November 13, 2025

# RELEASE NOTES

### This is a special iText PDF

As per usual, we have decided to create a PDF file that showcases a lot of what is possible with iText by Apryse.

This is no ordinary PDF, as it is PDF/UA-2 and PDF/A-4F compliant, created with the help of pdfHTML.

You can read more about it below.

## Table of Contents

## Release iText Core 9.4.0

> As always, we've made the Core release notes into a showcase PDF document. Not only does it conform to the latest PDF/UA-2 standard for accessible PDF documents, it's also digitally signed and has source code and resources required to recreate the document yourself embedded. Have fun!

**Release date: Nov 13th 2025**

For our fourth and final iText Core release of 2025 we're introducing advanced page resizing functionality for more effective merging and manipulation of PDF documents with varying page sizes. We've also been working on extending the new EU eIDAS Trusted Lists feature to support validation against specific signature profiles. In addition, we've enhanced our flex layout support within the layout module for use with the pdfHTML add-on.

## Advanced Page Resizing

This release introduces advanced page resizing functionality, allowing users to merge and manipulate PDF documents with varying page sizes more effectively. This was based on a customer request who was using similar functionality in iText 5. However, the way iText 5 did this was relatively unsophisticated, and ran into problems with tagged documents and different content types.

This new implementation addresses scenarios such as scaling content between different page formats (A0, A1, US Letter, etc.), maintaining aspect ratios, and handling annotations, patterns, gradients, and form fields during resizing. The result is a more flexible and reliable toolkit for document assembly and transformation, ensuring that content integrity and layout are preserved across diverse PDF workflows.

## Signature Profiles Support

This focuses on enhancing digital signature capabilities in iText Core by introducing support for signature profiles. This feature enables an easier way to ensure that digital signing and validation

processes meet the regulatory and compliance requirements for specific regions. As a demonstration, we've updated the LotlSimpleSignatureValidation sample (Java/.NET) on GitHub to show off the new QualifiedValidator class. This determines if a given certificate/signature meets the regulatory standards to be "qualified", which is vital for high-trust electronic transactions.

We've also made a change to LOTL validation reports to eliminate multiple successful validation messages during the process, along with some general improvements to LOTL caching and validation.

## Extending Flex Layout Support

This release significantly improves the flex layout support in the core Layout module. Since this primarily affects pdfHTML, you'll find more details in the release notes for pdfHTML 6.3.0.

## Pull Requests

A big thank you goes to kpldvnpne for suggesting a fix for the PdfOutline.removeOutline function where the parent outline was also removed after removing a child. As noted in the PR, our implementation goes a little further by adding tests, and maintaining the Count value when an outline is removed.

We'd also like to thank IBaltaga for their suggested NotSupportedException if AreaBreak is inside a flex container fix for .NET. After investigating in more detail we decided to implement the fix in a different way, however, an exception will no longer be thrown for AreaBreaks inside flex containers.

## Bug Fixes and Miscellaneous

PDF image and color depth handling has been enhanced when extracting images There are also accessibility and structure improvements in generated/tagged documents, and error messaging and recovery for malformed or edge-case PDFs is improved.

For .NET, we've implemented a wrapper for Asn1OutputStream which became available in version 1.0.2 of BC FIPS .NET. This improves cryptographic compatibility for FIPS workflows.

## Other Stuff

> If you use iText for digital signing, you may be interested in the Digital Signatures Hub which contains a ton of useful resources and examples.

Don't forget that in addition to the resources on our Knowledge Base, on our GitHub you can find a ton of useful up-to-date samples in the following repos:

**Java**

https://github.com/itext/itext-publications-examples-java
https://github.com/itext/itext-publications-book-java
https://github.com/itext/itext-publications-signing-examples-java
https://github.com/itext/itext-publications-signatures-java
https://github.com/itext/itext-publications-highlevel-java
https://github.com/itext/itext-publications-jumpstart-java

**.NET**

https://github.com/itext/itext-publications-samples-dotnet

> If you want to create ZUGFeRD/Factur-X-e-invoices with iText Core, we have both Java and .NET code samples available targeting the current ZUGFeRD/Factur-X specification. They demonstrate how to embed the XML invoice data and add the metadata required for conformance. Read this article to learn more about ZUGFeRD/Factur-X, and using these code samples to create EN 16931-compatible e-invoices.

Bear in mind that our `master` branch contains samples for the current stable release, while the default `develop` branch is for the bleeding edge commits towards the next release.

Also, don't forget to check out the release-related examples below, as well as the updated Core add-ons in the iText Suite we've released this time:

## iText Suite 9.4 Releases

Release pdfCalligraph 5.0.4
Release pdfHTML 6.3.0
Release pdfOCR 4.1.1
Release pdfOptimizer 4.1.1
Release pdfSweep 5.0.4
Release pdfXFA 5.0.4

## Downloads

| | GitHub | Maven | NuGet | Artifactory |
|---|---|---|---|---|
| iText Core – 9.4.0 (Java) | link | link | N/A | link |
| iText Core – 9.4.0 (.NET) | link | N/A | link | link |

## Changelog

### New features

- DEVSIX-9287: Be able to resize pages
- DEVSIX-9419: Signature profiles

### Improvements

- DEVSIX-9288: Support more of flexlayout

### Bug Fixes

- DEVSIX-2454: The color depth 2 is not supported
- DEVSIX-2941: Extracted Indexed monochrome PDF image image gets color inverted
- DEVSIX-2943: The color depth 4 is not supported
- DEVSIX-3634: When several Unicode characters map to a single glyph in a font, iText creates /ToUnicode mapping mapping only to one Unicode which distorts the original text on text extraction
- DEVSIX-3937: GlyphLine substitutions override font program's glyph unicode mapping
- DEVSIX-9128: FontProgram.getSubset is not thread safe
- DEVSIX-9202: PR: Fix bug: dont remove parent just because children is empty
- DEVSIX-9269: Spike: PR: Fix for NotSupportedException if AreaBreak is inside a flex container
- DEVSIX-9457: Fix NPE due to null color space while identifying image type

## Deprecation of Older iText Versions

We're taking this opportunity to announce End of Life dates for deprecated iText versions. EOL for iText 7.1 was April 2025, and iText 7.2 was October 2025. As for iText 8.0, this will reach EOL in October 2026.

> EOL for these versions means they will transition to maintenance mode, and only receive security patches. However, just as with iText 5 we will continue to provide support for our customers. Even though iText 5 has been in maintenance mode since 2016, we regularly release new versions to address CVEs and other security-related bugfixes. See CVEs or the iText 5-specific CVEs page for more information.

## Contributors

We'd like to shout out the following contributors for this release:

- Core team
  - Eugene Bochilo
  - Dmitry Chubrick
  - Angelina Pavlovets
  - Alexandr Pliushchou
  - Vitali Prudnikovich
  - Dmitry Radchuk
  - Andrei Stryhelski
  - Nanou Persoons
  - Glenn Volckaert
  - Alexandr Fedorov
  - Guust Ysebie
  - Yulian Gaponenko
  - Raf Hens
- Product / Marketing
  - André Lemos
  - Ian Morris
- Infrastructure / Devops
  - Marco Andries
  - Yauheni Borbut
- Research
  - Michaël Demey
  - Vlad Lipskiy
- Input from other teams
  - Rainer Plöckl
  - Alison Anderson
- Input from outside
  - Michael Klink
  - Matthias Valvekens

## eBooks

Blockchain for PDF Documents
iText: Jump-Start Tutorial for .NET
iText: Jump-Start Tutorial for Java
iText: Converting HTML to PDF with pdfHTML
iText: Building Blocks
Best iText 7 Questions on StackOverflow

## Installation Instructions

Installing iText for Java
Installing iText for .NET
Installing iText Community for Java developers
Installing iText Community for .NET developers

## Examples (latest ones)

Refactored PDF Conformance Architecture
PAdES Signing High Level API
iText Core: Signature Appearance Improvements
iText 8.0.2 Delivers PDF/A-4 Support
High-level Annotation Flattening

## FAQ (latest ones)

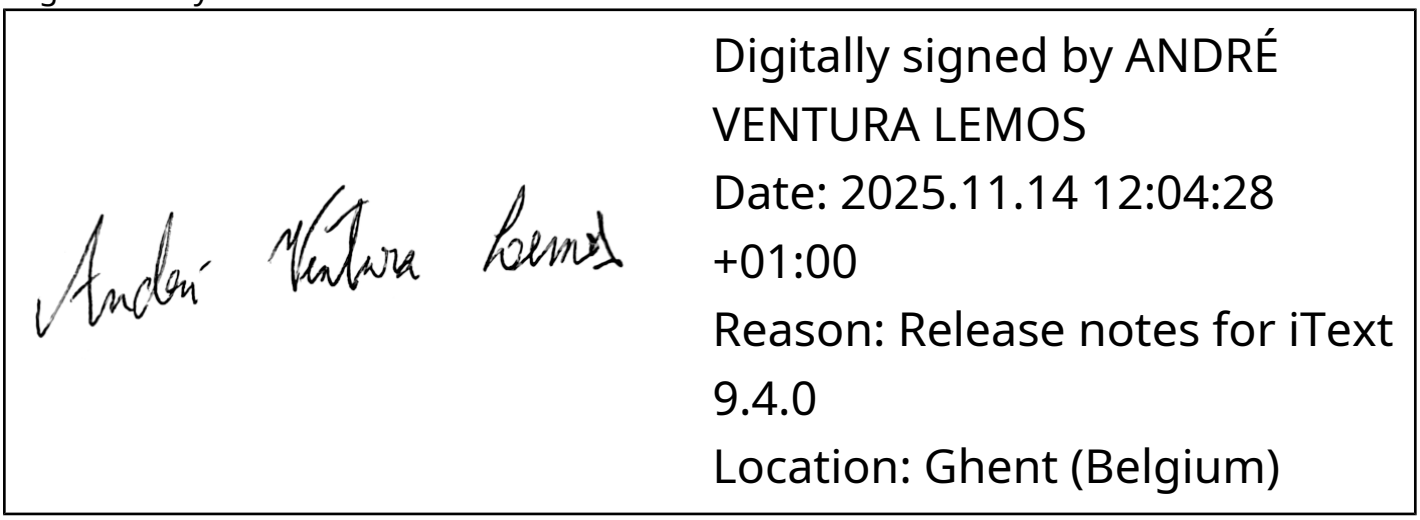
How does iText handle PDF encryption?
Resolving Microsoft.DotNet.PlatformAbstractions.dll Exception in Azure Functions
How can I find code examples for specific iText versions or releases?
How to deploy an iText License for Azure PowerShell?
License file was corrupted

## Features of this PDF

- Used pdfHTML to generate the PDF content (the HTML content is attached)
- It is both PDF/A-4f and PDF/UA-2 compliant (making it also a WTPDF)
- There is a MAC protected (AES 256 Encryted, SHA 256 Protected. Password is 'itext') version of this PDF attached (had to be separate, as PDF/A-4 does not allow it)
- Digitally signed using a Portuguese Identity Card (scroll below to check the code on how to validate it!)
- The main logo is generated using iText's custom SVG rendering engine
- Dynamically generated table of contents and bookmarks
- *Creatively* used Layers to toggle between Java and .NET code below
- Automatic Pagination by using our Events engine
- Fonts were *subsetted* for a smaller file size

To run the project (using .NET 8), just check the instructions on the attached file README.md.

Signed off by:




Generated by: Content: Ian Morris Source code: Guust Ysebie

Addendum: Verify the signature using iText (Java)

```java
package com.itextpdf.samples.sandbox.signatures.validation;

import com.itextpdf.kernel.pdf.PdfDocument;
import com.itextpdf.kernel.pdf.PdfReader;
import com.itextpdf.signatures.validation.SignatureValidator;
import com.itextpdf.signatures.validation.ValidatorChainBuilder;
import com.itextpdf.signatures.validation.lotl.LotlCountryCodeConstants;
import com.itextpdf.signatures.validation.lotl.LotlFetchingProperties;
import com.itextpdf.signatures.validation.lotl.LotlService;
import com.itextpdf.signatures.validation.lotl.QualifiedValidator;
import com.itextpdf.signatures.validation.lotl.RemoveOnFailingCountryData;
import com.itextpdf.signatures.validation.report.ValidationReport;

import java.io.IOException;
import java.util.Map;

public class  LotlSimpleSignatureValidation {

        public static final String SRC =  "./src/main/resources/pdfs"
            +  "/super_official_document_signed.pdf";

        public static void main(String[] args)  throws IOException {
            ValidatorChainBuilder builder =  new ValidatorChainBuilder();
            builder.trustEuropeanLotl(true);
            LotlFetchingProperties fetchingProperties =  new LotlFetchingProperties(
                                    new RemoveOnFailingCountryData());

             // Add other country codes if needed
            fetchingProperties.setCountryNames(LotlCountryCodeConstants.PORTUGAL);
            LotlService.initializeGlobalCache(fetchingProperties);

             // To perform Qualification validation, provide QualifiedValidator instance.
             // You can use this same instance to obtain the results after validation.
            QualifiedValidator qualifiedValidator =  new QualifiedValidator();
            builder.withQualifiedValidator(qualifiedValidator);

             try (PdfDocument document =  new PdfDocument(new PdfReader(SRC))) {
                SignatureValidator validator = builder.buildSignatureValidator(document);
                ValidationReport report = validator.validateSignatures();
                 // Separately, now you can obtain Qualification results
                Map<String, QualifiedValidator.QualificationValidationData> result =
                        qualifiedValidator.obtainAllSignaturesValidationResults();
            }  catch (IOException e) {
                 throw new RuntimeException(e);
            }
    }
}
```