

October 1, 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 9.8.0	3
Release date: 2026-10-01	3
Vertical Text Layout & CJK Writing Modes	
Layout, Rotations & Transformations	
Advanced SVG & Graphics Capabilities	
Markdown to PDF GitHub Action	4
Footnotes & Margin Enhancements	
Security and Stability	5
Pull Requests	5
Bug Fixes and Miscellaneous	6
Other Stuff	7
iText Suite 9.8 Releases	8
Downloads	8
Changelog	9
New features	9
Improvements	9
Bug fixes	10
Deprecation of Older iText Versions	10
Contributors	11
eBooks	12
Installation Instructions	12
Examples (latest ones)	12
FAQ (latest ones)	12
Features of this PDF	13

Release iText Core 9.8.0

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: 2026-10-01

We are pleased to announce the release of iText Core 9.8.0. This release introduces support for vertical text layout and CJK writing modes, radial gradients and SVG mask support, and advanced SVG stroke properties. In addition, we've introduced a higher-level transformation API with more user-friendly affine transform capabilities in the Layout module, enhanced footnote navigation, critical security updates across key cryptographic and utility dependencies, and substantial performance and standards-conformance improvements.

Vertical Text Layout & CJK Writing Modes

你 好 世 界	こ ん に ち は 世 界	안 녕 하 세 요 , 여 러 분 !	iText Core 9.8.0 introduces initial support for vertical writing modes in the Layout module, primarily aimed at CJK (Chinese, Japanese, and Korean) layouts. Whilst the implementation is not yet complete, the API has been finalized and supports line breaking, baseline adjustments, right-to-left vertical progression, accessibility tagging, and improved sizing behavior in tables, flex containers, inline blocks, and text renderers. The current implementation also covers word and character spacing, text alignment, overflow handling, text decorations, and text-rise behavior, with related pdfHTML alignment for CSS vertical writing modes.
------------------	---------------------------------	--	--

Layout, Rotations & Transformations

The Layout module now includes more user-friendly affine transformation capabilities, with simplified API options for `Property.TRANSFORM` ([Java](#)/[.NET](#)). The release also resolves layout issues affecting rotated table cell sizing, content offsets, rotated page coordinates, and relative image positioning with borders.

Advanced SVG & Graphics Capabilities

This release expands SVG and graphics fidelity with radial gradient support across `SVG`, `Kernel`, and `Layout`, SVG `<mask>` support via PDF soft masks, advanced stroke properties such as `stroke-linecap`, `stroke-linejoin`, and `stroke-miterlimit`, URL-based stroke references with fallbacks, and fixes for nested clip paths with translations.

Markdown to PDF GitHub Action

Related to these SVG improvements, we've been working on providing Markdown-to-PDF via GitHub Actions. We're proud to announce **itext-markdown2pdf** is now available on the GitHub Marketplace!

This action is available free-of-charge and is intended to help open-source developers convert Markdown in public repositories into accessible, standards-compliant PDF/UA-2 (and WTPDF) documents within their CI/CD workflows.

Accessibility is important to us, and we've worked hard to simplify creating documents accessible to everyone. With PDF/UA-2 as the latest and greatest standard for accessible PDFs, we offer this as a gift to the open-source community to help popularize it.

- **Marketplace Listing:** <https://github.com/marketplace/actions/itext-markdown2pdf>
- **Source Repository:** <https://github.com/itext/markdown2pdf-gh-action>

We'd love you to try it out for yourselves, and feedback on possible improvements is welcomed.

Footnotes & Margin Enhancements

Footnotes are now easier to navigate in digital documents thanks to bi-directional clickable links between anchors and footnote bodies. This release also improves footnote numbering with footer and page-break handling, font resolution for footnote anchors, and margin handling during document relayout.

Security and Stability

iText Core 9.8.0 includes important security and stability updates across cryptographic, JSON, HTTP, and logging dependencies, including IV generation fixes, Bouncy Castle FIPS, Bouncy Castle standard libraries, Jackson Databind, Apache HttpClient, and Logback. It also improves PDF/A and PDF/UA validation, text extraction performance, circular reference protection, digital signature revision validation, custom font link handling, font resource licensing clarity, and runtime compatibility for .NET 10 and GraalVM JDK 24/25 environments.

Pull Requests

We'd like to thank [netliomax25-code](#) for [their fix to improve](#) CSS `~` attribute selector matching in [pdfHTML](#) and [styled-xml-parser](#) by properly escaping selector values before pattern matching.

Bug Fixes and Miscellaneous

Beyond the headline features, iText Core 9.8.0 includes a broad set of fixes and maintenance updates focused on performance, standards conformance, document stability, packaging quality, and developer experience.

- **Performance and resource handling:** Restored Standard 14 font caching in text extraction to address a performance regression, added circular form XObject protection in canvas processing and cleanup workflows, and optimized conditional logging to avoid unnecessary string formatting when log levels are disabled.
- **PDF/A, PDF/UA, and validation improvements:** Improved validation for PDF/A documents with multi-stream page contents and refined glyph validation for substituted glyphs without Unicode mappings.
- **Digital signing and document integrity:** Corrected revision counting in digital signature validation to avoid over-counting document processing events and improve validation reliability.
- **Layout and rendering fixes:** Fixed issues involving custom font links, relative image positioning with borders, rotated table cell sizing, footnote numbering with footer and page-break handling, and margin handling during relayout.
- **Packaging, runtime compatibility, and documentation:** Improved .NET 10 and GraalVM JDK 24/25 compatibility, centralized .NET build configuration, fixed licensekey and iText Core tests for .NET 10, and expanded public API documentation for the **io** module.
- **Compliance and dependency hygiene:** Replaced legacy bundled font resources with clearly licensed equivalents, updated related notice information, and improved GraalVM WebP test stability.

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 additions in the [iText Suite](#) we've released this time:

iText Suite 9.8 Releases

- [Release pdfCalligraph 5.1.1](#)
- [Release pdfHTML 6.3.4](#)
- [Release pdfOCR 5.0.2](#)
- [Release pdfOptimizer 4.1.5](#)
- [Release pdfSweep 5.0.8](#)
- [Release pdfXFA 5.0.8](#)

Downloads

	GitHub	Maven	NuGet	Artifactory
iText Core – 9.8.0 (Java)	link	link	N/A	link
iText Core – 9.8.0 (.NET)	link	N/A	link	link

Changelog

New features

- DEVSIX-9388, DEVSIX-10118, DEVSIX-10136, DEVSIX-10137, DEVSIX-10155, DEVSIX-10161, DEVSIX-10162, DEVSIX-10163, DEVSIX-10166, DEVSIX-10167, DEVSIX-10168, DEVSIX-10175, DEVSIX-10183, DEVSIX-10190 – Initial vertical text layout & CJK writing modes
- DEVSIX-9377, DEVSIX-10102, DEVSIX-10150 – Higher-level affine transformations in the Layout module
- DEVSIX-4396, DEVSIX-10037 – Radial gradient support across SVG, Kernel, and Layout
- DEVSIX-2378 – Converting SVG `<mask>` elements into PDF soft masks
- DEVSIX-10025, DEVSIX-10031 – Bi-directional clickable links between footnote anchors and bodies, with custom styling support for footnote anchors

Improvements

- DEVSIX-2623, DEVSIX-10156, DEVSIX-10180 – Vertical text formatting, alignment, overflow, text-rise, and CJK font metrics
- DEVSIX-2258, DEVSIX-4857, DEVSIX-10171 – SVG stroke properties, URL stroke references, and nested clip paths
- DEVSIX-5334, DEVSIX-8934, DEVSIX-5029 – Rotated layout handling for table cells, offsets, and page coordinates
- DEVSIX-10034, DEVSIX-10101 – Margin handling during document relayout
- DEVSIX-10100 – Font resolution for footnote anchors
- DEVSIX-10192, DEVSIX-10142, DEVSIX-10174, DEVSIX-10165, DEVSIX-10172, DEVSIX-10170, DEVSIX-10203, DEVSIX-10169, DEVSIX-9733, DEVSIX-6172 – Security updates for IV generation, Bouncy Castle, Jackson Databind, HttpClient, and Logback
- DEVSIX-10113, DEVSIX-10092, DEVSIX-10026, DEVSIX-10179 – .NET 10 / GraalVM compatibility, and `io` API documentation
- DEVSIX-9682, DEVSIX-9683 – Font compliance updates and notice file cleanup
- DEVSIX-5525 – Conditional logging optimization to avoid unnecessary string formatting when log levels are disabled

Bug fixes

- DEVSIX-10112 – Restoring Standard 14 font caching in text extraction
- DEVSIX-3608 – Stack overflow protection for circular form XObjects
- DEVSIX-10089 – Multi-stream PDF/A validation syntax boundary fix
- DEVSIX-10160 – Substituted glyph validation via direct glyph-code lookup
- DEVSIX-10177 – Signature revision count accounting
- DEVSIX-10130 – Custom **FontProvider** handling on **Link** elements
- DEVSIX-10019 – CSS `~='` selector value escaping
- DEVSIX-10094 – Footnote numbering with footer and page-break handling
- DEVSIX-1022 – Relative image positioning with borders

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](#)

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)

- [How do I Configure Logging for Volume License Troubleshooting?](#)
- [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?](#)

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 Encrypted, 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 *subsetting* 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:

	Digitally signed by ANDRÉ VENTURA LEMOS Date: 2026.10.12 10:45:15 +02:00 Reason: Release notes for iText 9.8.0 Location: Ghent (Belgium)
---	---

Generated by: Content: [Ian Morris](#) Source code: [GitHub Release Notes To PDF Repo](#)

```

package com.itextpdf.samples.sandbox.layout;

import com.itextpdf.kernel.colors.DeviceRgb;
import com.itextpdf.kernel.font.PdfFontFactory;
import com.itextpdf.kernel.pdf.PdfDocument;
import com.itextpdf.layout.element.Div;
import com.itextpdf.layout.element.VerticalParagraph;
import com.itextpdf.layout.renderer.FlexContainerRenderer;

/*
 * Example showing how to render text vertically using VerticalParagraph.
 * Demonstrates both VERTICAL_LR and VERTICAL_RL, mixing English and CJK characters.
 */
public class VerticalText {
    public static final String DEST = "./target/sandbox/layout/verticalText.pdf";
    public static final String FONT = "./src/main/resources/font/NotoSansCJKjp-Regular.otf";

    public static void main(String args[]) throws java.io.IOException {
        try (PdfDocument pdfDoc = new PdfDocument(new com.itextpdf.kernel.pdf.PdfWriter(DEST);
            com.itextpdf.layout.Document doc = new com.itextpdf.layout.Document(pdfDoc)) {
            doc.setFont(PdfFontFactory.createFont(FONT, com.itextpdf.io.font.PdfEncodings.IDENTITY_H));

            // Left-to-right lines writing progression for vertical text.
            VerticalParagraph ltr = new VerticalParagraph(false);
            ltr.add("VERTICAL_LR\n").add("左から右へ\n").add("Columns: left to right.\n")
                .add("日本語・中文・한국어\n");
            ltr.setFontSize(14).setBackgroundColor(new DeviceRgb(235, 245, 255))
                .setFontColor(new DeviceRgb(40, 60, 120)).setHeight(400).setMarginRight(30);

            // Right-to-left lines writing progression — standard for CJK vertical typography.
            VerticalParagraph rtl = new VerticalParagraph(true);
            rtl.add("VERTICAL_RL\n").add("右から左へ\n").add("Columns: right to left.\n")
                .add("日本語・中文・한국어\n");
            rtl.setFontSize(14).setBackgroundColor(new DeviceRgb(235, 250, 238))
                .setFontColor(new DeviceRgb(20, 100, 60)).setHeight(400);

            Div columns = new Div().add(ltr).add(rtl);
            columns.setNextRenderer(new FlexContainerRenderer(columns));
            doc.add(columns);
        }
    }
}

```