
vera
PDF

VeraPDF:
Building the definitive PDF/A validator

The European Union's PREFORMA project

Boris Doubrov, Dual Lab and
Duff Johnson, PDF Association

on behalf of the veraPDF consortium
slightly changed by H-J Hübner, PDF Association

Why PDF/A?

- Traditional archival methods are becoming dated; more documents are becoming final without print
- As a document format, PDF is far superior to TIFF; and accommodates raster images of pages equally well
- As a preservation format, PDF/A restricts the use of PDF to the specific set of issues affecting reproducibility
- PDF/A is the platform on which to build solutions for creating, retaining, sharing, authenticating and using reliable, high-quality, high-value electronic documents

Why does validating PDF/A matter?

- Validation is crucial to the archiving mission
- Numerous regulated industries (nuclear, aerospace, engineering, financial services, legal) demand a means of assessing compliance with regulated practices
- In the real world: if “PDF/A software” disagrees with other “PDF/A software”, what’s a user to do?
- The PDF/A flag is only metadata, validation is laborious
- PDF/A is complex; more on that later…

Why veraPDF?

- PDF/A, and the standards on which PDF/A is based, are themselves complex and therefore imperfect
- When there's not much revenue in validation per se, widespread adoption demands open source solutions
- Industry acceptance requires that a **commercial** entity cannot define conformance with PDF/A
- A definitive validator guarantees the means of **interoperability** (it's PDF's holy grail, after all!)

veraPDF's target audience

- ISO committee and PDF Association members
- PDF software developers (from Adobe Systems to zpdf)
- ECM / DMS / CCM industry product managers
- Digital preservation practitioners
- Government and industry regulators
- Procurement organizations
- IT researchers
- End users checking PDF/A conformance at verapdf.org

Business case

- For organizations creating and using documents
  - Reduced operational risks and legal liabilities
  - More capable, more interoperable software, enabling…
    - Easier and more feature-rich document sharing and management
    - Process automation
    - Archival of rich engineering content
    - Enhanced accessibility
- For PDF technology developers
  - Reduced software development costs
  - Reduced support costs

The veraPDF consortium

- Digital preservationist community
  - **Lead:** Open Preservation Foundation
  - Digital Preservation Coalition
  - KEEP Solutions
- PDF technology industry
  - **Lead:** PDF Association and members of its PDF Validation TWG
  - Dual Lab (veraPDF lead developer)

Open Preservation Foundation

PDF association

dual lab

KEEP
University of Minho SPIN-OFF

Why PDF/A validation is so special

- Relies on two different versions of PDF (1.4, 1.7), with specifications 1000+ pages long
- PDF/A comes in three versions, three levels and two technical corrigenda for PDF/A-1
- The PDF/A standard aims for long-term (100+ years!) preservation of electronic documents
- Needs more formal analysis of the requirements

A way to start: test corpora

- Isartor, Bavaria, BFO: 323 test atomic self-documented test files
- Initial test case analysis shows about **500** extra files needed for proper test coverage of all PDF/A versions and levels
- subject to adjustment based on real-world cases and practical considerations
- *to compare: the W3C corpus for XML 1.1 (specified on 38 pages) contains over 600 test files*

How test files are created

- Sample clause of PDF/A-1 specification:
  - ISO 19005-1:2005, 6.1.6 String objects: Hexadecimal strings shall contain an even number of non white-space characters, each in the range 0 to 9, A to F or a to f.
  - ISO 19005-1:2005, 3.17 **white-space character**: **NULL** (00h), **HORIZONTAL TABULATION** (09h), **LINE FEED** (0Ah), **FORM FEED** (0Ch), **CARRIAGE RETURN** (0Dh) or **SPACE** (20h) character
- Test cases identified:
  - Odd number of characters 0-9,A-F,a-f (**fail**)
  - Non white-space characters outside the range 0-9,A-F,a-f (**fail**)
  - Even number of 0-9,A-F,a-f; all possible white-space characters (**pass**)

How test files are created

- Test files created (QA engineer):
  - veraPDF test suite 6-1-6-t01-fail-a.pdf
  - veraPDF test suite 6-1-6-t01-fail-b.pdf
  - veraPDF test suite 6-1-6-t01-pass-a.pdf
- Github pull request is submitted (QA engineer)
- The file is merged to the master branch after the internal review (Architect):

https://github.com/veraPDF/veraPDF-corpus-PDFA-1b

Acceptance process

- In case the specification allows several interpretations, or when **a representative set of existing validators** generates mixed results:
  - Discuss with industry experts on the PDF Validation TWG mailing list
  - Review and resolve discussions during TWG meetings + ISO Committee
- The corpus is announced as a release candidate (Github)
- Formal acceptance by PDF Validation TWG
- The next milestone: **July 15, 2015**

  (release candidate of the enhanced PDF/A-1b corpus)

Examples of TWG discussions

- Private data: does it need to be validated for File Structure requirements?
- Which glyphs shall have widths synchronized with embedded fonts data?
- What is the length of a Name, which is a subject to implementation limits?
- Is it a violation of PDF/A-2,3 requirements, if a Page object does not contain a Resources dictionary, but inherits resources from its parent Pages object?

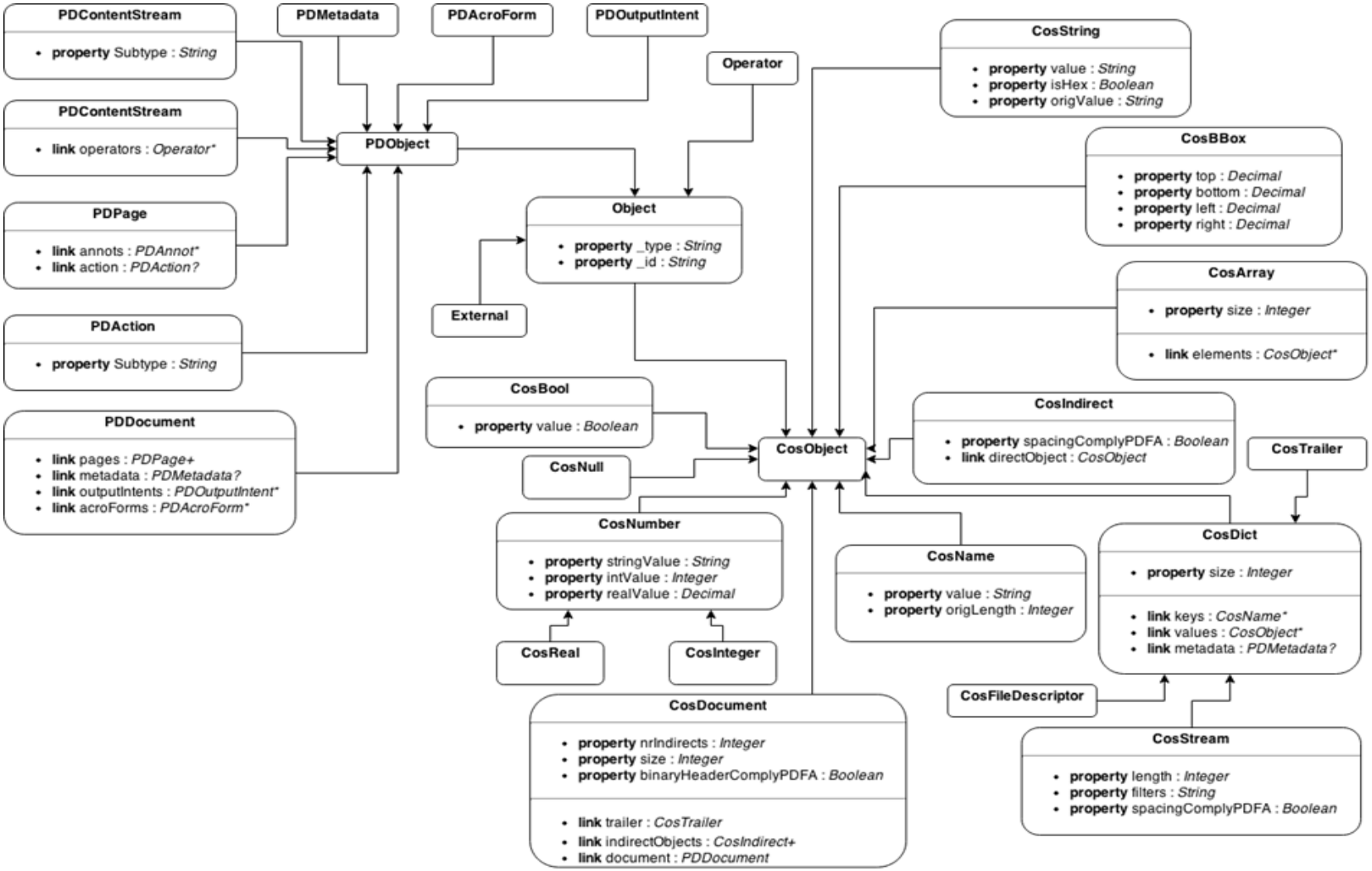
Validation extents

- PDF/A specification refers to PDF specifications, which rely on a number of external standards. How far PDF/A validation shall go to guarantee the reliable long-term preservation of PDF documents?
- Validity of embedded fonts, ICC profiles, font CMaps, XMP Metadata, Image compression, digital certificates is crucial!
- Collaboration with experts from relevant technologies is necessary for complete coverage

Validation model

- Hierarchy of object types
- Inheritable properties and named links to other object types
- Validation rules defined per object type (inheritable) by boolean expressions in JavaScript
- Each rule has metadata including the ID, description, normative references, error message
- Rules are combined into validation profiles and signed

veraPDF


PDContentStream
property Subtype : String
PDMetadata
PDAcroForm
PDOutputIntent
Operator
CosString
property value : String
property isHex : Boolean
property origValue : String
PDContentStream
link operators : Operator*
PDObject
CosBBox
property top : Decimal
property bottom : Decimal
property left : Decimal
property right : Decimal
PDPage
link annots : PDAnnot*
link action : PDAction?
Object
property _type : String
property _id : String
CosArray
property size : Integer
link elements : CosObject*
External
PDAction
property Subtype : String
CosBool
property value : Boolean
CosIndirect
property spacingComplyPDFA : Boolean
link directObject : CosObject
PDDocument
link pages : PDPage+
link metadata : PDMetadata?
link outputIntents : PDOutputIntent*
link acroForms : PDAcroForm*
CosObject
CosTrailer
CosNull
CosNumber
property stringValue : String
property intValue : Integer
property realValue : Decimal
CosName
property value : String
property origLength : Integer
CosDict
property size : Integer
link keys : CosName*
link values : CosObject*
link metadata : PDMetadata?
CosReal
CosInteger
CosFileDescriptor
CosDocument
property nrIndirects : Integer
property size : Integer
property binaryHeaderComplyPDFA : Boolean
link trailer : CosTrailer
link indirectObjects : CosIndirect+
link document : PDDocument
CosStream
property length : Integer
property filters : String
property spacingComplyPDFA : Boolean

veraPDF

PDF model in Xtext syntax

```
# a single root type
type Object {};

# parent object for all basic object types in PDF
type CosObject extends Object {};

# the object representing PDF Array type
type CosArray extends CosObject {
     property size: Integer;
     link values: CosObject*;
};

# parent object for all content stream operators
type Operator extends Object {};

# Operator q (save graphics state)
type qOperator extends Operator {
     property nestingLevel: Integer;
};
```

XML syntax for validation rules

```
<rule id="1a-6-1-12-r05" object="CosArray">
<description>Maximum capacity of array in elements less than 8192</description>
     <test>size &lt; 8192</test>
<error>
     <message>Capacity of array greater than 8191</message>
</error>
<reference>
     <specification>ISO 19005-1:2005</specification>
     <clause>6.1.12</clause>
</reference>
<reference>
     <specification>PDF Reference 1.4</specification>
     <clause>Table C.1</clause>
</reference>
</rule>
```

PDF Parser implementation

- PDF Parser implements the interfaces automatically generated from the PDF Model syntax
- Proof of concept is implemented using PDFBox, an open-source Java library
- Due to licensing requirements of PREFORMA all code of the PDF/A Conformance Checker shall be delivered under the dual license: GPLv3 + MPLv2
- As PDFBox is Apache-licensed, PREFORMA requested a greenfield implementation of a PDF parser

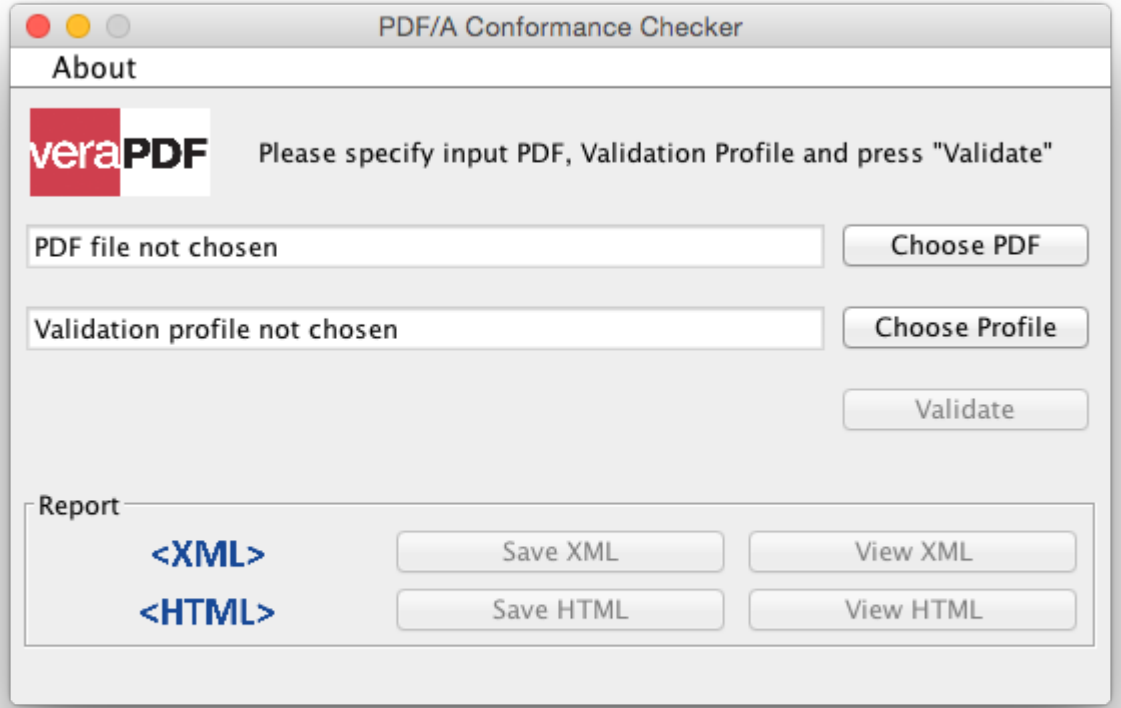
Development processes

- **Github** is the repository for both code and test corpora
  https://github.com/verapdf
- **Travis** manages the continuous builds
  https://travis-ci.org/veraPDF
- **Jenkins** manages automated tests and deployment
  http://jenkins.opf-labs.org/job/veraPDF-library/
- **Sonar** monitors code quality
  http://sonar.opf-labs.org/dashboard/index/8021

Minimum Viable Product (MVP) demo

- Clone Java repository
- Build the veraPDF library and the veraPDF GUI client
- Check the location of test files and validation profiles
- Run the veraPDF client selecting any PDF file and any validation profile
- Generate the machine-readable report (XML)
- Convert it to HTML and view in a system browser

veraPDF


PDF/A Conformance Checker
About
veraPDF
Please specify input PDF, Validation Profile and press "Validate"
PDF file not chosen
Choose PDF
Validation profile not chosen
Choose Profile
Validate
Report
<XML>
Save XML
View XML
<HTML>
Save HTML
View HTML

Report generation

- Machine-readable reports generated in XML format
- Error messages defined in validation profiles
- Conversion to HTML by XSLT technology
- Conversion to PDF by XSLT to XSL-FO and further to PDF by Apache FOP processor
- Localization of the final report messages via the standard TMX syntax of translations

Other components

- Metadata Fixer
  - Fix any incorrect claims of specific PDF/A validity
  - Add PDF/A identification to an otherwise conforming PDF/A document
- Policy Checker
  - Generate a PDF Features Report with page-count, lists of embedded fonts and images, and other information
  - Verify the Report against Policy requirements in Schematron syntax
- Plug-in infrastructure
  - Third-party tools can perform additional validation of embedded data
  - Interoperability with other PREFORMA suppliers

Timeline

2015-04-14	Start of Phase 2 (implementation)	2015
2015-06-08	First MVP released	
2015-07-15	Prototype PDF/A-1b support	
2015-10-30	Release candidate PDF/A-1b support, prototype PDF/A-1a, 2, 3	
2016-02-28	Release candidate PDF/A-1a, 2, 3 support	2016
2016-06-30	TWG approval for PDF/A-1b corpora	
2016-12-20	TWG approval for PDF/A-1a, 2, 3 corpora	
2016-12-31	End of Phase 2	
2017-01	Beginning of Phase 3 (acceptance testing)	2017
2017-06	End of Phase 3 (final delivery)	

Get involved!

- Join the PDF Association
- Join the PDF Validation Technical Working Group
- Download the candidate test suite files from GitHub
- Share your test files
- Test the code posted on GitHub and report issues
- Develop plug-ins for validating embedded data
- Think about how you could use definitive open source PDF/A validation