Accessible Tagged Report

EDITION	CAPABILITY	SIGNATURE	STANDARDS	GENERATED
Core	Tagged PDF / structure tree	—	ISO 32000-2 · ISO 14289-2 (PDF/UA-2)	2026-07-14

This report is generated with **Document::enableTaggedPdf()** — NextPDF's opt-in path to ISO 14289-2:2024 (PDF/UA-2). Every heading, paragraph, list, and table below is a real `StructElem` in the file's logical structure tree, wired with stable MCIDs, so a screen reader, a refreshable braille display, or any other assistive technology can traverse the document in its true reading order — independent of the visual column, table, or page layout on screen.

Why Structure Matters

A page is not a grid of pixels to an assistive technology — it is a tree. Sighted readers skim a page by its visual hierarchy: a bold headline, an indented list, a ruled table. A screen reader has no visual hierarchy to skim — it has only the structure tree the file declares. Without one, a PDF is, in practice, an opaque image of text.

What NextPDF Emits for This Page

- Headings (`H1`–`H3`) with no level skipped, so the outline a screen reader announces matches the one printed here.
- Paragraphs (`P`) carrying the reading order shown on the page.
- This list itself, as `L` with `LI` children — announced as a list of three items, not three stray sentences.
- Data tables with real header cells (`TH`), demonstrated by the spec panel above and the page-2 checklist.

Who This Helps

Tagged structure is what makes a PDF usable with a screen reader (JAWS, NVDA, VoiceOver), a refreshable braille display, or a text-to-speech tool — and it is what lets a sighted reader reflow the same content to a phone-sized column without losing the outline. It is also, in most jurisdictions that require it, the difference between a document that satisfies a legal accessibility obligation (the EU's EN 301 549, the US Section 508, WCAG 2.2) and one that only looks like it does.

What This File Declares

Beyond the visible content, `enableTaggedPdf()` writes the machine-readable conformance markers a validator inspects first — each row below is present in this very file:

Marker	Value in this file	What it tells a validator
`/MarkInfo`	`/Marked true`	The document claims a complete structure tree
`/StructTreeRoot`	present	Root of the logical structure an assistive technology walks
`Catalog /Lang`	`en`	Validated BCP 47 language for speech synthesis
`XMP pdfuaid:part / :rev`	`2 / 2024`	The file self-identifies as PDF/UA-2 (ISO 14289-2:2024)

PDF/UA-2 Conformance Checklist

The table below is itself a demonstration: its header row is tagged with real `TH` elements, so an assistive technology announces each column's label — Criterion, Requirement, Status — before reading a row's value, rather than leaving the reader to guess what a bare number or word refers to.

Criterion	Requirement	Status
Document title & language	Catalog /Lang and the Info dictionary Title resolve to a real, validated BCP 47 tag	Met
Logical structure tree	Every heading, paragraph, list, and table maps to a StructElem with a stable MCID	Met
Heading order	H1 precedes H2 precedes H3 on page 1, with no level skipped	Met
Data table headers	Column headers are real TH structure elements, not styled TD cells	Met
Reading order	The content stream is ordered exactly as an assistive technology will announce it	Met
Image alternative text	Every meaningful image carries a text alternative	N/A — no images in this report

Verifying These Claims Yourself

A vendor's own checklist is a claim, not proof. Point an independent validator at the file rather than taking this page's word for it:

- `verapdf --flavour ua2 accessible-report.pdf` — the open-source veraPDF reference implementation of the ISO 14289-2 Machine-Checkable Requirements.
- The PDF Association's PAC tool, which walks the same structure tree visually, node by node.
- Adobe Acrobat's built-in Accessibility Checker, or any screen reader's own reading-order inspector.

This sample is generated end to end by NextPDF Core — no post-processing accessibility remediation pass was applied. What an independent validator sees in the structure tree is exactly what `Document::writeHtml()` produced from the markup on this page.