# Tagging in LaTeX Documenten

**Abstract**
In juli 2025 bezocht ik in India een workshop van het LaTeX-team over Tagged PDF. In dit artikel beschrijf ik wat ik daar leerde: waarom tagging ertoe doet, hoe tagging tegenwoordig in LaTeX werkt (met `tagpdf` en `latex-lab`, inclusief sockets en templates), welke standaarden relevant zijn (PDF 2.0, PDF/UA-2, WTPDF) en welke praktische valkuilen je in de dagelijkse praktijk tegenkomt. Tot besluit ga ik in op nuanceringen en actuele uitdagingen in de *regulatory* context, met name rond `enumitem` (lijsten) en `titlesec` (koppen), die de weg naar goed getagde PDF's bemoeilijken.

**Keywords**
Tagged PDF; PDF/UA-2; WTPDF; LaTeX; tagpdf; latex-lab; sockets; templates; toegankelijkheid; regulatory; enumitem; titlesec

## Inleiding

LaTeX-documenten werden lange tijd geproduceerd zonder expliciete semantische structuur voor schermlezers en formele toegankelijkheidsvalidatie. Tijdens de workshop in India (juli 2025) werd duidelijk hoeveel vooruitgang er is geboekt met `tagpdf` en de integratie daarvan in `latex-lab`. In wat volgt zet ik die inzichten op een rij en plaats ik ze in de context van dagelijkse documentproductie.

## Waarom tagging?

Tagging maakt PDF-documenten toegankelijker (denk aan WCAG en Section 508), verbetert de machineleesbaarheid van koppen, links en lijsten en sluit beter aan op moderne PDF 2.0-checkers. Het is geen wondermiddel voor automatische informatie-extractie of NER, maar consistente LaTeX-markup voor bijvoorbeeld citaties, wiskunde en opsommingen draagt wel aantoonbaar bij aan betrouwbare extractie.

## Kernconcepten en commando's

De bouwstenen van tagging in LaTeX bestaan uit twee lagen. Aan de basis staat `tagpdf`, dat laag-niveau commando's biedt voor structuur en *Marked Content* (zoals `\tagstructbegin` en `\tagmcbegin`). Daarbovenop levert `latex-lab` modules die gebruikelijke LaTeX-constructies *tagging-aware* maken zodra tagging is geactiveerd via

```
\DocumentMetadata{tagging=on}
```

Een belangrijk mechanisme daarbij zijn de zogeheten "sockets": haken waarin, als tagging actief is, automatisch de juiste tagcode wordt uitgevoerd. Templates en bijbehorende instanties bouwen hierop voort om vormgeving op hoger niveau te structureren zonder de tagstructuur te breken. Tot slot helpt *expl3* met consistente naamgeving en argument-signaturen, waardoor interfaces rond tagging beter onderhoudbaar en modulair blijven.

## Standaarden en validatie

De beoogde doelstandaarden zijn PDF 2.0 en PDF/UA-2 (WTPDF). In de praktijk werkt dit het robuuststst met LuaLaTeX. Controle van de uitkomst blijft noodzakelijk: een combinatie van een UA-2-validator (zoals veraPDF), een syntaxiscontrole (bijvoorbeeld via de Arlington Model Checker) en een visuele inspectie van de tag tree (met Acrobat, PDF-XChange of NGPDF) geeft snel zicht op eventuele hiaten in de structuur of metadata.

## Praktische aandachtspunten

In productieomgevingen draait het om drie dingen. Allereerst de leesvolgorde: figuren en bijschriften horen als *siblings* onder de juiste ouder in de tagstructuur, en floats mogen niet buiten de actuele context eindigen. Daarnaast vragen alt-teksten en links om zorgvuldigheid: betekenisvolle afbeeldingen hebben een *Alt* en links moeten correct als gestructureerde bestemmingen worden opgenomen. Tot slot kost tagging extra tijd; vermijd pathologische constructies en verzamel representatieve voorbeelden zodra performance of stabiliteit onder druk staat, zodat gerichte optimalisatie mogelijk is.

## Regulatory: prioriteiten en blokkades

Tussen de sessies door in India stelde ik Frank Mittelbach de eenvoudige vraag hoe ik mijn *regulatory* oplossing goed getagd krijg. Zijn antwoord was even helder als direct: zolang `enumitem` en `titlesec` niet aanslui-

ten op de nieuwe interface (denk aan heading sockets en listhooks), blijven ze een bron van frictie richting *well tagged* PDF's. Dat gesprek zette de toon voor mijn verdere verkenning: precies dáár waar lijsten en koppen bepalend zijn voor navigatie en validatie, voelen we de randen van compatibiliteit het eerst. In de *regulatory* context — waar juridische en regulatoire documenten strakke eisen aan structuur en navigatie stellen — zijn twee veelgebruikte pakketten vandaag een wezenlijke belemmering voor goed getagde PDF's. `enumitem` geeft uitgebreide controle over lijstlay-out, maar ingrijpende herdefinities kunnen de listhooks omzeilen. Daardoor raakt de beoogde `L`/`LI`/`Lbl`/`LBody`-structuur uit balans, met labels die als gewone `P` of `Span` eindigen, met onnodig geneste `L`-niveaus of met onjuist afgesloten `LI`-nesten. De leesvolgorde en *structure destinations* rondom items worden dan minder voorspelbaar en validators signaleren inconsistenties in lijststructuur en navigatie.

Iets soortgelijks speelt bij `titlesec`. Het pakket maakt rijke kopopmaak mogelijk, maar kan langs de heading-sockets en automatische bestemmingen heen werken. Het gevolg is dat koppen soms niet als `H1`/`H2`/… in de tag tree landen, maar als `P` of `Span`, of dat bestemmingen ontbreken. Dat tast de outline en schermlezer-navigatie aan en leidt tot opmerkingen over de heading-hiërarchie bij controle.

Voor *regulatory* documenten, met hun nadruk op geneste opsommingen en betrouwbare kophiërarchieën, zijn deze effecten blokkerend: interpretatie, verwijzing en validatie berusten op precies die elementen die nu onder druk staan.

### `\DocumentMetadata` via de commandline, zonder bronaanpassing

Aan het einde van mijn verkenning van *regulatory* liep ik tegen een praktische spagaat aan. In de dagelijkse productie wil ik bestaande stromen niet onderbreken of bronnen wijzigen — zeker niet zolang niet alle gebruikte pakketten compatibel zijn met tagging — terwijl ik tegelijk wél Tagged PDF wil kunnen inschakelen om gedrag te observeren en te valideren. Precies voor die situatie heb ik een lichte commandlineaanpak ontwikkeld: *la-lua-opts*[1]. Daarmee kan ik de bedoelde metadata op compileertijd aanreiken zonder de bronbestanden te veranderen.

Concreet werkt dit als een commandlinetoggle: je geeft je intentie mee op de opdrachtregel en een initscript vertaalt dat bij het compileren naar precies één regel `\DocumentMetadata{...}` aan het begin van de invoer. Het kleine hulpproject *La Lua Opts* biedt daarvoor een wrapper rond `lualatex` en een bijbehorend initscript.

In de praktijk werkt het als volgt. De wrapper `la-lua-opts` start de engine met `--lua=...` zodat het initscript actief is. Dat script leest de commandline en beslist of er één `\DocumentMetadata{...}` moet worden toegevoegd vóór de `\documentclass`. Staat er al een `\DocumentMetadata` als allereerste regel, dan blijft de invoer intact. Begint het bestand daadwerkelijk met `\documentclass`, dan wordt er exact één regel met de gevraagde metadata geprepend. Begint het document op een andere manier (bijvoorbeeld met alleen commentaar of een preambulefragment), dan grijpt het script niet in en noteert het dat er niets is ingevoegd. Zo blijft het gedrag voorspelbaar en omkeerbaar.

De belangrijkste schakelaars spreken voor zich. Met `-t` wordt tagging aangezet (gelijk aan `tagging=on` in de metadata), met `-l nl` geef je de documenttaal door, en met `--pdfversion=2.0` dwing je een PDFversie af. Indien je conformiteit wilt toetsen, kun je één of meer standaarden doorgeven via `--pdfstandard=PDF/A-4f`; voor inspectie kan `--uncompress` handig zijn om objecten minder compact weg te schrijven. In woorden: je beschrijft de intentie op de opdrachtregel, en het initscript vertaalt dat eenmalig naar de juiste LaTeX-regel, zonder dat je je bron hoeft te bewerken.

Wie `latexmk` gebruikt, kan hetzelfde principe toepassen door de wrapper als engine te configureren. Daarmee kun je eenvoudig naast een gebruikelijke build ook een *tagged* build draaien, puur door het aanroepcommando te variëren. Let er wel op dat deze route is ontworpen voor *LuaLaTeX*; dat een reeds aanwezige `\DocumentMetadata` bovenaan het bestand leidend blijft; en dat voor PDF/A 4f de voorkeur uitgaat naar declaratie via `\DocumentMetadata` in plaats van oudere `pdfx`paden richting PDF/A 1b, zodat moderne validatoren de metadata eenduidig interpreteren.

Samengevat geeft de combinatie van wrapper en initscript een lichte manier om tagging en metadata aan of uit te zetten op compileertijd. Dat is bijzonder handig om ondersteuning te verkennen, regressies te reproduceren en bevindingen te onderbouwen, zonder je bronnen te vervuilen met tijdelijke wijzigingen.

## Engines

Voor tagging is *LuaLaTeX* de aangewezen route. *pdfLaTeX* blijft bruikbaar maar is gevoeliger voor randgevallen, terwijl *XeLaTeX* en dvips-achtige routes niet langer actief voor tagging worden ondersteund. Wie op UA-2/WTPDF mikt, kiest in de praktijk dus het best voor LuaLaTeX.

## Aan de slag

Wie nu met tagging wil werken, zorgt eerst voor een actuele TeX-distributie (bijvoorbeeld TeX Live 2025 of een up-to-date MiKTeX) en activeert tagging vroeg in het document via:

```
\DocumentMetadata{lang=nl,tagging=on}
```

Vervolgens is valideren de regel: combineer een UA-2-validator met een visuele inspectie van de tag tree om gericht eventuele hiaten op te sporen.

## Conclusie

Met de huidige toolchain (LuaLaTeX, tagpdf, latex-lab) is het goed mogelijk om robuuste, getagde PDF 2.0-documenten te produceren. Voor de regulatory-context blijven `enumitem` en `titlesec` vandaag de belangrijkste blokkades richting *well-tagged* PDF's. Tot er pakket-compatibiliteit of expliciete integraties met sockets beschikbaar zijn, is het raadzaam deze beperkingen in acht te nemen. Een commandline-toggle voor tagging maakt het eenvoudig om ondersteuning te verkennen en bevindingen te onderbouwen zonder bronnen aan te passen.

## Noot

1. La Lua Opts broncode:
`https://github.com/Xerdi/la-lua-opts`

Erik Nijenhuis
erik (at) xerdi dot com
`https://github.com/MacLotsen`