April 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 9.6.0

> 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: 01 Apr 2026

Our second iText Core release of 2026 will please those with a focus on creating accessible and reusable content in PDF. We've introduced the ability for iText to perform automatic color contrast checks during creation of PDF/UA documents, which will aid when ensuring generated documents meet many countries' accessibility regulations.

We've also added high-level functionality to make creation of Well Tagged PDF documents easier, by letting iText's comprehensive conformance checking architecture handle all the tricky stuff behind the scenes. You can also target the reuse and accessibility conformance levels defined in the WTPDF specification, depending on your requirements. The accessibility level directly aligns with PDF/UA-2, while reuse is optimized for content extraction, reflow, and machine-driven transformation.

In addition, we've extended trusted list validation to include non-EU countries, along with general improvements to digital signature validation and certificate retrieval.

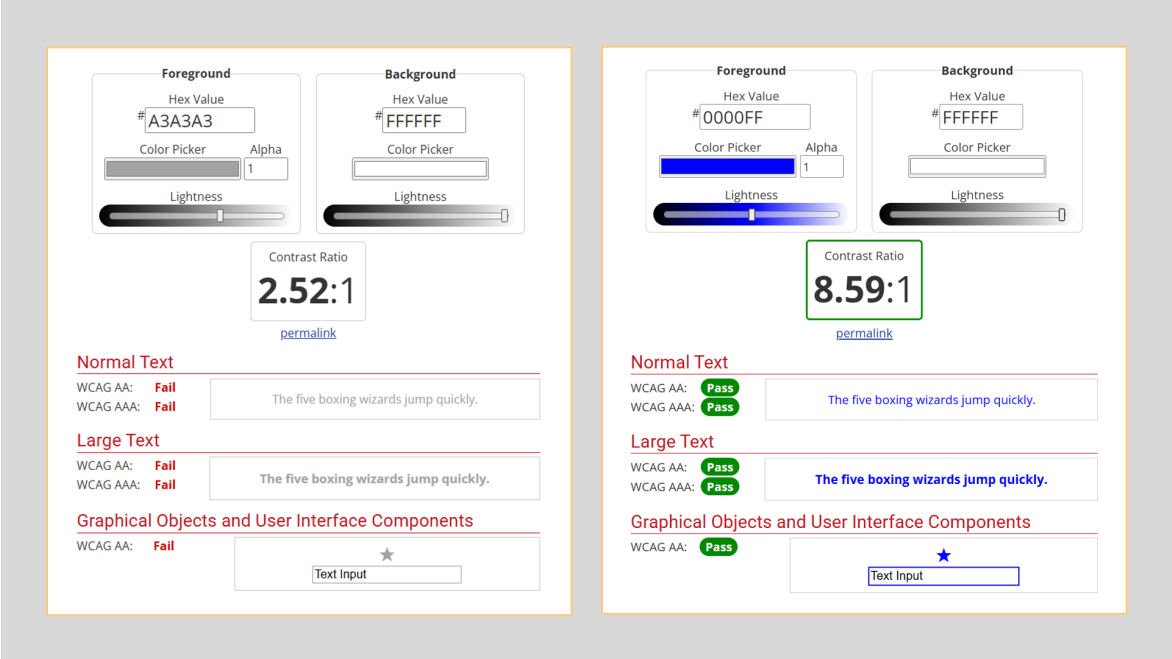
PDF/UA Color Contrast Checks

For documents to comply with the PDF/UA specifications, you must ensure that foreground and background colors meet the necessary WCAG standards for accessibility. If there is not enough contrast then not only will documents fail in accessibility checkers, text and graphical content will be difficult to read.

Color contrast checking is something that still requires human intervention in most cases, however, iText Core can now warn you if such issues are detected during document creation. It should be noted that this is not a comprehensive check and cannot substitute for an actual

“human in the loop”. However, it will help to identify and avoid obvious issues at an early stage.

This functionality is similar to that provided by the [WebAIM Contrast Checker](), where you can select different values for the foreground and background colors.



A comparison of color contrast ratios

As you can see in the comparison above, the example on the left fails all WCAG accessibility checks. For WCAG 2.1 level AA which is commonly specified in accessibility regulations, normal text must have a contrast ratio of at least 4.5:1, and 3:1 for large text, graphics, and user interface components (such as form input borders).

We have also improved iText’s layer handling to better accommodate these tests. Previously, reopening a document and accessing layers via PdfPage.getLayers() always returned layers in the “on” state regardless of their persisted visibility. Now, layer properties are correctly persisted after saving and reopening.

Easier WTPDF Creation

The PDF Association's Well-Tagged PDF (WTPDF) standard specifies requirements for creating PDF 2.0 documents geared towards reuse and accessibility. While both use cases have a large overlap in requirements, some are critical for reuse while others are only required for accessibility. Therefore the standard has conformance levels that clearly identify the requirements for each.

The reuse level is intended for PDF content to be more easily repurposed, extracted, or reflowed, including use cases such as PDF to HTML conversion and enabling responsive layouts on mobile devices. The accessibility level uses the same structural tagging model, but focuses more on ensuring people using assistive technologies can navigate and understand the document.

WTPDF document creation with iText Core was introduced along with the support for PDF/A-4 and PDF/UA-2, which are both based on the PDF 2.0 specification. However, we've since added a comprehensive validation architecture with specialized document classes and validation checkers to enforce compliance when targeting specific standards.

To bring WTPDF into line there is now a dedicated `WellTaggedPdfDocument` subclass of `PdfDocument`, to join the existing `PdfADocument` and `PdfUADocument` classes. In addition, the `PdfConformance` class now includes the accessibility and reuse conformance levels.

All these changes mean creating WTPDF documents with iText is just as easy as PDF/A and PDF/UA, with no need to worry about remembering which requirements apply to the conformance level you're targeting. You can now simply use `WellTaggedPdfDocument` to create a WTPDF document just like you would use `PdfUADocument` to create PDF/UA, and let iText do the hard work for you.

Non-EU Trusted List Validation

In iText Core 9.3.0 we introduced the List of Trusted Lists (LOTL) system, which lets iText fetch, validate, and cache European lists for eIDAS compliance. To better accommodate other countries which also issue and maintain trusted lists of their own, we've extended the feature to include a broad non-EU processing implementation.

This was achieved by adding a convenient **"**single file trust list**"** alternative implementation path, useful for countries where you have a single XML trust list rather than an EU-style LOTL + pointers ecosystem. The existing LOTL services have been refactored to support this, and the existing Ukraine and Moldova trusted lists samples have been updated to use this new implementation.

You can refer to the Java and .NET sample repositories to find the LOTL validation examples updated for this release.

Improved Digital Signature Validation and Certificate Retrieval

For digital signature validation we've introduced a single, configurable mechanism to control all online data retrieval (such as CRL, OCSP, issuing certificates, and LOTL), making it easier to enforce network policies and increase reliability in restricted environments.

Certificate chain building and validation have been updated to better handle complex PKI setups. Chain construction now collects all reachable certificates, including multiple CA candidates and cross-signed certificates, without enforcing constraints during the building phase.

Validation then applies RFC 5280–compliant checks over the resulting paths, with a particular focus on correctly enforcing inherited name constraints across the entire certificate chain. This separation of chain discovery and validation improves robustness and correctness when working with real-world certificate hierarchies.

Pull Requests

For this release we'd like to thank schallb for their contribution to improve `LocationTextExtractionStrategy` (Java/.NET) by adding support for custom element and newline separators.

As the author describes, they needed a way to better distinguish between spaces and newlines that are part of the extracted text, and the separators inserted by iText when it decides a word boundary or new line between chunks exists. Thanks to this change, the separators inserted in the formatted extracted text can be overridden.

Bug Fixes and Miscellaneous

We also resolved a few customer and internally-reported issues. The MemoryLimitsAwareHandler (Java/.NET) which protects applications from using too much memory when reading or processing PDF content was modified to have different behavior when creating xref tables rather than updating them. The documentation has been updated to reflect these changes.

We fixed an issue where certain Type3 fonts are processed incorrectly, causing multiple glyphs that share the same Unicode mapping to render with incorrect or duplicated content streams. This resulted in missing or wrong glyph appearances when using or extracting the font.

A regression was reported where signing a PDF removed the /DA (default appearance) entry from signature fields, triggering change warnings in Acrobat and affecting re-signing. The fix ensures the DA entry remains untouched during signing to preserve appearance and avoid Acrobat warnings.

An issue was fixed when copying pages from certain PDF/A-converted documents whose outline/action dictionaries were altered (using /Dest instead of /A), leading to a PdfException error reporting an object belonged to a different PDF. Another PDF/A-related fix prevents null pointer exceptions during PDF/A compliance checks by ensuring the content-stream validation parser is given the correct resource dictionary.

Error handling was improved in a few other cases too. For example, iText now throws a clear, specific error when a PDF annotation is missing its required subType property, instead of failing with a generic null pointer exception. A generic null reference exception when merging certain PDFs was replaced by clearer error reporting when input PDFs have syntax issues, making it easier to identify where the problem lies.

For .NET, we resolved an issue where older versions of the Newtonsoft.Json transitive dependency could be pulled for .NET Framework 4.6.1, causing security tools to flag as vulnerable to CVE-2024-21907 due to outdated dependencies. This has been resolved by updating the Microsoft.Extensions.DependencyModel to 10.0.2. This did not apply to .NET Standard 2.0, where since iText Core 9.2.0 we switched to use System.Text.Json instead.

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 9.6 Releases

Release pdfCalligraph 5.0.6
Release pdfHTML 6.3.2
Release pdfOCR 5.0.0
Release pdfSweep 5.0.6
Release pdfXFA 5.0.6

Release Related Examples

pdfOCR: Custom session options for an ONNX model
pdfOCR: PaddleOCR model support

Downloads

	GitHub	Maven	NuGet	Artifactory
iText Core – 9.6.0 (Java)	link	link	N/A	link
iText Core – 9.6.0 (.NET)	link	N/A	link	link

Changelog

New features

- DEVSIX-9361 – Easily create WTPDF
- DEVSIX-9364 – Perform Color Contrast checks for PDF/UA
- DEVSIX-9626 – Create an implementation for non-EU trusted lists processing

Improvements

- DEVSIX-9675 – Online data retrieval customization during signature validation
- DEVSIX-9727 – Certificate chain building must take multiple CA candidates into account
- DEVSIX-9746 – CertificateChainValidator must check name constraint
- DEVSIX-9719 – PdfLayers are always on when reopening document and accessing them through pdfPage.getLayers()

Bug fixes

- DEVSIX-9678 – Unexpected behavior of MemoryLimitsAwareHandler
- DEVSIX-9796 – Type3 Font not processed correctly (missing glyphs) when there are multiple glyphs with same unicode mapping
- DEVSIX-9455 – DA entry is removed from signature field during signing
- DEVSIX-9817 – Copying converted file using iText causes PdfException: Object belongs to other PDF
- DEVSIX-4742 – Throw specific error for an annotation missing a subtype.
- DEVSIX-9231 – Generic NRE is thrown when converting from PDF to PDF/A
- DEVSIX-9752 – Fix Newtonsoft.Json vulnerabilities coming from Microsoft.Extensions.DependencyModel

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
  - 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

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)

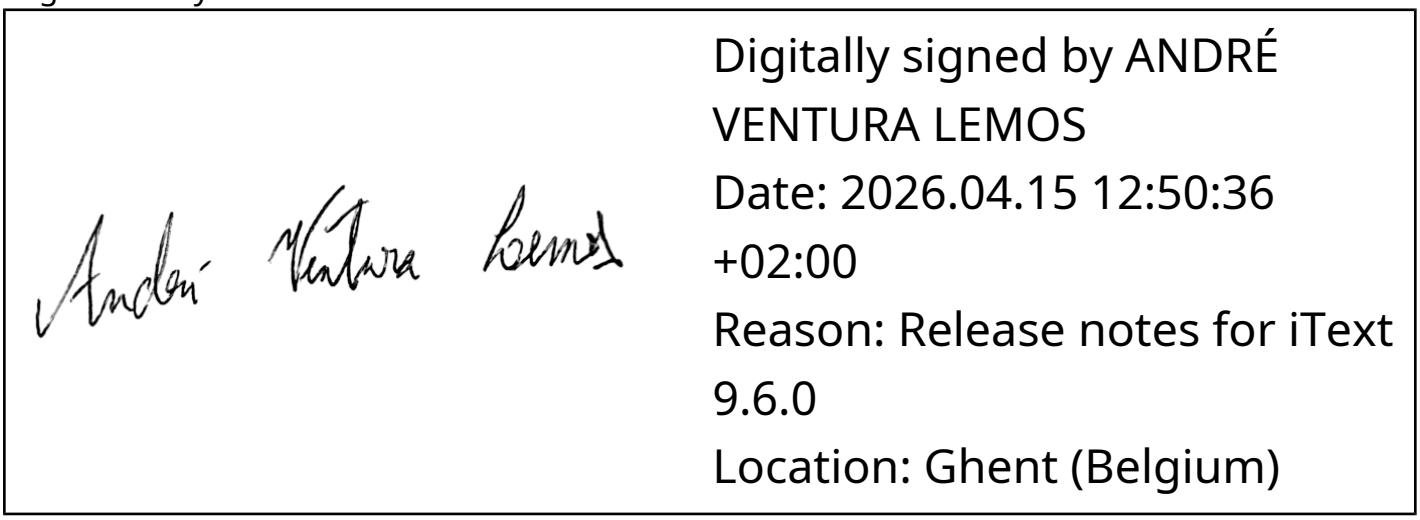
How to merge PDFs and add bookmarks?
How can I find code examples for specific iText versions or releases?
How can I find the border color of a field?
How to set the export value of a check box?
How to send a 'success' response back to Acrobat Reader from a java servlet?

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

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().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);
        }
    }
}
```