PDF association

IEEE-ISTO PWG, May 18 2022

Understanding 3D PDF

Peter Wyatt, CTO

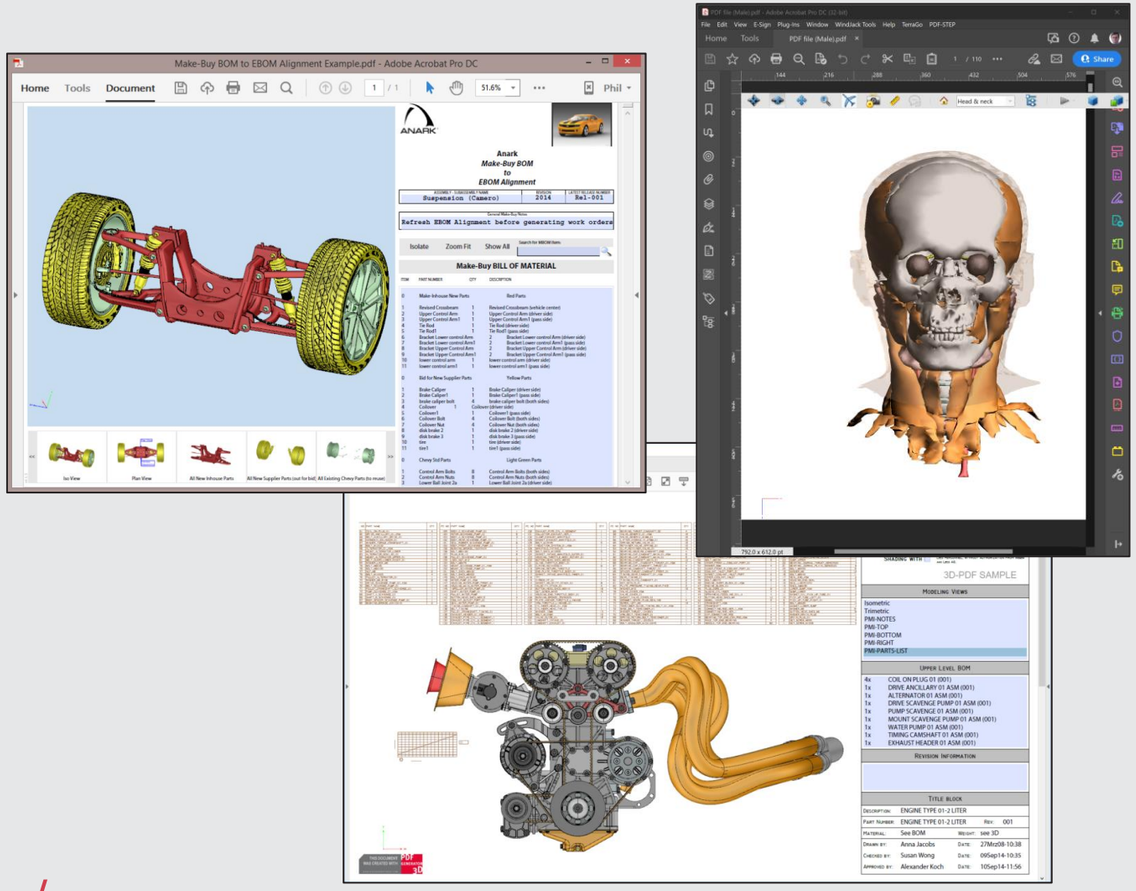
v1.1

Agenda

- 3D formats in PDF
- 3D PDF support
- Assumptions for IEEE-ISTO PWG 3D print use case
- PDF basics
- How 3D models are stored in PDF
  - 3D and RichMedia Annotations
- Future thoughts
- Summary

3D formats and PDF

- **Now**:
  - U3D
  - PRC
  - STEP AP 242
- **In the future**:
  - glTF – *under discussion*



https://www.pdfa.org/resource/3d-formats/

https://www.pdfa.org/3d-pdf-showcase/
https://3dpdfconsortium.org/showcase/
http://anatomy.dongguk.ac.kr/

U3D = Universal 3D

- Defined by ECMA-363, *Universal 3D File Format, 3rd Edition (U3D), June 2006*
  - https://www.ecma-international.org/publications/standards/Ecma-363.htm
  - *U3D 4th edition is **not** supported in PDF*
  - ECMA committee is inactive
- Since PDF 1.6 using 3D annotations
- Since PDF 2.0 using RichMedia annotations
- Registered MIME media type: `model/u3d`
  - Registered by PDF Association
- Cannot depend on external resources, but can have external references
  - https://www.iana.org/assignments/media-types/model/u3d

PRC = Product Representation Compact

- Defined by ISO 14739-1:2014 *Document management — 3D use of Product Representation Compact (PRC) format — Part 1: PRC 10001*
  - Developed by ISO TC 171 SC 2 – *same ISO sub-committee as PDF*
  - Known errata with dated revision update to ISO 14739-1 expected
- Introduced in PDF 2.0 using 3D *or* RichMedia annotations
- Registered MIME media type: `model/prc`
  - Registered by ISO TC 171 SC 2 via PDF Association
- Cannot have external dependencies
  - https://www.iana.org/assignments/media-types/model/prc

STEP = “STandard for the Exchange of Product model data”

- Defined by ISO 10303-242, *Industrial automation systems and integration — Product data representation and exchange — Part 242: Application protocol: Managed model-based 3D engineering*
  - Managed by ISO TC 184 SC 4
- Introduced in a PDF 2.0 extension using RichMedia annotations (*only*)
  - ISO/TS 24064 *Document management – Portable Document Format – 3D data streams conforming to the ISO 10303:242 (STEP AP242) specification*
- Registered MIME media types:
  - `model/step, model/step+xml, model/step+zip, model/step-xml+zip`
- **Can** have external dependencies

3D PDF support

- Many commercial CAD/CAM/ACE packages and applications can view, create, edit and annotate 3D PDF files (U3D, PRC)
- **Free** interactive 3D PDF viewing (U3D, PRC) is supported by multiple vendors:

- No known native support for 3D PDF in browser-based PDF viewers, Apple iOS, or Android platforms
- No vendor has announced STEP support yet (*that I know of*)
  - ISO/TS 24064 passed final ballot recently

Assumptions for 3D print use case

- Need to extract raw 3D model asset files (U3D, PRC, STEP)
  - *And any other embedded dependent asset required by the 3D model asset…*
- 3D print use case does **not** need:
  - Interactivity data, views, camera & lighting angles, animations, model annotations, etc.
- Parsing/processing of 3D models for 3D printing is entirely independent of PDF
  - PDF is a container file format and does **not** define how 3D models or 3D runtimes operate
  - PDF is a vendor neutral file format and does **not** define APIs – that is vendor specific
- PDF does **not** contain printability metadata or job tickets related to 3D models
  - This *may* be inside or referenced by 3D model assets
- You will use a PDF SDK (*so won't discuss PDF parsing & basic data structures*)

Some PDF basics

- PDF is a file format
- PDF is object based
  - Dictionaries, arrays, streams, names, strings, numbers, …
- PDF objects can be reused
- PDF is a container-based format
- PDF document object model (DOM) forms a tree
  - Root of the tree is the Document Catalog dictionary
  - *Certain* objects can be accessed via different paths
- PDF is page-based
- PDF data can be encrypted & compressed
- 3D PDF is about **interactive 3D models**
- ***I will describe what the PDF file format standard defines***
  - ***Not any specific implementation***

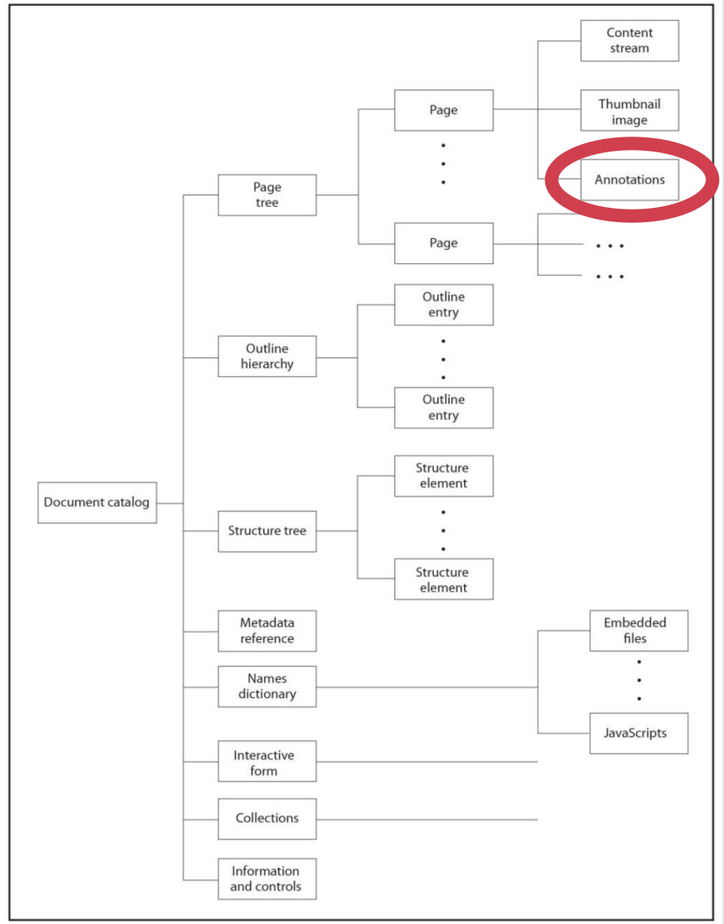


Figure 5 — Structure of a PDF document

3D models in PDF files

- A single PDF file can have many 3D models of any type
- Annotations are associated with individual pages
  - Thus 3D models are associated with individual pages
    - Users perceive an interactive 3D model as "*it's on page 3*" but are unaware of the 3D format
- PDF objects containing the 3D models can be reused across pages
- 3D model assets are stored as PDF stream objects
  - Likely compressed and/or encrypted
- PDF supports embedded external dependencies required by 3D models
  - As other embedded data streams inside the PDF

My terminology

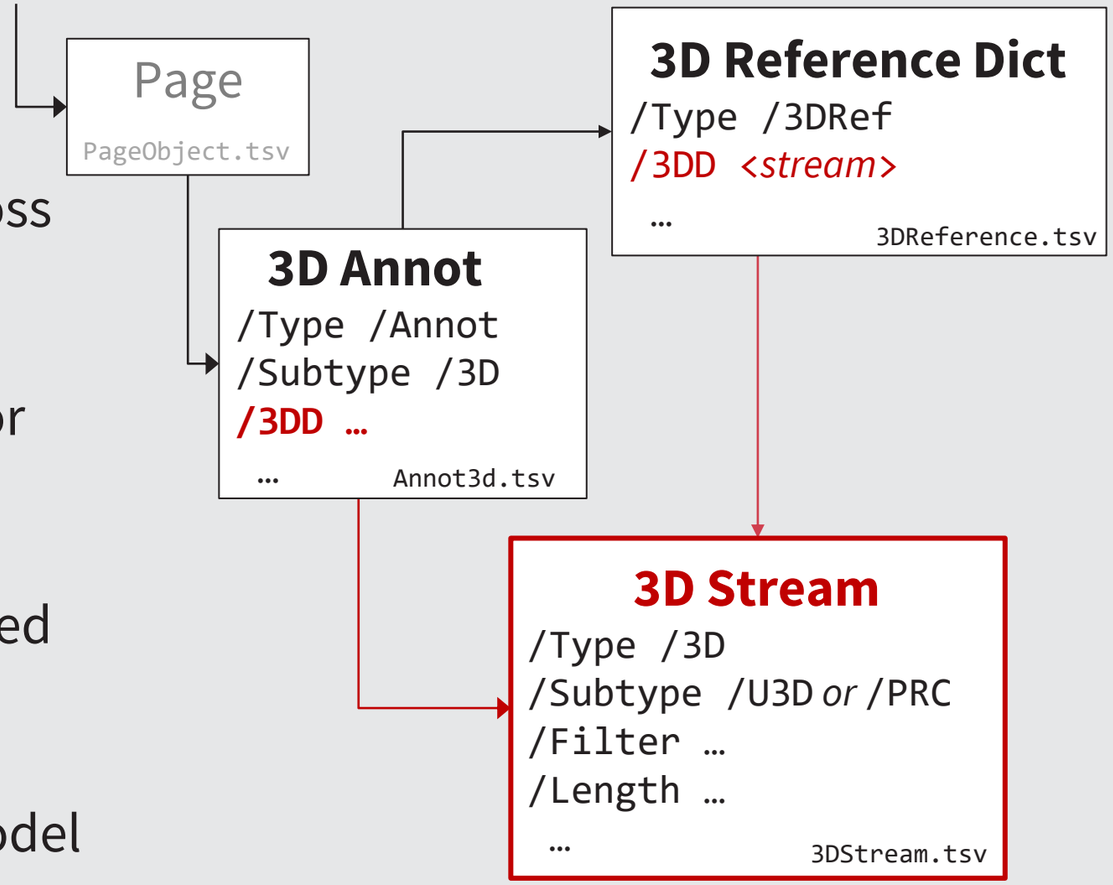
- 3D model = everything that makes up an interactive 3D PDF model
  - May include JavaScript, 3D projection annotations, 3D views, camera views, …
- 3D model asset = the main U3D, PRC, STEP data stream
- 3D asset = the 3D model asset and anything it depends on
  - e.g. other embedded data streams with associated filename

Challenge: identify all 3D assets (stream objects) in a PDF

- References to TSV refer to definitions in the Arlington PDF Model: https://github.com/pdf-association/arlington-pdf-model

PDF 1.6 3D annotations – U3D or PRC only

- **3D Annotation** /3DD key can be a 3D stream, or a 3D Reference Dictionary
- **3D Reference Dictionaries** support synchronized interactive models by using a common runtime across 3D models
  - /3DD key is then the shared 3D stream
- Check /Subtype of 3D stream to determine if U3D or PRC format
  - Has usual keys for streams (/Filter, /Length, etc.)
- Use PDF object number to determine if this is a shared 3D stream
  - i.e. referenced from multiple 3D Reference Dictionaries
- Decompress/decrypt as necessary to extract raw model



PDF 2.0 RichMedia annotations

- Only in PDF 2.0 (ISO 32000-2)
  - Originally an Adobe proprietary extension with proprietary and open formats
  - PDF 2.0 prohibits proprietary formats
- RichMedia annotations are more complex, but more flexible, adaptable & configurable
  - RichMedia annotations are *preferred* for U3D and PRC
  - STEP and glTF *only* use RichMedia
- https://github.com/pdf-association/PDF-RichMedia-Annotations

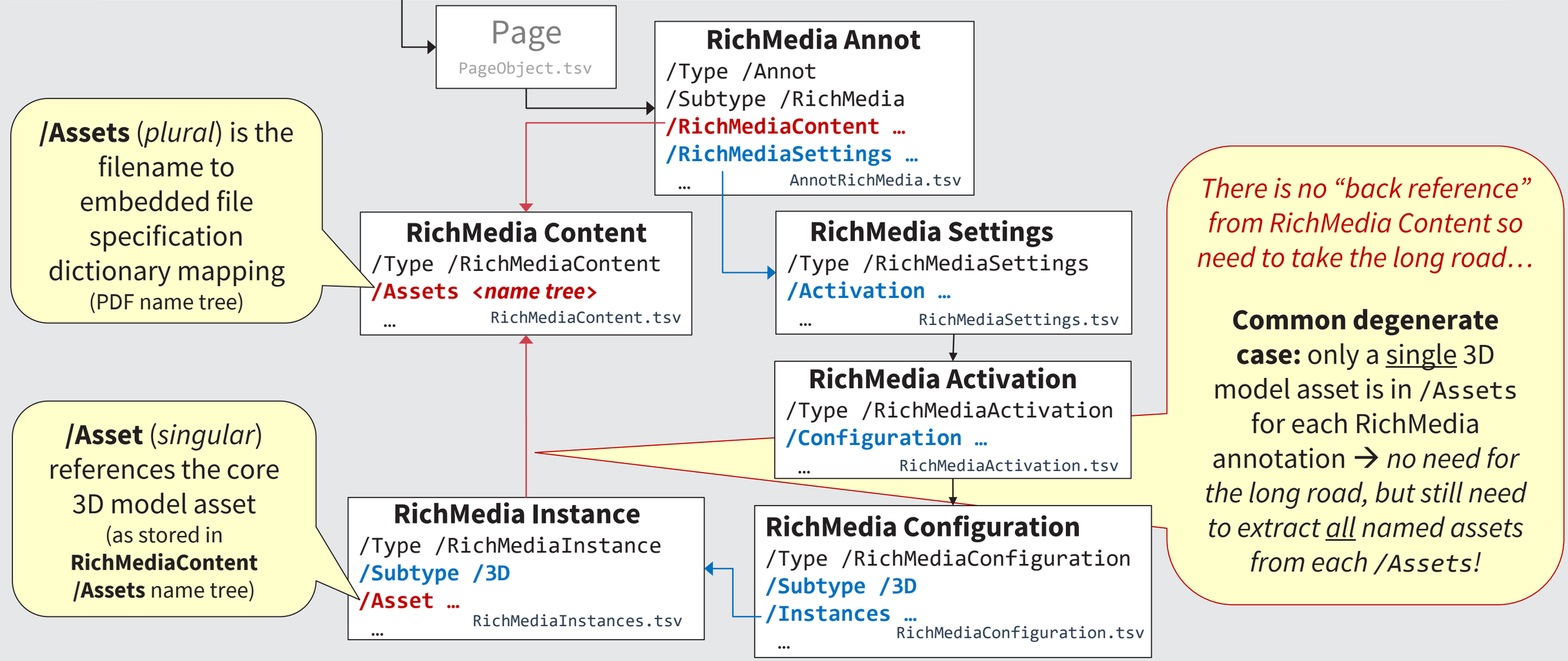
PDF 2.0 RichMedia annotation model

- Each RichMedia annotation can have its own set of named assets
  - As needed by the PDF RichMedia annotation itself (i.e. *understood by PDF*)
  - As needed by the media runtime engine + core asset (i.e. *not understood by PDF*)
- Each RichMedia annotation uses a PDF ***name tree*** to map filenames to embedded data streams
  - i.e. maps Unicode strings to Embedded File Specification Dictionaries
  - PDF name trees can be complex data structures

PDF 2.0 RichMedia challenges

- **Challenges:**
  1. For each annotation: need to know which asset is the core 3D model asset
     - e.g. a STEP model can reference other STEP models, images, etc.
  2. For each annotation: need to extract all named assets as they *might* be needed
     - Only known by the media runtime engine when things are referenced at runtime
  3. Embedded File streams may not have a MIME media type
     - Determining what an embedded file stream is requires knowing referred context from the annotation, and/or examining the raw data stream itself

PDF 2.0 RichMedia objects



RichMedia: Embedded Files

```
21 0 obj    % Trivially simple(!) RichMedia Assets name tree
<<  /Names
    [ (script.js) 22 0 R
      (3d.prc)    23 0 R
    ]
>>
endobj

22 0 obj    % File specification dictionary for JavaScript
<<  /Type /FileSpec
    /F  (script.js)
    /UF (script.js)
    /EF << /F 24 0 R >>
>>
endobj

23 0 obj    % File specification dictionary for the 3D model
<<  /Type /FileSpec
    /F  (3d.prc)
    /UF (3d.prc)
    /EF << /F 25 0 R >>
>>
endobj
```

Name tree filename string and File Specification Dictionary /F and /UF all have to be the same. And can be Unicode!

```
24 0 obj % embedded file stream for ECMAScript
<<  /Type /EmbeddedFile
    /Subtype /text#2Fjavascript
    /Length ...
    /Filter ...
>>
stream
% Compressed/encrypted raw data for script.js...
endstream
endobj

25 0 obj    % Embedded file stream for 3D PRC data
<<  /Type /EmbeddedFile
    /Length ...
    /Filter ...
>>
stream
% Compressed/encrypted raw data for 3D.prc ...
endstream
endobj
```

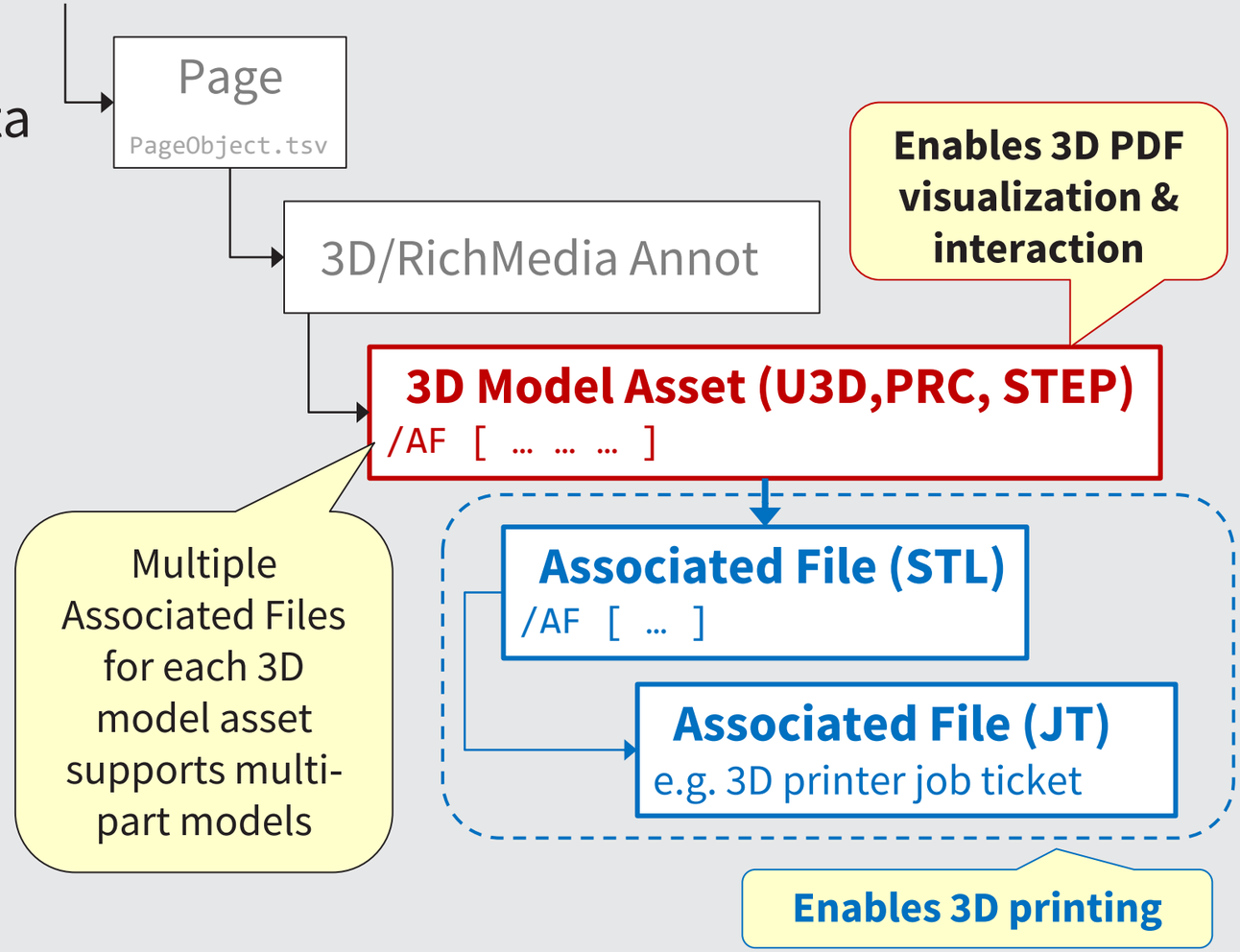
Optional /Subtype happens to be meaningful – *but what if it was `text/plain`*?

No /Subtype with a MIME media type so need to rely on referring context from annotation, and/or explicitly checking extracted data stream.

Adapted from https://github.com/pdf-association/PDF-RichMedia-Annotations/blob/main/RichMedia%20Annotation%20-%20example.md

Future thoughts

- Standardize location of 3D printable data streams relative to each interactive 3D model asset inside a container 3D PDF document
  - Interactive 3D PDF model = existing 3D model asset (U3D, PRC, STEP)
  - 3D printable data streams = 3D printer job tickets, sliced model formats (STL), …
  - Support multi-part models
    - e.g. a single 3D interactive model comprises multiple 3D printed parts, each with a specific job ticket
  - Use PDF 2.0 Associated File feature with `AFRelationship` semantic linkage



Summary

3D format	3D format specification	Since PDF version	PDF feature(s)
U3D	ECMA-363 (3rd edition)	PDF 1.6 PDF 2.0	3D annotations RichMedia annotations
PRC	ISO 14739-1	PDF 2.0	RichMedia annotations
STEP	ISO 10303-242	PDF 2.0	RichMedia annotations
glTF	*T.B.D.*	*T.B.D.*	RichMedia annotations

- No need to check PDF version → processing is backwards compatible
- A single PDF file can have many 3D models of any type on any page → need to iterate
- 3D models can be reused across pages → identify by object number of core 3D asset
- STEP models can depend on other named files in the PDF → extract all named assets
- **Always** refer to vendor neutral ISO 32000-2:2020 (PDF 2.0) → *every prior PDF reference has many errors & issues*!

Thank you!

peter.wyatt@pdfa.org