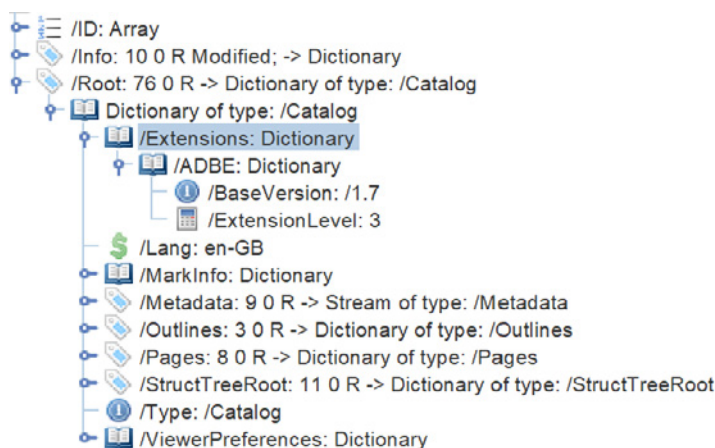
Commonly encountered 2nd class names include Adobe’s [/ADBE_FillSignInfo](#) and [/ADBE_FillSign](#) data structures related to their electronic signing workflow; [Apple’s /AAPL:Keywords](#) [custom keywords array of text strings](#) added to the Document Information Dictionary (which decomposes the standardized **Keywords** text string entry into an array); and Enfocus’ applications storing private data beneath a [/FICL:Enfocus](#) key added to the Document Catalog as disclosed in their US patent [US7783972B2](#). In the 1M PDF file corpus in the SafeDocs “PDF Observatory”, about 1.3% of PDF files contained a dictionary key that started [/AAPL](#), just under 1% contained a key that started [/ADBE](#), and 0.2% contained a key that started [/FICL](#).

Keys that do not conform to these three classes are “private data”. An example includes various “[PTEX](#)”-prefixed keys added by the [pdfTeX LaTeX plugin](#). PTEX keys use a “.” separator between the prefix and the key suffix which is non-standard. In the 1M PDF file corpus in the SafeDocs “PDF Observatory”, 0.6% of PDF files contained a dictionary key that started [/PTEX](#).

3.4. Extensions dictionaries

Extensions dictionaries have existed since PDF was originally standardized in ISO 32000-1:2008 (PDF 1.7). The feature provides a means for developers to declare new features on top of the standardized PDF format. ISO has not published any extensions for PDF 1.7; the working groups instead focused on defining and supporting PDF 2.0, however some vendors do utilize the original extensions mechanism as defined in PDF 1.7.

An example of the use of Extensions dictionaries include Adobe’s [/ADBE](#) extensions defined in “*Adobe supplements to ISO 32000-1, BaseVersion 1.5, Extension Level 3*” and “*Adobe supplements to ISO 32000-1, BaseVersion 1.5, Extension Level 5*” documents (see the [PDF Association’s PDF specification archive](#)).



In the 1 million file PDF corpus in the SafeDocs “PDF Observatory”, about 0.6% of PDF files included an extensions dictionary (although oddly, 3% of these files claimed to conform to a PDF version prior to PDF 1.7 that does not officially support this feature!).

All new extensions defined by ISO TS publications are defined only for PDF 2.0 (ISO 32000-2). The Adobe Extensions to PDF 1.7 (ISO 32000-1:2008) were largely adopted by the ISO committee and are now incorporated into PDF 2.0 in a vendor-neutral manner as first class PDF data structures and without the need for Extensions dictionaries.



3.5. What not to do: vendor-specific rendering

Developers’ proprietary extensions can alter the rendered appearance of a PDF document. Although **very strongly discouraged** because of the different rendering experience (violating PDF’s prime directive), software can use this data when rendering “general PDF” with a general-purpose PDF viewer.

3.5.1. Example: private drop shadow effect

A major PDF producer once used an extension to the PDF graphics state parameter dictionaries that enables a Gaussian blur drop shadow effect that is common in that producer’s user interface.

Support for this private undocumented feature still exists in their general PDF viewers, thus creating noticeable visual differences for users across platforms.

Vendor-specific rendering	Vendor-independent rendering
	

More recently this producer redesigned their drop shadow feature to make it compatible with standardized PDF software ensuring a common multi-platform experience. This compatibility is achieved by providing a pre-rastered Image XObject of the dropshadow (painted with the standardized **Do** operator) enclosed within a marked content sequence (using the standardized **BDC** / **EMC** operators) with an associated property list (see ISO 32000-2:2020, clause 14.6). In this way, the producer's PDF editors might be able to recognize the idiom and automatically update the drop shadow if the graphic content is edited, but the rendered appearance is now built around standardized methods, and is reliable across all platforms.

4. PDF processors

4.1. Processor requirements

PDF subset standards consist primarily of file format requirements, though also include a series of explicit processor requirements, and any discussion about private data and valid or invalid data structures in PDF is obliged to consider the software which processes PDF files.

In ISO terminology, PDF software is referred to as a “PDF processor”:

A PDF processor is any software, hardware or any other active agent that writes, reads, updates or otherwise processes a PDF file which conforms to this document, and does so in a manner that conforms to this document. (ISO 32000-2:2020, 6.3.1)

ISO standards for PDF also include the terms “interactive PDF processor” and “non-interactive PDF processor” to describe classes of PDF software. Examples of “non-interactive PDF processors” might include printer RIPs, server or batch processing software, whereas desktop viewing software is typically interactive. “PDF writers” and “PDF readers” are two other orthogonal classes of “PDF processor”, although, of course, a single piece of software may be both a “PDF writer” and a “PDF reader”.

File format requirements specify *what* a valid PDF file must or must not contain, while processor requirements are additional requirements for *how* information in a PDF file is to be handled by conforming software. Processor requirements in PDF's subset standards are generic and do “... *not prescribe any specific technical design, user interface or implementation details of conforming processors*” (ISO 19005-4:2020, 5.2).

File format requirements apply equally to all PDF software:

- PDF writers, so that they only create valid PDFs, and
- PDF readers, so that they know what data they need to be able to handle when reading a valid PDF, and thus when data is invalid or not covered by any specification it is treated as private data.

PDF's subset standards, such as ISO 19005 (PDF/A), include the key term “conforming PDF processor”, which is distinguished from generic “PDF processors”. “Conforming PDF processors” are stricter than general PDF processors; the distinction lies between complying with the general file format and processor requirements described in ISO 32000 vs. additionally complying with the detailed and specific file format and processor requirements defined in PDF subset standards such as PDF/A.

In most cases the PDF specification does not prohibit the use of private data, allowing general “PDF processors” to provide additional functionality or features. However, if software claims to be a “conforming PDF/A processor” then it must adhere to all of PDF/A’s restrictions on processors, in particular, ignoring private data when performing page rendering.

PDF/A files do not require that only “conforming PDF/A processors” are used to view them. Such a provision would negate the compatibility achieved by PDF/A being a formalized *subset* of general PDF! However, a conforming PDF/A processor is recommended when an end user wishes to utilize PDF/A’s processor restrictions when viewing PDF/A files.

General-purpose PDF software is likely to supply a conforming PDF/A processor mode only as an option, and not as its default, in which case the viewer cannot be assured that the rendering presented is the same one that would be seen when using a “conforming PDF/A processor”. Such differences in rendering are due to a “general PDF processor” not doing things that are necessary for PDF/A (such as use of the OutputIntent for colors) as well as choosing to perform various unspecified activities, including recovery, fix-ups, private data, proprietary extensions, user overrides, etc., that are prohibited for conforming PDF/A processors.

Because using PDF/A’s processing rules to display PDF/A files is optional (except for software deliberately operating as a PDF/A processor), some vendors have chosen to implement “modes” – switching between viewing in “PDF processor” or “conforming PDF/A processor” modes depending on PDF metadata or the user’s choice. PDF standards never define – or even suggest – such detailed software design choices so implementers are free to support their users however they wish. Depending on your needs it may be critical to know if and when your software is a “conforming PDF/A processor” (and thus, is configured to deliver the reliable static appearance that PDF/A guarantees), or is just a general “PDF processor” without those guarantees.

4.2. Validation, why it matters

PDF/A self-declaration does not necessarily imply that the rendered appearance of a PDF/A file as seen on-screen will always be based on a strict “conforming PDF/A processor” implementation.

PDF files presumptively (see ISO 19005-4:2020, 5.1) self-declare their conformity with PDF’s subset standards and conformance levels (such as PDF/A-2b, PDF/A-3u, or PDF/UA-1) through the use of document level XMP metadata. However, since changes to the file by software that isn’t PDF/A-aware may result in a non-conforming file, self-declarations may not be accurate. Accordingly, users can choose to check a file’s conformance and re-assert (or if necessary, remove) the XMP conformance statement by employing a category of software commonly referred to as “validators”.

PDF/A validators check the correctness and conformance of a PDF file against all the requirements of either the file’s declared PDF/A standard or a PDF/A standard selected by the user of the validator. In some industries, this might be referred to as “[preflighting](#)” a PDF.

The specific checks performed by validators are not prescribed in PDF’s subset standards and thus may vary between implementations. The basis of validation checks arises from the interpretation, and subsequent implementation, of the “shall” and “shall not” statements in the standards, so minor variations between vendors can occur. The PDF Association’s

[PDF/A Technical Working Group](#) is the forum for vendors to highlight and discuss any such differences in order to develop a common understanding. Once achieved, this understanding can be articulated for all PDF/A software developers via errata, Technical Notes, referral to the ISO TC 171 SC 2 WG 5 (the ISO committee responsible for PDF/A), or fixes by one or more vendors, as appropriate.

Validation is not discussed in the ISO standards for PDF. Professional-quality PDF software may offer validation functionality (sometimes called “preflighting”), or provide other informative output (such as reporting on private data) to assist in understanding PDF/A files.

Comprehensive validation of PDF/A’s file format requirements is complex and can be slow, so unsurprisingly, “conforming PDF/A processors” are not required to also be validators; in fact most are not. As the “*proper mechanism by which a file can presumptively identify itself as being a conforming PDF/A-4 file*” (ISO 19005-4:2020, 5.1) is the files’ claim of conformance provided in the document-level XMP metadata, conforming software is free to assume that a PDF/A file is, in fact, valid, and process it accordingly, without rigorous (and slow) validation. Software might inform a user that a PDF file declares itself to comply with a specific PDF subset and conformance level based solely on the XMP metadata without running any validation process.

5. Questions and answers about private data in PDF/A

PDF/A’s core value proposition over and above that of “general PDF” is that it supports retention of file data with a reliable static page appearance and (optionally) extractable text. However PDF/A does not define a standardized appearance for interactive content such as form-fields. Likewise, PDF/A does not specify dynamic document behavior beyond page content, or define a runtime environment for complex multimedia such as 3D or movies.

Although the remainder of this article focuses mainly on PDF/A-4, the requirements it discusses are similar to those in earlier editions of PDF/A.

5.1. Q & A: Does "containment" imply "use"?

Understanding the significance of various types of private data in PDF/A requires addressing the question of *containment* (i.e., the mere fact of private data being present in a PDF/A file) as distinguished from the active *use* (processing) of private data by PDF software.

Every valid PDF/A file is a valid PDF file with additional constraints to ensure a reliable static page appearance when used with a conforming PDF/A processor. This objective does not imply that containment of private data is problematic, as all PDF/A conforming processors are required to implement a standardized static viewing environment for page rendering irrespective of any private data that may be present in the file.

General-purpose PDF viewers, such as those commonly found in current-generation web browsers or mobile devices, are not necessarily PDF/A processors. That is, not all PDF viewers are designed to support the reliable standardized processing requirements of PDF/A, so opening a PDF/A file with a viewer that doesn’t support PDF/A’s restrictions will result in non-standardized rendering and/or processing.

5.2. Q & A: Private data in PDF/A-4

5.2.1. Can PDF/A-4 files contain private extensions, as declared via Extensions Dictionaries?

ANSWER: The answer depends on the nature and function of each individual extension.

As with any other feature in PDF not explicitly mentioned in PDF/A, Extensions dictionaries (see ISO 32000, clause 7.12) are valid PDF objects allowed in PDF/A-3 and PDF/A-4 files. However most extensions define new or changed data structures, so each extension must be individually assessed against PDF/A for other constraints or limitations.

Furthermore, since PDF/A requires that *“any data contained in a conforming file that is not described in ISO 32000-2 or this document ... shall not be used to render content on a page”*, PDF/A conforming software must not permit any such extension to alter the appearance of rendered static page content (see ISO 19005, clause 6.1.1).

5.2.2. Can PDF/A-4 files use the new digital signature ISO TS extensions?

ANSWER: Yes. Neither PDF/A nor PAdES restrict the set of algorithms that can be used by digital signatures. However as documented in the ISO TS, the Extensions dictionary must be used to declare the presence of such extensions.

A list of all current ISO TS extensions is provided in [Annex B](#).

5.2.3. Can PDF/A-4 files use the new PDF 2.0 file encryption extensions?

ANSWER: No. PDF/A does not allow any form of file encryption.

5.2.4. Can PDF/A-4 files use ISO/TS 32005 regarding PDF 1.7 and PDF 2.0 namespaces?

ANSWER: Yes. ISO/TS 32005 is not an extension to PDF and does not alter requirements in ISO 32000-2. It rather provides additional guidance on the combined use of PDF 1.7 and PDF 2.0 namespaces. Thus a Tagged PDF 2.0 file conforming to ISO/TS 32005 is also conformant with ISO 32000-2 and thus can also conform to PDF/A-4. However conformance to PDF/A does not require and does not guarantee conformance with ISO/TS 32005.

5.2.5. Can PDF/A-4e files contain RichMedia annotations with assets other than those specified in PDF 2.0 (i.e., such as ISO/TS 24064 STEP or ISO 32007 glTF extensions)?

ANSWER: Yes. ISO 19005-4:2024, Annex B only requires that all file specification dictionaries present in either the **Assets** name tree of a RichMediaContent Dictionary or the **Asset** entry in RichMediaInstance dictionary shall contain a **Subtype** key whose value is a valid [IANA Media Type](#), as well as the **F** and **UF** keys.

5.2.6. Do ISO 32000-2:2020 [errata resolutions](#) apply to PDF/A-4?

ANSWER: It depends. Not all errata apply to PDF/A-4 because of the additional requirements imposed by PDF/A. ISO-approved errata are clarifications and corrections that resolve identified ambiguities and errors in PDF specifications and help achieve reliable rendering. This intent was indicated by the use of an undated normative reference to ISO 32000-2 in ISO 19005-4:2020. All

errata against PDF 2.0 (ISO 32000-2:2020) are carefully considered with respect to its impact on PDF/A-4.

As with all file format specifications errata, it is essential for developers to remain apprised of the current state of the specification to mitigate confusion, minimize differing implementations, and assure long-term integrity and reliable rendering. New PDF functionality is not added via errata.

5.3. Q & A: Specific examples of private data

As discussed in this article, the term “private data” might include the following (*not intended as exhaustive*):

5.3.1. Can PDF/A files contain 2nd class name keys that get added by some application software, using publicly documented and [registered prefixes](#) as per ISO 32000-2:2020 Annex E?

ANSWER: Yes, PDF/A files may contain keys with 2nd class names, but this data “*shall not be used to render content on a page*” by PDF/A conforming software.

5.3.2. Can PDF/A files contain 2nd class name keys, using unknown developer prefixes but otherwise following ISO 32000-2:2020 Annex E conventions?

ANSWER: Yes. PDF/A files may contain keys with 2nd class names of unknown origin, but this data “*shall not be used to render content on a page*” by PDF/A conforming software. Unknown developer prefixes include prefixes privately registered with Adobe prior to ISO standardization of PDF that, for legal reasons, cannot be made public. To avoid collisions and possible loss of data, the owners of such prefixes are strongly encouraged to re-register all their prefixes via <https://github.com/adobe/pdf-names-list>.

5.3.3. Can PDF/A files contain 3rd class name keys (i.e., keys that start with “xx**” as per ISO 32000-2:2020 Annex E) as may occur within organizations that use custom software and internal private workflows?**

ANSWER: Yes, PDF/A files may contain keys with 3rd class names, but this data “*shall not be used to render content on a page*” by PDF/A conforming software.

5.3.4. Can PDF/A files contain unknown values of keys where the PDF specification explicitly permits values beyond what is standardized (e.g. annotation subtypes; custom rendering intents; custom crypt filters)?

ANSWER: The answer depends on PDF/A’s specific limitations for different kinds of PDF objects, PDF/A files may contain such data, but this data “*shall not be used to render content on a page*” by PDF/A conforming software.

Note however that PDF/A explicitly prohibits certain instances: non-standard annotations are not allowed “*Annotation types not defined in ISO 32000-2:—, 12.5.6.1, Table 171 shall not be permitted. Additionally, the **Sound**, **Screen**, and **Movie** types shall not be permitted.*” (ISO 19005-4:2020, 6.3.1) and PDF/A does not allow encryption. All parts of PDF/A require that only the 4 standard rendering intents specified in PDF are used in PDF/A conforming files.

5.3.5. Can PDF/A files contain unknown values of keys where the PDF specification explicitly constrains the value to a known range or a set of values?

ANSWER: No. PDF/A files must not contain out-of-specification data, because a valid PDF/A file must also conform to the requirements of PDF.

5.3.6. Can PDF/A files contain Page-piece dictionaries (also known as [PieceInfo data](#)) with private data as defined by clause 14.5 in ISO 32000-2:2020?

ANSWER: Yes. PDF/A files may contain Page-piece dictionaries that fully comply with ISO 32000 clause 14.5 specification (including all required keys, etc). But this data “*shall not be used to render content on a page*” by PDF/A conforming software.

5.3.7. Can PDF/A-4 files contain stream objects containing unspecified (private) data?

ANSWER: Yes. PDF defines specifications for the content of most stream objects. ICC profiles, JPEG, JPEG 2000, CCITTFax, JBIG2, embedded font programs (TrueType, OpenType), XMP metadata, CMaps, and others must comply with their relevant specifications *and* any additional requirements for their use in both the PDF and PDF/A standards.

PDF allows streams with unspecified content within:

- The value of any 2nd or 3rd class named key of a dictionary (*see above*)
- a page-piece dictionary’s **Private** data structure (*see above*),
- 3D Node **Data** streams (ISO 32000-2:2020, Table 323),
- 3D View Params **Data** streams (ISO 32000-2:2020, Table 345),
- embedded file streams (ISO 32000-2:2020, 7.11.4),
- various web capture streams, and
- various streams associated with digital signatures.

In all these cases, these data streams are not needed or used for static page rendering. On this basis, then, arbitrary stream data for standardized PDF keys is limited to embedded file attachments, which in PDF/A are restricted to PDF/A-3, PDF/A-4e and PDF/A-4f.

PDF/A-4 imposes specific requirements on Embedded Files (which are implemented as PDF stream objects) as defined in ISO 19005-4:2020, 6.9 and PDF/A-4f, Annex A. Furthermore PDF/A-4e imposes additional requirements on 3D streams, but does not otherwise constrain other assets that might be in the RichMedia asset tree.

5.3.8 Does PDF/A conformance cover document navigation user interface elements such as outlines, Layer names, file attachments, etc.?

ANSWER: Yes, the PDF/A specification includes both the file requirements that all PDF strings available to the user shall be Unicode compliant, as well as PDF/A conforming processor requirements to present such strings to the user. This includes destinations of external URLs, outlines, fonts, Layers, and embedded file names.

5.3.9. Does the “reliable rendering” of PDF/A conforming software include dynamic document behavior such as selection and focus effects, interactive forms, or multimedia interactivity?

ANSWER: No. PDF/A ensures reliable *static* page rendering, not non-static rendering or aspects of dynamic document behavior. In addition, PDF/A-4e *“introduces some new directions in archiving non-static content that can be present in PDF documents, such as form fields and ECMAScript. It seeks to preserve more information in the file (by not requiring its removal during the archival process) and puts a greater burden on conforming viewers to ensure that such information does not alter the visual appearance of the file during consumption.”* (ISO 19000-4:2020, Introduction). PDF/A-4e allows JavaScript to be in a PDF/A file but requires that *“... they are invoked explicitly by a user (such as via outlines or buttons). A conforming processor, that is non-interactive, shall not execute them at all.”*

5.3.10. Can PDF/A-4e files contain ECMAScript containing undocumented or proprietary (private) extensions not defined in ISO 21757-1:2020?

ANSWER: Yes. PDF/A-4e *“introduces some new directions in archiving non-static content that can be present in PDF documents, such as form fields and ECMAScript. It seeks to preserve more information in the file (by not requiring its removal during the archival process) and puts a greater burden on conforming viewers to ensure that such information does not alter the visual appearance of the file during consumption.”* (ISO 19000-4:2020, Introduction).

For interoperability, any ECMAScript used in PDF/A-4e needs to be according to the [ECMA-262 specification for “ES2020”](#).

As the introduction to the standard implies, PDF/A-4e does not additionally constrain the ECMAScript (JavaScript) that may be present in PDF/A-4e files. Such PDF ECMAScript data is retained for preservation, as PDF/A-4e does not mandate a standardized interactive execution and runtime environment for interacting with 3D models or other PDF dynamic runtime functionality. However, ISO 21757-1:2020 (ECMAScript for PDF 2.0) does provide this strong recommendation: *“Undocumented properties and methods should not be used”*.

5.3.11. Can PDF/A contain undocumented (private) operators in content streams between BX/EX compatibility operators?

ANSWER: No. All PDF/A standards explicitly state that PDF/A files must not contain any undocumented operators: *“A content stream shall not contain any operators not defined in [the relevant PDF specification] even if such operators are bracketed by the BX/EX compatibility operators.”*

5.3.12. Can PDF/A contain undocumented (private) operators in content streams outside BX/EX compatibility operators?

ANSWER: No. All PDF/A standards explicitly state that PDF/A files must not contain any undocumented operators: *“A content stream shall not contain any operators not defined in [the relevant PDF specification] even if such operators are bracketed by the BX/EX compatibility operators.”*

5.3.13. Can PDF/A contain non-standard (private) operands to standardized content stream operators where the appropriate PDF specification does not explicitly constrain the range and does not also define a method to handle out-of-range values?

For example, PDF standardizes behavior for handling out-of-range index values to Indexed color spaces and unrecognized rendering intents.

ANSWER: No, except where extensibility is provided for, such as the Marked Content operators. All operand values for operators need to comply with the PDF and PDF/A standards.

5.3.14. Can PDF/A contain “legacy” PDF features that were removed from PDF prior to the base standard of the relevant PDF/A standard (e.g., PDF 1.0 supported Passthrough PostScript XObjects, CalCMYK)?

Such legacy features are thus not described in any ISO 32000 standard, whereas deprecated features remain documented in ISO 32000.

ANSWER: Yes, legacy PDF features can occur in a PDF/A file but must not be used for rendering since *“Any data contained in a conforming file that is not described in ISO 32000... or [the PDF/A standard] should be ignored by a conforming processor and shall not be used to render content on a page.”*

5.3.15. Is PDF/A susceptible to malware or vulnerabilities?

ANSWER: Yes. Virtually all computer applications are susceptible to malware and vulnerabilities. Implementation issues caused by bugs (software defects) don’t pertain to the PDF/A file format. Although JavaScript (ECMAScript) is a common PDF feature that’s also a well-known attack vector, PDF/A-4e requires conforming interactive software to demand an explicit user action before invoking any JavaScript (see ISO 19005-4, 6.6.2), thus providing a degree of protection for end users.

6. Conclusion

Although PDF/A’s file format requirements permit including extensions and so-called “private data” within PDF/A files, ISO 19005’s requirements for conforming PDF/A processors do not permit such data to alter the static rendering of page content.

Vendors claiming to have implemented conforming PDF/A processors are obliged to implement these requirements. Likewise, as with any specific file-format, PDF/A’s end-users should ensure that they always use conforming PDF/A processors, and not rely on a general-purpose PDF viewer when viewing PDF/A files. This can be achieved by the alignment of IT and preservation-related policies, such as ensuring the standardized use of particular PDF/A-aware viewers.

Appendix A: PDF/A conformance levels

PDF/A Identifier	Description
PDF/A-1a PDF/A-2a PDF/A-3a	Level A (“accessible”) ensured a well-defined representation of the structure tree also enabling logical reading order.
PDF/A-1b PDF/A-2b PDF/A-3b	Level B (“basic”) focuses on reliable static rendering of page content.
PDF/A-2u PDF/A-3u	Level U (“unicode”) additionally requires reliable Unicode mapping of all textual content.
PDF/A-3 PDF/A-4f	PDF/A-3 and PDF/A-4f add support for embedded file attachments other than other PDF/A compliant files.
PDF/A-4	PDF/A-4 introduced support for PDF 2.0 and incorporated the requirements of conformance level B. It “offloaded” level A to WTPDF and/or PDF/UA-2 .
PDF/A-4e	PDF/A-4e also enabled support of engineering documents with 3D content.

PDF/A-4, 6.2.10.1 states *"The intent of the requirements in [Font clauses in ISO 19005-4:2020] is to ensure that the future rendering of the textual content of a conforming file matches, on a glyph by glyph basis, the static appearance of the file as originally created and, when possible, to allow the recovery of semantic properties for each character of the textual content"*.

Appendix B: ISO extensions, clarifications and errata

ISO Technical Specification (TS) documents provide an efficient pathway for the standardization of new PDF features without the overhead, cost, and delay needed to iterate the full 1000 page PDF standard. ISO TS documents may define either extensions (new features) for PDF 2.0, or provide additional clarification or guidance for existing features.

For extensions that introduce new features, the ISO TS documents also define precisely how PDF files will declare the presence of the new feature using the existing PDF Extensions Dictionary mechanism ([see 3.4](#) and ISO 32000-2:2020, 7.12).

As of Q1, 2024, the following ISO Technical Specifications address a wide range of new capabilities in PDF 2.0, including:

ISO Technical Specification (TS)	PDF Association Community / ISO TC 171 SC 2 WG
ISO/TS 32001 Document management – Portable Document Format – Encryption and Hash algorithm support in ISO 32000-2 (PDF 2.0)	Digital Signature TWG / WG 8
ISO/TS 32002 Document management – Portable Document Format – Extensions to Digital Signatures in ISO 32000-2 (PDF 2.0)	Digital Signature TWG / WG 8
ISO/TS 32003 Document management – Portable Document Format – Adding support of AES-GCM in PDF 2.0	Digital Signature TWG / WG 8
ISO/TS 32004 Document management – Portable Document Format – Integrity Protection in encrypted document in PDF 2.0	Digital Signature TWG / WG 8
ISO/TS 32005 Document management – Portable Document Format – PDF 1.7 and 2.0 namespace inclusion in ISO 32000-2	PDF/UA TWG / WG 9
ISO/TS 24064 Document management – Portable Document Format – 3D data streams conforming to the ISO 10303:242 (STEP AP242) specification	3D PDF TWG / WG 7
ISO/TS 32007 Document management – Portable Document Format – RichMedia annotations conforming to glTF assets	3D PDF TWG / WG 7

In addition to Technical Specifications, the PDF Association [publishes “errata”](#) – corrections against various published ISO standards for PDF technology. These errata address important ambiguities, mistakes, typos, omissions, or other confusing issues that cause differences between implementations if left unaddressed. By resolving such issues the portability and interoperability of PDF documents is ensured.

Acknowledgements

Although this document was developed and approved by the PDF Association's PDF/A TWG, the following individuals had a leading role in its development.

[Kevin De Vorse](#)y, National Archives and Records Administration and Convenor for ISO TC 171 SC 2 WG 5

[Boris Doubrov](#), Dual Lab and co-Chair of the PDF Association's [PDF/A TWG](#)

[Duff Johnson](#), PDF Association CEO and ISO Project co-Leader for ISO 32000

[Kate Murray](#), Library of Congress

[Leonard Rosenthol](#), Adobe, Project Leader for ISO 19005 (PDF/A) and co-Chair of the PDF Association's [PDF/A TWG](#).

[Dietrich von Seggern](#), callas software and PDF Association ISO Liaison Officer

[Peter Wyatt](#), PDF Association CTO and ISO Project co-Leader for ISO 32000