A document that has to last

PDF/A is the profile of PDF for keeping things. It removes everything whose meaning depends on something outside the file - a font installed on the machine, a colour profile named rather than embedded, a JavaScript action, an external stream - so that the bytes are the whole document. This file claims PDF/A-3b, which is why it carries an ICC profile and an XMP packet it would otherwise have no use for.

The three profiles

PdfA1B (ISO 19005-1)

The strictest. PDF 1.4 constructs only, so no transparency, no JPXDecode, no cross-reference stream, and no embedded files at all.

PdfA2B (ISO 19005-2)

Defined against PDF 1.7. Transparency and JPXDecode are allowed. Still no embedded file unless that file is itself PDF/A.

PdfA3B (ISO 19005-3)

As PDF/A-2b, and the only profile that may carry an attachment of any kind - which is what hybrid e-invoices such as ZUGFeRD and Factur-X are built on.

Only the B levels are here

The A levels - PDF/A-1a and its successors - additionally require a full tagged structure tree. That is a different piece of work with a different point to it, and it is the Accessibility demo. A document may claim both: PDF/A-3 says it will still open in fifty years, PDF/UA-1 says it can be read aloud, and neither implies the other.

The claim is checked, not stamped

Setting Options.Conformance does more than label the file. Before a byte is written the writer walks the document and throws on the first rule it can settle by looking - naming the rule and what to do about it. A library that writes pdfaid:part 3 onto a file and leaves the caller to hear from a validator, or from their customer, that it does not conform has made things worse rather than better. The third page of this demo is those refusals, in their own words.

And what a successful save is still not

A validator's verdict. Some real rules are not checked here and are said plainly rather than implied by silence: that a PDF/A-1 document uses no transparency means walking every page's resources, and that no image is JPXDecode means walking every image. Neither is done. What is checked is checked properly; veraPDF has the last word.

The metadata packet

Every fact in the packet also lives in the document information dictionary, and a validator compares the two and complains when they disagree - so the packet is built from the dictionary at save time rather than kept beside it and hoped about. The conformance identifier is the one thing that has no dictionary entry at all, which is why XMP is not optional for a document making a claim.

A namespace of your own needs declaring

AdditionalDescriptions writes what it is given, verbatim, and PDF/A accepts no property whose schema the file has not either predefined or described. So a namespace nobody has heard of - this demo's own, or an invoice format's - is declared in a pdfaExtension:schemas block naming every property before any of them is written. The FacturX demo is that done for real, by PdfPinata.EInvoice rather than by hand.

Written by a document just like this one

Read back out of a probe document built with the same options, saved to memory and reopened. It is the bytes, not a description of them. Note that the packet is left uncompressed: it carries those xpacket markers so a tool can find it by scanning for them without parsing the PDF around it, and a compressed packet is invisible to one. The probe claims conformance and adds nothing of its own, so what follows is the packet a document gets for free.

```
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/">
 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="" xmlns:dc="http://purl.org/dc/elements/1.1/">
   <dc:title><rdf:Alt><rdf:li xml:lang="x-default">A probe</rdf:li></rdf:Alt></dc:title>
   <dc:creator><rdf:Seq><rdf:li>PdfPinata sample app</rdf:li></rdf:Seq></dc:creator>
  </rdf:Description>
  <rdf:Description rdf:about="" xmlns:pdf="http://ns.adobe.com/pdf/1.3/">
   <pdf:Producer>PdfPinata 1.0.0 (https://github.com/PinataLabs/PdfPinata)</pdf:Producer>
  </rdf:Description>
  <rdf:Description rdf:about="" xmlns:xmp="http://ns.adobe.com/xap/1.0/">
   <xmp:CreatorTool>PdfPinata 1.0.0 (https://github.com/PinataLabs/PdfPinata)</xmp:CreatorTool>
   <xmp:CreateDate>2026-09-22T12:20:59+00:00</xmp:CreateDate>
  </rdf:Description>
  <rdf:Description rdf:about="" xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/">
   <pdfaid:part>3</pdfaid:part>
   <pdfaid:conformance>B</pdfaid:conformance>
  </rdf:Description>
 </rdf:RDF>
</x:xmpmeta>



<?xpacket end="w"?>
```

The refusals, in their own words

Each block below is a real exception message, caught from a document built to break one rule and then asked to save. Nothing here is quoted - the text is whatever the library said when this demo was run, so a reworded message reaches this page by itself.

No title

```
A PDF/A document has to have a title, in the document information dictionary and in its XMP metadata alike. Set Info.Title.
```

CMYK with no output intent

```
A PDF/A document has to embed an ICC profile saying what its colours mean, and CMYK numbers mean nothing at all without one — the same four numbers are a different colour on every press. This library supplies sRGB for an RGB document, where the numbers describe themselves, and cannot do the equivalent here. Set Options.OutputIntentIccProfile to the profile your work was made for.
```

Encrypted

```
A PDF/A document may not be encrypted. Either drop the conformance claim or clear SecuritySettings.DocumentSecurityLevel.
```

PDF/A-1 with a cross-reference stream

```
PDF/A-1 is defined against PDF 1.4 and a cross-reference stream is a PDF 1.5 construction, so the two cannot both be asked for. Either claim PDF/A-2 or later, or leave Options.CrossReferenceFormat as Classic.
```

PDF/A-1 asked for a PDF 1.7 feature

```
PDF/A-1 is defined against PDF 1.4, and this document is written as PDF 1.7. Either claim PDF/A-2 or later, or stop asking for the feature that raised the version.
```

The output intent

A PDF/A document using a device colour space has to embed an ICC profile saying what its colours mean, and RGB - the default - is a device colour space. The profile is embedded rather than referenced, and that is the whole point of the rule: naming a well-known profile is exactly what PDF/A exists to stop, since the name means nothing once the machine that understood it is gone.

An RGB document is given one, and that is new

Colours written as RGB by a library nobody told otherwise are sRGB - that is what every reader assumes of them - so a PDF/A document whose ColorMode is Rgb and which names no profile of its own is given PdfOutputIntents.SrgbProfile, and the sRGB condition to name it by. That is a description rather than a guess, which is why it can be done at all. Whatever you set yourself always wins.

CMYK is still refused, and so is Undefined

The same four CMYK numbers are a different colour on every press, so there is nothing true to supply and the writer says so instead - the last refusal on the previous page is that one. ColorMode.Undefined writes each colour as the XColor gave it, so a document may hold RGB and CMYK together and no one profile describes it. Both name what to set.

What that profile is

456 bytes from the Compact ICC Profiles collection, released to the public domain under CC0 - which is what makes it shippable at all. It states the true sRGB primaries and samples the transfer curve at 42 points rather than stating it parametrically, which is where the size goes. ICC version 2 rather than 4, because PDF/A-1 predates version 4 and will not take one, so a v2 profile is the one that serves every part. A document that matters should still embed the profile its colours were actually made in.

Profile size	456 bytes
Device class	mntr (display), ICC version 2.1
Colour space	RGB, PCS XYZ
Tags	desc, cprt, wtpt, rXYZ, gXYZ, bXYZ, rTRC, gTRC, bTRC
Licence	CC0 1.0, public domain - the cprt tag says so itself
/OutputConditionIdentifier	sRGB IEC61966-2.1
/S	/GTS_PDFA1, for every part of PDF/A and not only the first

/GTS_PDFA1 for all three parts

The output intent's subtype names the family rather than the part, so a PDF/A-3 file carries /GTS_PDFA1 too. It reads like a mistake and is not; writing /GTS_PDFA3 is the mistake.

Fonts were never optional

PDF/A requires every font to be embedded, and this library embeds every font with no setting to disable it - so the rule that catches most producers out cannot be broken here. TrueType outlines are subsetted; PostScript (CFF) outlines cannot be and go in whole.