July 8, 2026

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 9.7.0

> 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: 2026-07-08

Our third iText Core release of 2026 brings some significant new capabilities. We've added support for the WebP image format, introduced dynamic page margins and footnotes to the layout module, and strengthened decompression bomb protection. We've also improved append mode reliability and updated several security-related dependencies.

WebP Image Format Support

iText now natively supports the WebP image format, developed by Google to provide superior lossless and lossy compression for web images. This new capability allows you to integrate high-quality, highly compressed modern web images directly into your PDF workflows, significantly reducing file sizes without sacrificing visual fidelity.

Our implementation uses the TwelveMonkeys ImageIO library for WebP decoding on Java, and SkiaSharp on .NET and is delivered as a separate, optional module: webp-image-support. When the module is present on the classpath (Java) or referenced in your project (.NET), iText Core automatically detects and processes WebP images without requiring any additional configuration.

Capabilities

- Automatic WebP image type detection via ImageTypeDetector
- Lossy and lossless WebP image decoding
- ICC color profile handling for accurate color reproduction
- Transparency (alpha channel) support
- Base64-encoded WebP images in HTML/SVG sources

Notes/Limitations

- For animated WebP images only the first frame will be rendered.
- The rotation property is not (yet) supported, but browsers currently ignore this anyway.
- The module is not currently supported on Android, due to the javax.awt and imageio decoding libraries not being available.

Dynamic Margins & Footnotes

We have completely overhauled the way iText handles complex document layouts. This not only allows developers to adjust page margins on the fly, but also includes footnote support with automatic numbering, customizable footnote containers, and intelligent layout logic that ensures footnote anchors and text remain perfectly positioned.

To achieve this, we have introduced a powerful new framework for managing complex document layouts. This overhaul allows developers to adjust page margins on the fly using the new SectionBreak element.

Additionally, we've launched comprehensive footnote support, enabling automatic numbering, customizable footnote containers, and intelligent layout logic that ensures footnote anchors and text remain perfectly positioned.

Signatures and Validation

We've resolved critical issues related to Append Mode, ensuring that incremental updates to PDF documents are more reliable. This includes better tracking of modified objects and fixing stream inconsistencies, allowing for a more robust document history and preservation of digital signatures.

Page rotation handling for signature layers is also improved, and we fixed some edge-case crashes during form flattening.

To ensure uninterrupted support for digital signature validation, we have released a new version of our EU Trusted Lists resources. This update incorporates the recently updated European Journal sources, maintaining compliance with the latest eIDAS standards.

Security and Stability

iText's protection against "decompression bombs" inside PDF streams has been enhanced, particularly for single /FlateDecode streams and image-based decompression bombs. The change applies a 100× growth factor limit to all streams by default. Note that the existing API to override protection remains in place, in cases where edge cases may trigger false positives. However, this should be extremely unlikely for real-world uses cases.

Pull Requests

For this release we've incorporated two pull requests. The first is from iText alumni Dmitry Radchuk to add support for the :is() and :where() selectors for the styled-xml-parser module utilized by the pdfHTML add-on. Therefore, full details can be found in the pdfHTML 6.3.3 release notes.

The other pull request comes from Netliomax25 who proposed a way to increase security in iText's PDF/UA accessibility checks, specifically for rich text inside annotations. The fix swaps in a hardened XML parser to block potential XML External Entity injection attacks.

As always, many thanks to our contributors!

Bug Fixes and Miscellaneous

A customer-reported NullPointerException during form field flattening was fixed. The issue occurred when a signature field contained an invalid appearance entry (a non-stream object declared as a Form XObject). iText now logs a warning and skips the invalid appearance instead of crashing.

A PDF/A-2 conformance validation bug was fixed where PdfA2Checker.checkSeparationCS incorrectly threw an exception when two Separation color spaces with *different* names were used. The checker was not properly comparing the color space names, treating all Separation spaces as having the same name. This fix also applies to PDF/A-3 and PDF/A-4 conformance checking.

We added a high-level mechanism to prevent infinite loops in the layout module. In rare edge cases involving complex combinations of keep-together, forced placement, and nested containers, layout could enter an infinite loop. The engine now detects such situations and forces placement after a

configurable number of iterations.

The Jackson dependency has been updated to version 2.22.0 to stay current with security patches.

An informational warning is now logged when text requires the pdfCalligraph add-on for correct rendering but pdfCalligraph is not available. This helps developers identify rendering issues with complex scripts early in development.

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 9.7 Releases

- Release pdfCalligraph 5.1.0
- Release pdfHTML 6.3.3
- Release pdfOCR 5.0.1
- Release pdfOptimizer 4.1.4
- Release pdfSweep 5.0.7
- Release pdfXFA 5.0.7

Downloads

	GitHub	Maven	NuGet	Artifactory
iText Core – 9.7.0 (Java)	link	link	N/A	link
iText Core – 9.7.0 (.NET)	link	N/A	link	link

Changelog

New features

- DEVSIX-9369 – Support the WebP image format in iText
- DEVSIX-9375 – Allow adjusting margins for pages after they've been initialized (dynamic page margins and footnotes)

Improvements

- DEVSIX-10016 – Improve handling of decompression bombs inside PDF streams with single /FlateDecode filter
- DEVSIX-9276 – Handle page rotation for signature appearance layers set by the user
- DEVSIX-7483 – Update BC FIPS .NET: Use BasicOcspRespGenerator instead of own implementation
- DEVSIX-9919 – Release new version of eu-trusted-lists-resources
- DEVSIX-9923 – Update Macedonian trust list URL
- DEVSIX-9979 – Improve LOTL transition period warning
- DEVSIX-9922 – Update BouncyCastle on Java to 1.84+ (CVE-2026-5588)
- DEVSIX-9895 – Bump Jackson to 2.21.2
- DEVSIX-10083 – Bump Jackson dependency to 2.22.0
- DEVSIX-9931 – iText add-ons no longer always require BC adapter on .NET
- DEVSIX-9255 – Let the user know that pdfCalligraph is required to process certain scripts
- DEVSIX-9988 – Add support for .NET 8 runner for iText Core
- DEVSIX-9960 – High-level mechanism to prevent infinite loops in layout

Bug fixes

- DEVSIX-10007 – PDF/A validation error when two Separation colors are added
- DEVSIX-9936 – NullPointerException during form field flattening

- DEVSIX-9777 – Missing /Subtype /Form in stream when using append mode
- DEVSIX-9881 – Fix WTPDF checks when both conformance levels are requested
- DEVSIX-9916 – When checking for WTPDF conformance in XMP metadata only first element checked
- DEVSIX-9977 – Embedded file attachments: deferred stream consumption & orphaned PdfStream on duplicate names

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
  - Andrei Stryhelski
  - Nanou Persoons
  - Glenn Volckaert
  - Maksim Bezrukov
  - Evgeniy Zhavoronok
  - 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

Webinar

eBooks

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

Installation Instructions

- Installing iText for Java
- Installing iText for .NET

Examples (latest ones)

- Resizing Pages using the PageResizer Class
- Refactored PDF Conformance Architecture
- Overview: Using CSS in SVG with iText SDK
- iText 9 Signing Improvements
- Getting Layers from a PDF Page

FAQ (latest ones)

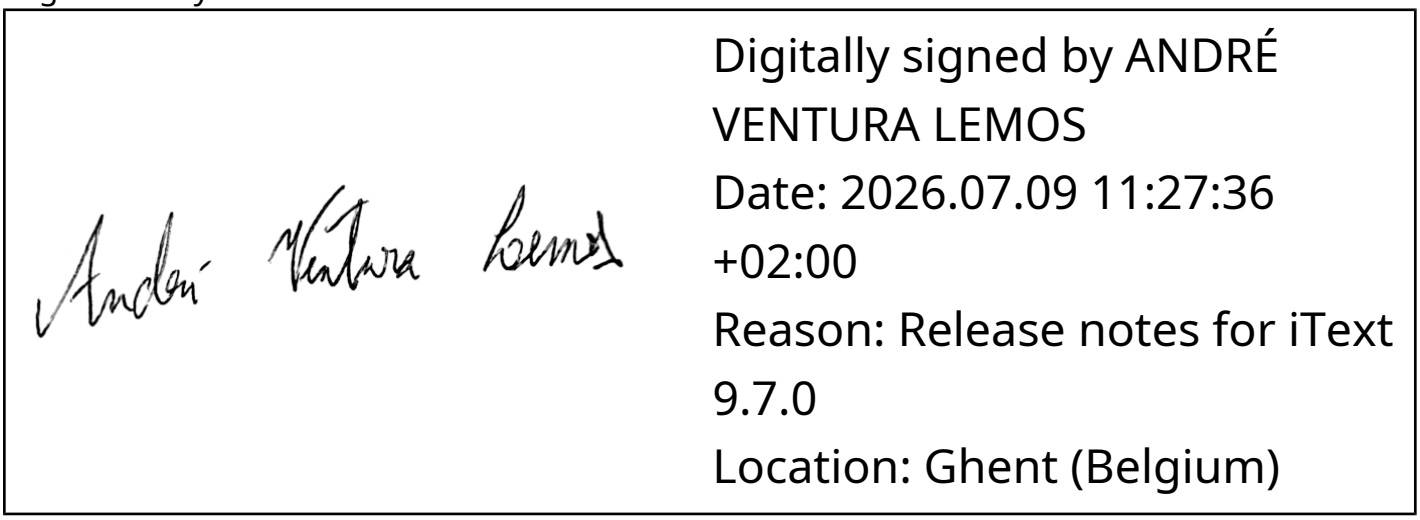
- What features are supported or unsupported in pdfHTML?
- Resolving Microsoft.DotNet.PlatformAbstractions.dll Exception in Azure Functions
- How can I find code examples for specific iText versions or releases?
- How to deploy an iText License for Azure PowerShell?
- How does iText handle PDF encryption?

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: GitHub Release Notes To PDF Repo

iText by Apryse

WebP images in PDF documents

To enable WebP[1] image support in your PDF, you must first include the official iText WebP module in your project. In .NET, simply add the following NuGet[2] package reference to your project file (or use the Package Manager Console):

```
<PackageReference Include="itext.webp-image-support" Version="9.7.0" />
```

In Java, the same result is achieved by adding the following Maven dependency to your pom.xml – the artifact is hosted on Maven Central[3] and is automatically resolved by your build tool:

```
<dependency>
    <<groupId>com.itextpdf</groupId>
    <<artifactId>webp-image-support</artifactId>
    <<version>9.7.0</version>
</dependency>
```

Once this dependency is in place, you can seamlessly embed WebP images into your PDF documents using the standard ImageDataFactory, and iText will automatically handle the decoding and rendering.

[1] WebP is a modern image format that provides superior lossless and lossy compression for images on the web. Using WebP, webmasters and web developers can create smaller, richer images that make the web faster.
[2] NuGet is the package manager for .NET.
[3] The Maven Central Repository is the default remote repository used by Maven to download project dependencies for Java.

Addendum: Add footnotes using iText (Java)

```
package com.itextpdf.samples.sandbox.layout;

import com.itextpdf.kernel.colors.ColorConstants;
import com.itextpdf.kernel.colors.DeviceRgb;
import com.itextpdf.kernel.pdf.PdfDocument;
import com.itextpdf.kernel.pdf.PdfWriter;
import com.itextpdf.layout.Document;
import com.itextpdf.layout.Style;
import com.itextpdf.layout.borders.SolidBorder;
import com.itextpdf.layout.element.Paragraph;
import com.itextpdf.layout.properties.margins.*;

import java.io.IOException;

public class Footnotes {
    public static final String DEST = "./target/sandbox/layout/footnotes.pdf";

    public static void main(String args[]) throws IOException {
        try (PdfDocument pdfDoc = new PdfDocument(new PdfWriter(DEST));
             Document doc = new Document(pdfDoc)) {
            // Configure automatic numbering (i, ii, iii, ...) resetting on each page and styles.
            doc.setFootnotesProperties(new FootnotesProperties()
                    .setFootnoteNumberingType(FootnoteNumberingType.ROMAN_LOWER)
                    .setFootnoteNumberingConfig(FootnoteNumberingConfig.PER_PAGE)
                    .setFootnotesContainerStyle(new Style()
                            .setBorderTop(new SolidBorder(ColorConstants.LIGHT_GRAY, 1))
                            .setBackgroundColor(new DeviceRgb(250, 250, 250))
                            .setPaddingTop(8)));

            Footnote fn1 = new Footnote("Coffee was first cultivated in Ethiopia.");
            Footnote fn2 = new Footnote("Decaffeination removes at least 97% of caffeine.");
            Footnote fn3 = new Footnote("Arabica and Robusta are the most important species.");

            doc.add(new Paragraph()
                    .add("Coffee").add(new FootnoteAnchor(fn1))
                    .add(" is widely consumed. Some prefer decaffeinated").add(new FootnoteAnchor(fn2))
                    .add(" for the taste without the stimulant effect.")
                    .setMarginBottom(15));
            doc.add(new Paragraph()
                    .add("There are dozens of coffee species").add(new FootnoteAnchor(fn3))
                    .add(", but only a few are grown at commercial scale."));
        }
    }
}
```