

Het regulatory pakket*

Erik Nijenhuis
erik@xerdi.com

2026-09-10

Dit bestand wordt onderhouden door Xerdi . Foutmeldingen kunnen worden gemeld op https://github.com/Xerdi/regulatory .

Samenvatting

Het regulatory pakket is geschreven voor juristen in brede zin. Het biedt de structuren waaruit zo'n document is opgebouwd — artikelen, leden, onderdelen en definities — zonder iets van $\LaTeX$ weg te nemen.

Verwijzen binnen het juridisch domein is lastiger dan het lijkt, en daarom verwijst dit pakket zoals $\LaTeX$ dat doet: door te labelen waarnaar verwezen wordt en dat label te noemen. De macrofamilies `\rref`, `\nref` en `\aref` verwoorden de verwijzing van daaruit, in het Nederlands of in het Engels.

Definities worden in één $\LaTeX$ -bestand bijgehouden dat ieder document bedient dat dezelfde begrippen aanhaalt, en ze worden aangehaald met de `\gls` familie van glossaries, waarop dit pakket voortbouwt in plaats van hem te vervangen. Naar artikelen, leden, onderdelen en definities van een *ander* document dat met dit pakket geschreven is wordt op dezelfde manier verwezen, zodat twee documenten elkaars bepalingen kunnen noemen en elkaars begrippen kunnen delen — algemene voorwaarden en een onderhoudsovereenkomst, bijvoorbeeld. Het document waarnaar verwezen wordt kan als bijlage in het PDF-bestand worden ingesloten, zodat de lezer het bij de hand heeft.

*Dit document hoort bij regulatory 1.0.0, uitgebracht op 2026-09-10.

Inhoudsopgave

1	Gebruik	4
2	Voordat u begint	7
2.1	Engines en vereisten	7
2.2	Definities	7
2.3	HTML	7
2.4	Bijlagen	8
2.5	Talen	8
2.6	PDF-conformiteit	8
2.7	Bekende beperkingen	9
2.8	Wat de tests vaststellen	9
2.9	Publiek, uitbreiding en intern	10
3	Overstappen naar 1.0	11
4	Structuur	13
5	Definities	15
6	Verwijzingen	17
6.1	Conjunctie	18
7	Interdocumentaire verwijzingen	19
7.1	Welke link een bijlage opent	21
8	Handtekeningen	23
9	Externe bronnen	24
10	Taalondersteuning	29
10.1	Engelse definities	31
10.2	Nederlandse definities	34
10.3	Duitse definities	38
10.4	Franse definities	40
11	Markdown	43
11.1	Labels en verwijzingen	43
11.2	Een eigen identifier	44
11.3	Definities in Markdown	44
12	HTML	46

13 Tests	48
13.1 De testdocumenten	48
13.2 Wat de suite naleest	53
13.3 PDF-conformiteit	55
13.4 De testbestanden	58
14 Implementatie	69
14.1 regulatory-struct	69
14.2 regulatory-defs	79
14.3 regulatory-ref	83
14.4 regulatory-attachments	95
14.5 regulatory-md	105
14.6 regulatory-sign	113
14.7 regulatory-sources	116
14.8 Markdown syntaxuitbreiding	137
14.9 regulatory.4ht	139
Register	149

1 Gebruik

Deze bundel schrijft PDF, dus de engine is `pdflatex` of `lualatex`. Er is ook een HTML-conversie, die op `make4ht` draait en een eigen configuratie nodig heeft; zie paragraaf 12.

```
\documentclass[dutch]{article}
\usepackage{regulatory}
\begin{document}
  \article{...}
\end{document}
```

Codelijst 1: `main.tex`

`regulatory` zelf is niet meer dan een omhulsel: het laadt de zes modules hieronder en geeft iedere optie door aan de eerste ervan. Elke module is ook afzonderlijk te laden, en elke module vraagt zelf om wat hij nodig heeft, dus de volgorde hieronder is de afhankelijkheidsketen en geen eis.

- `regulatory-struct` de structuren van paragraaf 4. Deze module vormt de basis van de bundel: hij bevat de opties van alle modules en leest `regulatory.cfg`.
- `regulatory-defs` de definities en definitielijsten van paragraaf 5.
- `regulatory-ref` de verwijzmacrofamilies van paragraaf 6 en de taalondersteuning van paragraaf 10.
- `regulatory-attachments` de interdocumentaire verwijzingen en PDF-bijlagen van paragraaf 7, die op zijn beurt `regulatory-ref` en `regulatory-defs` laadt.
- `regulatory-sign` de handtekeningvelden van paragraaf 8. Deze drukt niets af totdat een van zijn twee commando's gebruikt wordt.
- `regulatory-sources` de aanhalingen van regelgeving buiten het document van paragraaf 9.

Een zevende module, `regulatory-md`, komt daar bovenop zodra `markdown` geladen is, in welke volgorde de twee ook staan (zie paragraaf 11). De HTML-conversie is geen module maar een `tex4ht`-configuratie, `regulatory.4ht`, die `make4ht` zelf vindt.

Iedere optie is een sleutel van de familie `/regulatory`, die `regulatory-struct` voor de hele bundel bijhoudt. Het voorbeeld hierboven geeft er geen mee, wat inhoudt dat deze standaardwaarden gelden:

- `bib2gls true` definitielijsten worden met `bib2gls` gebouwd. Geef `bib2gls=false` mee om ze in plaats daarvan als $\TeX$ -code te laten schrijven.
- `md false` het laden van `markdown` wordt aan het document gelaten. Met `md` laadt het pakket het zelf, wat vóór `enumitem` moet gebeuren.

<code>alldefs</code>	<code>false</code>	een definitielijst bevat de definities die het document gebruikt. Met <code>alldefs</code> bevat hij iedere gedeclareerde definitie, wat Algemene Voorwaarden willen, waarin niet ieder begrip per se in de tekst voorkomt.
<code>legacy</code>	<code>false</code>	de conjunctie van paragraaf 6. Met <code>legacy</code> wordt de verouderde gebruikt, zie <code>\setjuncto</code> .
<code>nameinlink</code>	<code>true</code>	de hyperlink van een verwijzing wordt om het gehele label heen getrokken. Met <code>nameinlink=false</code> alleen om het nummer.
<code>defstyle</code>	<code>almtree</code>	de woordenlijststijl waar <code>\printdefs</code> op terugvalt als hij er geen krijgt; zie paragraaf 5.
<code>sourcestyle</code>	<code>full</code>	of een aanhaling van een externe bron voluit (<code>full</code>) of verkort (<code>short</code>) geschreven wordt; zie paragraaf 9. Hij luistert ook naar bronstijl, en naar voluit en verkort als waarden. Een waarde die geen van de vier is wordt geweigerd waar hij gegeven wordt in plaats van iedere opzoeking die hem leest leeg te maken.
<code>attachmentlink</code>	<code>goto</code>	hoe een bijlage uit paragraaf 7 vanaf de pagina geopend wordt: met een embedded go-to action (<code>goto</code>) of met een file attachment annotatie (<code>attachmentlink=annotation</code>). De twee verschillen in wat ze betekenen, in welke viewers eraan gehoor geven en in wat de PDF-standaarden ervan vragen; paragraaf 7.1 zet ze naast elkaar. Hoe dan ook staat het bestand ook in het bijlagenpaneel van de viewer, dat geen van beide nodig heeft.

Gekleurde kaders van hyperlinks worden verborgen met `\hypersetup{<hidelinks>}`, aangezien dit pakket `hyperref` zelf laadt. Een bestand `regulatory.cfg` naast het document wordt gelezen vóór de opties die aan het pakket meegegeven zijn verwerkt worden, zodat het andere standaardwaarden kan zetten voor een hele map met documenten tegelijk; de log vermeldt welke van de twee er was. Het wordt gelezen terwijl het pakket laadt, dus het mag ook een pakket vereisen waar een huis van afhangt.

Dat is het weten waard voor één ding dat deze uitgave weghaalt. `xifthen` was een vereiste en is dat niet meer, aangezien niets hier een commando van dat pakket gebruikt dat het gewone `ifthen` niet heeft. Het trok `calc` mee, dat deze bundel evenmin ooit gebruikte, dus `calc` is er niet langer vanzelf. Een document dat `\widthof`, `\heightof` of een van de rekenvormen van `calc` schrijft leunde daarop en houdt op met een ongedefinieerd commando; het laadt voortaan zelf `\usepackage{<calc>}`, of een hele map documenten beantwoordt het in één keer in `regulatory.cfg`.

Een document dat andere documenten als bijlage voegt (zie paragraaf 7) heeft de PDF-administratie van $\text{\LaTeX}$ nodig, die geactiveerd wordt met `\DocumentMetadata{<>}` vóór `\documentclass`. Zonder dat wordt iedere bijlage als platte tekst gezet en meldt het pakket dat eenmalig.

Een bron die ook naar HTML wordt omgezet declareert hem voorwaardelijk, en dat is de ene regel om te schrijven. Onder `tex4ht` haalt diezelfde declaratie de lijstcode

van $\LaTeX$ binnen en laat de omzetting niets om een paras onderdeel in om te zetten, zodat de onderdelen als lopende tekst verschijnen; paragraaf 12 bevat de meting. `tex4ht` definieert `\HCode` voordat het het document leest, en daar vraagt de regel naar. Ieder voorbeeld van deze bundel is zo geschreven, en een bouw die het misdoet krijgt dat te horen in plaats van dat het aan de uitkomst wordt overgelaten.

```
\ifdefined\HCode\else\IfDocumentMetadataTF{}{\DocumentMetadata{}}\fi
\documentclass[dutch]{article}
\usepackage{regulatory}
```

Codelijst 2: Eén bron, beide uitvoerformaten

De wacht `\IfDocumentMetadataTF` laat een declaratie die er al staat met rust, en dat is wat dezelfde bron nog een keer laat bouwen met een PDF-standaard ervóór gedeclareerd; zo zijn de gevallen van paragraaf 13.3 gemaakt.

Het voorbeeld van codelijst 1 kan als volgt gegenereerd worden naar PDF:

```
pdflatex main
# Or
lualatex main
# Or keep generating
latexmk -pvc -lualatex -interaction=nonstopmode main
```

Codelijst 3: Commandline voorbeelden

Een document met definitielijsten voegt een stap tussen de $\LaTeX$ -draaien toe:

```
lualatex main
bib2gls main
lualatex main
lualatex main
# Or for bibtex
lualatex main
makeglossaries main
lualatex main
lualatex main
```

Codelijst 4: Commandline met definities

Onder `latexmk` kunnen `bib2gls` en `makeglossaries` in een eigen terminal draaien: `latexmk` ziet de bestanden wijzigen en zet het document opnieuw.

2 Voordat u begint

Wat deze bundel nodig heeft, waaraan hij gemeten is, en wat hij niet kan. Alles hier komt verderop uitgebreid aan bod; dit onderdeel is het korte antwoord op de vraag of regulatory bij een document past voordat dat document eromheen geschreven wordt.

2.1 Engines en vereisten

De engine is `pdflatex` of `lualatex`: beide leveren rechtstreeks PDF, en ieder voorbeeld van deze bundel wordt bij iedere draai van de suite met allebei gezet. Niets hier hangt van een van de twee af, en deze handleiding kent geen verschil tussen beide. `xelatex` en een DVI-route zijn niet getest en worden niet geclaimd.

De route `md` is de uitzondering: `\markdownInput` leest zijn bron via de Lua-interpreter onder `lualatex` en grijpt onder `pdflatex` naar shell escape, dus een Markdown-document zit onder `lualatex` ruimer in zijn jas.

Alle $\TeX$ -afhankelijkheden van de bundel zijn beschikbaar in $\TeX$ Live en $\text{Mi}\mathcal{K}\TeX$: `enumitem`, `fntcount`, `glossaries` en `glossaries-extra`, `hyperref`, `scrextend`, `titlesec` (alleen zolang de nieuwe kopinterface er niet is), `translations`, `xcolor`, `xstring` en de `zref` familie. Twee pakketten die eerdere uitgaven meetrokken zijn weg; zie paragraaf 3.

Buiten $\TeX$ is er één vereiste, en maar op één route: de standaardroute voor definities draait `bib2gls`, dat Java nodig heeft. Paragraaf 5 bevat de drie routes die dat niet doen.

2.2 Definities

Er leiden vier wegen naar een definitielijst, en de keuze bepaalt wat er tussen de $\mathcal{E}\mathcal{T}\mathcal{X}$ -draaien moet lopen. Paragraaf 5 behandelt ze voluit; in één regel per stuk:

`bib2gls` de standaard. Eén $\text{B}\mathcal{E}\mathcal{T}\mathcal{X}$ -bestand bedient ieder document dat dezelfde begrippen aanhaalt, gesorteerd en geselecteerd, en het is wat `\masterdocument` de definities van een ander document laat binnenhalen. Het vraagt Java.

`bib2gls=false` in plaats daarvan `makeglossaries` en `makeindex`, die een bestand met `\newglossaryentry` lezen. Geen Java, nog steeds een programma tussen de draaien, en van de vier de minst beproefde.

`\newdefinition` definities die in het document zelf gedeclareerd worden. Eén $\mathcal{E}\mathcal{T}\mathcal{X}$ -draai en verder niets, tegen de prijs dat geen ander document ze kan aanhalen en dat ze in declaratievolgorde worden afgedrukt.

met de hand de lijst zelf uitgeschreven met `\describe`, die bepaalt wat er opgesomd wordt en in welke volgorde. De definities komen nog steeds uit een van de drie hierboven.

2.3 HTML

Een document wordt omgezet met `make4ht`, dat `tex4ht` aanstuurt. `tex4ht` levert voor dit pakket geen configuratie, dus levert dit pakket `regulatory.4ht` naast de pakketbestanden

mee: een document dat `regulatory` vindt, vindt dat er meteen bij, en zonder dat bestand slaagt een omzetting en levert een document zonder ook maar één kop of sectie.

Eén voorwaarde: `\DocumentMetadata` mag in een HTML-bouw *niet* actief zijn, anders verdwijnt de lijstopmaak van paras. Codelijst 2 is de regel om te schrijven, en paragraaf 12 bevat de metingen.

2.4 Bijlagen

Een ander document in het PDF-bestand insluiten vraagt de PDF-administratie van $\text{\LaTeX}$, geactiveerd met `\DocumentMetadata` vóór `\documentclass`. Zonder dat wordt iedere bijlage als platte tekst gezet en meldt het pakket dat eenmalig.

Hoe de bijlage vanaf de pagina geopend wordt is de optie `attachmentlink`, en geen van beide waarden is voor iedereen de goede: `goto` is de semantisch juiste `embedded go-to action`, die poppler niet implementeert, en `attachmentlink=annotation` is een file attachment annotatie, die iedere gemeten viewer wel opent. Paragraaf 7.1 zet de twee naast elkaar. Hoe dan ook staat het bestand in het bijlagenpaneel van de viewer, dat geen van beide nodig heeft.

2.5 Talen

Het Nederlands en het Engels zijn volledig ondersteund. Het Duits en het Frans leveren de benamingen en het citeerregister van een handeling van de Unie mee, allebei gemeten, en *niet* de verwoording van een interne verwijzing: een Duits document krijgt Duitse woorden in de Engelse opbouw. Iedere andere taal valt volledig terug op het Engels, en krijgt dat één keer te horen. Paragraaf 10 bevat de tabel en hoe ieder geval er op de pagina uitziet.

2.6 PDF-conformiteit

Welke standaard een document kan dragen is geen eigenschap van dit pakket maar van het document: van zijn tagging, van wat het bijvoegt, en van waar het naar verwijst. Paragraaf 13.3 is de meting, gedraaid door een op digest gepinde veraPDF, en dat is wat de claims hieronder waard zijn. Samengevat, voor de documenten van deze bundel:

Profiel	Gemeten	Waarop
PDF/A-1	buiten bereik	verbiedt ingesloten bestanden categorisch
PDF/A-2b	slaagt	gewoon, en ondertekend
PDF/A-2b	faalt	met een bijlage die geen PDF is
PDF/A-2a + UA-1	slaagt	getagd, met <code>attach-css=false</code>
PDF/A-3	buiten bereik	verwijzend naar bepalingen van een ander document
PDF/A-3a + UA-1	slaagt	getagd, met een PDF als bijlage
PDF/A-4	slaagt	getagd, met <code>attach-css=false</code>
PDF/A-4f + UA-2	slaagt	getagd, met welke bijlage dan ook

Lees dat als: *dit document in deze samenstelling is gemeten en slaagde*, en nooit als “*regulatory is PDF/A-compatibel*”. De regels verschillen elk in één ding, en dat is wat ze iets waard maakt.

2.7 Bekende beperkingen

PDF/A-3 een document dat naar de *bepalingen* van een extern document verwijst haalt het niet: de `/GoToR`-link wordt geschreven met een bestandsspecificatie die faalt op regel 6.8-4 van ISO 19005-3, en die specificatie wordt niet door dit pakket geschreven. Een ander document bijvoegen en zijn definities aanhalen kost niets. Zie paragrafen 7 en 13.3.

Talen het Duits en het Frans kunnen een interne verwijzing en de naam van een ander document niet verwoorden; zie paragraaf 10. Geen van beide heeft een nationaal citeerregister.

Sorteren `\loadglsdefs` sorteert onder de Nederlandse collatie `n1-NL`, in welke taal het document ook geschreven is. Een document in een andere taal dat een eigen collatie wil, drukt zijn lijst af met `\printunsrtglossary` na een eigen `\GlsXtrLoadResources`, of maakt hem met de hand op met `\describe`.

HTML de omzetting vraagt dat `\DocumentMetadata` er niet is, en dat is dezelfde declaratie die de bijlagen nodig hebben, dus één bron kan niet in dezelfde draai omgezet worden én bijlagen dragen. De definitiestijl `almtree` levert in HTML geen enkele lijststopmaak; `labeling` en `description` wel. Zie paragraaf 12.

Markdown een Markdown-bron bevat geen $\TeX$: een backslash wordt afgedrukt in plaats van uitgevoerd, en dat zonder melding. Alleen de onderdelen van paragraaf 11 hebben een tegenhanger, en een bracketed span mag geen tweede span en geen link met blokhaken bevatten, en daar is de identifier van paragraaf 11.2 voor.

`\aref` bij meer dan één label heeft `nameinlink` geen effect en wordt de hyperlink alleen om de nummers getrokken.

`\loadsources` de bib lezer neemt een beperkte, regelgebaseerde syntaxis aan en is geen $\text{Bib}\TeX$ -parser; een volstrekt gewoon bib-bestand kan best geweigerd worden. Paragraaf 9 geeft de grammatica.

Juriconnect deze uitgave legt de datum waarop een aanhaling spreekt vast en bewaakt hem, en drukt hem niet af en bouwt geen Juriconnect-verwijzing. Zie `\sourcedate` in paragraaf 9.

Regelgeving er komt geen regelgeving met deze bundel mee. Hij bevat het gereedschap om instrumenten aan te halen en niet de instrumenten: een citeertitel verandert op zijn eigen klok en een pakket op CTAN niet.

2.8 Wat de tests vaststellen

Eén ding is het waard om nu al te zeggen, want het is waar de beweringen in deze handleiding op rusten. Een document dat zonder fout gegenereerd wordt is daarmee nog niet correct. Vrijwel alles wat deze bundel verkeerd kan doen komt er evengoed als afgerond

document uit: een aanhaling in het verkeerde register, een definitie die stilzwijgend niet aankwam, een label dat nergens heen wees, een kop die een vetgedrukte zin werd.

De suite controleert dus niet of een bouw slaagt. Zij leest de uitkomst terug — verwijzingen en de labels waar ze op uitkomen, definities en de volgorde waarin een programma ze sorteert, de bewoording van veertig aanhalingen woord voor woord, de secties en koppen van de HTML-omzetting, de annotaties en structuurelementen in het PDF-bestand zelf, en de waarschuwingen die het pakket wel en niet gaf. Twee documenten staan er om te *falen* en twee om te *waarschuwen*, want een garantie is zoveel waard als haar weigering. Paragraaf 13 behandelt het geheel.

2.9 Publiek, uitbreiding en intern

Er komen in deze handleiding drie soorten namen voor, en ze dragen verschillende beloften voor de 1.x-uitgaven.

Publieke API wat een gewoon document schrijft: de omgevingen `paras`, `provisions`, `definitions` en `externals`; `\article` en `\para`; de families `\rref`, `\nref` en `\aref` en de koppelcommando's; `\loadglsdefs`, `\newdefinition`, `\printdefs` en `\describe`; `\refdocument`, `\masterdocument`, `\newartifact` en de `\document...` commando's; `\signaturefield`; `\loadsources`, `\newsources`, `\srcref`, `\srcfield`, `\srctype`, `\ifsourceexists` en `\sourcedate`; `\autocitedefs` en de Markdown-constructies; en de pakketopties. Die blijven de hele 1.x-reeks werken, en waar er een moet veranderen blijft de oude schrijfwijze als alias staan.

Extension API wat een taalbestand, een citeerregister of een integratie schrijft: `\ProvidesRegulatoryLanguage` en `\RegulatoryLoadLanguage`; `\rref@setup` en de formaatcommando's die hij meekrijgt, `\rref@reformat@...`, `\rref@label@...` en `\rref@group@braced`; `\newsourceregister`, `\newsourcedeterminer`, `\newsourcetype`, `\newsourcetypealias`, `\newsourcefieldalias` en de `\regulatory@src@...` formaten waaruit een register wordt opgebouwd; `\regulatorysetup` en `regulatory.cfg`; en `\regulatory@ifmodule`. Verscheidene daarvan dragen een apenstaartje, en dat is geen slordigheid: ze zijn bedoeld om in een `.def` bestand geschreven te worden, dat gelezen wordt met het apenstaartje als letter. Ze worden zoals ze zijn de hele 1.x-reeks ondersteund.

Intern al het overige, waaronder iedere `\regulatory@...`, `\artifact@...` en `\rref@...` die hierboven niet genoemd is, en iedere l3 naam met een dubbele underscore. Die staan in paragraaf 14 omdat de implementatie gedocumenteerd is en niet omdat ze gebruikt mogen worden; ze veranderen zonder aankondiging. Twee artefactvelden horen hier ook en staan beschreven bij `\refdocument`: `referred`, dat het pakket zelf zet, en `url`, dat gereserveerd is en niets doet.

3 Overstappen naar 1.0

Waar een document dat voor een eerdere uitgave geschreven is tegenaan loopt. Alles hieronder is een verandering die van buiten het pakket te zien is; de rest van 1.0 is aanvulling.

- calc** **Dit is degene die een bouw stopt.** `xifthen` is geen vereiste meer, aanzien niets hier een commando van dat pakket gebruikte dat het gewone `ifthen` niet heeft. Het trok `calc` mee, dat deze bundel evenmin ooit gebruikte, dus `calc` is er niet langer vanzelf. Een document dat `\widthof`, `\heightof` of een van de rekenvormen van `calc` schrijft leunde daarop en houdt nu op met een ongedefinieerd commando. De oplossing is `\usepackage{calc}` in het document, of `\RequirePackage{calc}` in een `regulatory.cfg` voor een hele map ervan. Deze handleiding is een van de documenten die het betrapte, en vraagt zelf om `calc`.
- Markdown** de module werd geladen met de optie `hybrid` van `markdown`, die iedere backslash doorliet, en bronnen die zo geschreven waren schreven hun labels en verwijzingen als $\TeX$. `markdown` heeft die optie afgeschaft. Zonder haar wordt een backslash *afgedrukt* in plaats van uitgevoerd, dus zo'n bron verliest zijn verwijzingen zonder dat er ook maar iets misgaat. Alles wat die optie nodig had wordt nu in Markdown geschreven; zie paragraaf 11, en lees een oudere bron één keer door.
- Talen** een niet-ondersteunde taal leent niet langer de woorden van het taalbestand dat toevallig als laatste geladen was. Gemeten aan hetzelfde Spaanse document: een eerdere uitgave gaf het Nederlandse koppen en liet de benamingen stilzwijgend uit de verwijzingen weg, waar 1.0 alles in het Engels verwoordt en één keer zegt voor welke taal het geen definities heeft. Een document dat stilzwijgend op het eerste leunde ziet er anders uit en zegt nu waarom.
- Structuur** één pakket werd een bundel. Eerdere uitgaven leverden één `regulatory.sty`; 1.0 levert er acht, waarvan `regulatory` er zes laadt, plus `regulatory.4ht` en de taalbestanden. Een document dat het pakket zoals altijd laadt merkt er niets van, en wie met de hand installeert kopieert meer bestanden.
- Handtekeningen** `regulatory-sign` is waar `xdp-sign` heen ging. `\SignatureField` luistert nog steeds, zodat een document dat voor dat pakket geschreven is blijft werken wanneer het overkomt. Een handtekening kost het document zijn PDF/A-2b conformiteit niet meer.
- Bron sleutels** de types en velden van paragraaf 9 hebben Engelse namen, en iedere Nederlandse naam is daar een alias op, dus een `bib`-bestand dat op de ene manier geschreven is wordt op beide manieren gelezen. Twee namen zijn veranderd, en ze verschillen in wat er van de oude geworden is. De optie `bronsstijl` werd `sourcestyle` en heeft *wel* een alias: `bronsstijl` luistert nog steeds, en de waarden `voluit` en `verkort` ook, dus een document dat hem

gebruikte hoeft niets te veranderen. De vertaalsleutel subparagraaf werd point en heeft *geen* alias, want buiten deze bundel had nog niemand hem opgevraagd: een document dat die vertaling bij naam opvroeg vraagt nu om point.

4 Structuur

Dit pakket levert bekende structuren, zonder iets van $\LaTeX$ weg te nemen. De drie niveaus heten *artikel*, *lid* en *onderdeel*, waar dit pakket mee begonnen is.¹

`\article` `\article{<titel>}` en `\para{<titel>}` zijn de twee koppen. Deze zijn als aparte macro's gedefinieerd en opgemaakt met `titlesec`, of, zodra de nieuwe $\LaTeX$ -kopinterface beschikbaar is, met een `heading`-instantie. Beide doen mee in de inhoudsopgave, op het niveau van subsecties en subsubsecties.

`paras` (*env.*) De omgeving `paras` bevat de twee niveaus onder een artikel, gebouwd met `enumitem`. Een lid krijgt het label `\thearticle.\arabic*`, en herhaalt daarmee het nummer van het artikel waar het bij hoort, en een onderdeel het label `\abalphnum{\arabic*}`.

Die `\abalphnum` komt uit `fmtcount`² en is wat een artikel voorbij 26 onderdelen laat lopen: `\alph` houdt daar op, terwijl `\abalphnum` van bijvoorbeeld 125 'du' is.

`provisions` (*env.*) Niet ieder document is in artikelen verdeeld. De omgeving `provisions` is een vlakke opsomming, genummerd binnen de `\section` waarin hij staat in plaats van binnen een artikel, voor een tekst met één niveau en zonder eigen koppen.

```
\article{...}
\begin{paras}
  \item ...
  \begin{paras}
    \item ...
    \item ...
  \end{paras}
  \item ...
\end{paras}

\article{...}
...

\para{...}
...

\section{...}
\begin{provisions}
  \item ...
  \item ...
\end{provisions}
```

¹De Engelse namen zijn ingekort waar $\LaTeX$ de volledige al bezet houdt: een kop van het tweede niveau is `\para` en niet `\paragraph`, en zijn lijst heet `paras` en niet `paragraphs`. Welk van de twee woorden 'section' en 'paragraph' welk niveau aanduidt, verschilt per rechtsorde, en geen van beide lezingen is die welke $\LaTeX$ ermee bedoelt.

²De Nederlandse taaldefinitie is nu inbegrepen in `fmtcount` zelf. Werk het pakket bij wanneer `fmtcountch.def` ontbreekt.

`\end{provisions}`

Zie codelijsten 5 en 6 in paragraaf 13.4 voor meer $\text{\LaTeX}$ -voorbeelden.

5 Definities

Definities worden aangehaald met de `\gls{<label>}` familie van `glossaries-extra`, die deze module laadt en instelt.

`definitions (env.)` Opdat begrippen, afkortingen en definities niet botsen voegt dit pakket twee `glossary-` types toe. Definities van dit document komen onder `definitions`; definities van een ander document onder `externals` (zie paragraaf 7).

Waar de definities vandaan komen is een vraag op zich, en het antwoord ligt niet vast. Er staan vier wegen open, en hoe de definities binnenkomen staat los van hoe de lijst gezet wordt:

`bib2gls` de standaard. `\loadglsdefs{<src>}` leest een `BIBTEX`-bestand, dat `bib2gls` sorteert en waaruit het selecteert. *Voor:* één bestand bedient ieder document dat dezelfde begrippen aanhaalt, de opsomming wordt gesorteerd tegen het woordenboek van de taal, en alleen de definities die het document werkelijk gebruikt verschijnen — of allemaal met `alldefs`. Het is ook wat `\masterdocument` in staat stelt de definities van een ander document binnen te halen. *Tegen:* het vraagt Java en een `bib2gls` draai tussen de twee `TEX`-draaien.

`bib2gls=false` `\loadglsdefs{<src>}` leest dan een bestand met `\newglossaryentry` regels. *Voor:* geen Java. *Tegen:* het sorteren blijft aan `makeindex`, dus er staat nog steeds een programma tussen, en van de vier is dit de weg waarop het pakket het minst beproefd is.

`\newdefinition` `\newdefinition{<label>}{<naam>}{<omschrijving>}` declareert een definitie in het document zelf. *Voor:* geen tweede bestand en geen tweede programma — één `TEX`-draai volstaat. *Tegen:* de definities horen bij dit document, dus geen ander document kan ze aanhalen, en ze worden gezet in de volgorde waarin ze gedeclareerd zijn in plaats van gesorteerd.

met de hand de lijst wordt zelf uitgeschreven, met `\describe{<label>}` waar iedere omschrijving hoort. De definities zelf komen nog steeds uit een van de drie wegen hierboven. *Voor:* het artikel bepaalt welke begrippen het opsomt, in welke volgorde, en wat er tussen staat. *Tegen:* de lijst wordt met de hand onderhouden, dus een nieuwe definitie verschijnt er niet vanzelf in.

Alleen de tweede weg vraagt om een programma tussen de `TEX`-draaien. De andere twee leveren definities aan die al in de volgorde staan waarin ze gezet moeten worden, en daarom grijpt `\printdefs` daar naar een ander zetcommando — zie paragraaf 14.

Hoe de opsomming eruitziet is een tweede keuze, los van de eerste. `\printdefs` neemt de stijl als optioneel argument en de optie `defstyle` zet de standaard voor het document. Er zijn er drie: `alttree`, de standaard die `glossaries` meebrengt en die een naam boven zijn omschrijving zet; `labeling`, dat iedere omschrijving uitlijnt op de breedte die aan

`\printdefs` is meegegeven; en `description`, dat de uitlijning aan de klasse laat. Iedere andere stijl die `glossaries` kent kan net zo goed genoemd worden. De lijst met de hand uitschrijven, de vierde weg hierboven, gebruikt dezelfde twee omgevingen, dus de opmaak komt overeen hoe de lijst ook tot stand komt.

Die keuze reikt verder dan de gedrukte pagina. Gemeten aan een lijst van twee definities, omgezet met `make4ht`: `almtree` levert in het geheel geen lijstopmaak, terwijl `labeling` en `description` beide een `<dl>` geven met per definitie een `<dt>` en een `<dd>`. Een document dat ook als HTML verschijnt (paragraaf 12) heeft daarmee een reden om een van de twee te kiezen.

`\printdefs` `\printdefs{<breedte van tekst>}` drukt de definitielijst af. De `<breedte van tekst>` is de `[<style>]` breedste naam waarop de lijst moet uitlijnen, en dat is wat de stijl `labeling` gebruikt; de `{<width of text>}` andere stijlen doen er niets mee.

`\describe` `\describe{<label>}` drukt de beschrijving van één definitie af op de plek waar hij staat, wat de manier is om een definitieartikel met de hand op te maken in plaats van als lijst. Hij voegt het ankerpunt toe dat de hyperlinks van `\gls` nodig hebben, en dat is wat hem onderscheidt van het afdrukken van de beschrijving met `\glsentrydesc`. Een zo opgebouwde lijst wordt uitgelijnd door wat eromheen staat, dus `\glssetwidest` is alleen van belang wanneer dat een eigen `labeling` omgeving is.

`\newdefinition` `\newdefinition` declareert één definitie in het document zelf, onder hetzelfde type en dezelfde categorie als die uit een bestand, zodat hij in dezelfde opsomming terechtkomt.

`{<label>}`
`{<name>}`
`{<description>}` `\loadglsdefs{<src>}` leest een `BIBTEX`-bestand in onder het type `definitions` en de categorie `definitions`. De definities die het document gebruikt worden opgesomd zodra `\printdefs` wordt aangeroepen, of allemaal met de optie `alldefs`. De opsomming wordt gesorteerd volgens de Nederlandse collatie, `n1-NL`, in welke taal het document ook geschreven is — zie paragraaf 2.7, waar staat wat eraan te doen is.

`\loadextdefs` `\loadextdefs[<categorie>]{<src>}` leest de definities van een ander document in, onder een eigen categorie zodat definities uit verschillende bronnen uit elkaar blijven. `\masterdocument` roept hem al met de juiste categorie aan, dus een document roept hem zelden zelf aan. Hij hoort bij `regulatory-attachments` en wordt beschreven in paragraaf 7.

6 Verwijzingen

Verwijzingen naar artikelen, leden en onderdelen zijn gebouwd op `zref`, waarvoor iedere structuur uit paragraaf 4 is ingesteld. `zref` zelf biedt minder greep op de bewoording dan `cleveref`.

`\rref` Daarom verwoordt deze bundel zijn eigen verwijzingen, te beginnen met `\rref{⟨label⟩}`,
`\Rref` die naar een artikel verwijst zoals `\ref` naar een `\section` verwijst. De familie heeft vier leden:

`\rref` Beginnend met een kleine letter en met hyperlink.

Artikel	Lid	Onderdeel
2	eerste	a

`\rref*` Beginnend met een kleine letter en zonder hyperlink.

Artikel	Lid	Onderdeel
2	eerste	a

`\Rref` Beginnend met een hoofdletter en met hyperlink.

Artikel	Lid	Onderdeel
2	Eerste	A

`\Rref*` Beginnend met een hoofdletter en zonder hyperlink.

Artikel	Lid	Onderdeel
2	Eerste	A

De voorbeelden hierboven laten al zien waarin dit van `\zref` verschilt: een verwijzing is niet het nummer dat de kop draagt. Het lid `ex1-lid:lorem` is op de pagina genummerd als ‘2.1’ en wordt aangehaald als eerste.

`\nref` De familie `\nref{⟨label⟩}` zet de benaming bij het nummer, en heeft dezelfde vier leden
`\Nref` als `\rref`. De tabel hieronder toont alleen `\Nref`.

	Artikel	Lid	Onderdeel
NL	Artikel 2	Eerste lid	Onderdeel a
EN	Article 2	(1)	Point (a)

Waarin `\nref` van `\zcref` verschilt, is dat de benaming niet vóór zijn nummer hoeft te staan: het Nederlands schrijft een lid als *⟨rangtelwoord⟩ ⟨benaming⟩* en een onderdeel als *⟨benaming⟩ ⟨letter⟩*, en de taal bepaalt welke van de twee.

`\rref` en `\nref` dragen elk één niveau. `\nref` geeft een lid zijn benaming maar niet het artikel waar het bij hoort, en een verwijzing naar een lid die zijn artikel niet noemt, `\aref` noemt de helft van wat ze bedoelt. `\aref{⟨labels...⟩}` schrijft de hele verwijzing uit: ie-
`\Aref` der niveau tot aan het genoemde, in de volgorde en de bewoording die de taal vraagt. Hij neemt meer dan één label en koppelt ze, als opsomming, artikel 2, tweede lid, onderdelen a, c en d, van Voorbeeld Één³, of als reeks, artikel 2, tweede lid, onderdelen a tot d, van Voorbeeld Één. Eén beperking: bij meer dan één label heeft `nameinlink` geen effect

³Voorbeeld Één

en wordt de hyperlink alleen om de nummers getrokken. Dezelfde labels lezen zoals de taal van het document dat doet:

```
\selectlanguage{dutch} / \selectlanguage{english}
Zie / See \aref{ex1-sub:lorem,ex1-sub:lorem3,ex1-sub:lorem4}
en / and \aref{ex1-sub:lorem,ex1-sub:lorem2,ex1-sub:lorem3,ex1-sub:lorem4}.
```

See article 2(2), points (a), (c) and (d), van Voorbeeld Één and article 2(2), points (a) to (d), van Voorbeeld Één.

Wat de taal niet volgt is de verwoording die een *ander* document noemt: dat is de ref label van paragraaf 7, die één keer geschreven wordt en blijft zoals ze geschreven is.

6.1 Conjunctie

```
\setmiddleconjunction Labels worden via zref gekoppeld, en welke woorden ze koppelen wordt ingesteld zoals
  {\format}} cleveref dat doet:
\setlastconjunction \setmiddleconjunction{, }
\setrangeconjunction \setlastconjunction{ \GetTranslation{and} }
\setconjunction \setrangeconjunction{ \GetTranslation{to} }
  {\middle} \setconjunction{, }{ \GetTranslation{and} }{ \GetTranslation{to} }
  {\last}
  {\range}
```

\setjuncto Twee andere schakelen naar de oudere notatie en terug. \setjuncto maakt het laatste koppelwoord vanaf dat punt ‘ jo.\ ’, en \unsetjuncto zet ‘ en ’ terug. Beide overschrijven wat er eerder met \setlastconjunction is meegegeven, dus een document dat een eigen bewoording heeft gezet herstelt die met \setlastconjunction{\waarde} en niet met \unsetjuncto. De pakketoptie legacy roept \setjuncto meteen aan.

7 Interdocumentaire verwijzingen

Een document dat met dit pakket geschreven is verwijst naar de artikelen, leden, onderdelen en definities van een ander document dat daarmee geschreven is.

Voordat u een documentarchitectuur hierop bouwt. Een document dat naar de bepalingen van een extern document verwijst haalt **PDF/A-3** niet. Iedere zo'n verwijzing wordt een /GoToR-actie die naar een bestandsspecificatie wijst die /AFRelationship/Unspecified draagt, geen /EF heeft en niet in /AF staat, wat faalt op regel 6.8-4 van ISO 19005-3. Die specificatie wordt voor de link geschreven, door de PDF-administratie van L^AT_EX en niet door dit pakket, dus het profiel blijft buiten bereik tot dat verderop opgelost is.

Gemeten aan de twee voorbeelddocumenten: `example1-nl` maakt elf van zulke verwijzingen en draagt één zo'n specificatie, en faalt daarop voor PDF/A-3b; `example2-nl` verwijst naar geen enkele bepaling van een ander document, *voegt* er een bij en haalt diens definities aan, heeft geen enkele /GoToR, en haalt PDF/A-3a. Een ander document bijvoegen en zijn definities aanhalen kost het profiel dus niet — naar zijn artikelen, leden en onderdelen verwijzen wel. Ieder ander profiel in paragraaf 13.3 blijft hoe dan ook ongemoeid.

`\refdocument` `\refdocument` in de preamble is wat de artikelen, leden en onderdelen van een ander document aanhaalbaar maakt; `\aref` leest diens labels dan via `\xexternaldocument` van `{<name>}` `zref-xr`. Beide documenten kunnen hetzelfde label gebruiken, dus ieder label van het andere krijgt een `prefix, ext-` tenzij er een andere genoemd wordt: naar `lid: lorem` wordt verwezen als `ext-lid: lorem`. De prefix geldt voor verwijzingen en niet voor definities, die `\gls` onder hun eigen label aanhaalt.

`\masterdocument` `\masterdocument` legt de hele koppeling en niet de helft ervan. Hij doet wat `\refdocument` doet en daarbij nog drie dingen:

- `{<name>}`
 - `{<opts...>}` 1. verwijzen met de `\aref` familie;
 - 2. definities van dat document, aangehaald met de `\gls` familie;
 - 3. het document genoemd in de zin die het aanhaalt;
 - 4. een voetnoot bij de eerste verwijzing of definitie, met het document zelf als bijlage ingesloten in het PDF-bestand.

Een document kan meer dan één masterdocument hebben, zoals deze handleiding:

```
\newcommand\definitionlabel[1]{~(zie #1)}
\masterdocument[ex1-]{example1-nl}{
  path=../test/,
  defs=example1,
  author=E. Nijenhuis,
  subject=Voorbeeld Één,
  description=Het éérste voorbeeld document,
  ref label=van Voorbeeld Één,
  def label=\definitionlabel
```

```

}

\masterdocument[ex2-]{example2-nl}{
  path=../test/,
  defs=example2,
  author=E. Nijenhuis,
  subject=Voorbeeld Twee,
  description=Het tweede voorbeeld document,
  ref label=van Voorbeeld Twee,
  def label=\definitionlabel
}

```

Het derde argument van `\refdocument` en `\masterdocument` bevat de velden hieronder. Een veld dat dit pakket niet kent is een fout waar het gegeven wordt, en niet een waarde die onder een naam wordt opgeslagen die niemand terugleest.

- `name` standaard het eerste argument van de macro. Het is de naam waaronder het aux-bestand van het andere document gelezen wordt.
- `filename` standaard de name met `.pdf` erachter, voor een document waarvan het bestand niet naar hem genoemd is.
- `path` standaard `./`, geplaatst vóór zowel de naam als de bestandsnaam, voor documenten die in een andere map staan.
- `ref label` de verwoording die het andere document in een zin noemt, zoals in “artikel 2 *van de algemene voorwaarden*”. Hij is standaard leeg, en `\documentlabel` stelt er dan zelf een samen uit de subject: `\↵ GetTranslation{of the} \artifactssubject{#1}`. Niet iedere taal kan die samenstellen; zie paragraaf 10, waar het Duits en het Frans dat hardop zeggen.
- `def label` wat er achter een definitie staat die uit het andere document komt. De standaardwaarde is `~(\GetTranslation{see} #1)`. Het argument is de subject, met de voetnoot van de eerste verschijning eraan vast.
- `footnote` standaard `true`, zodat de eerste verwijzing of definitie een voetnoot draagt met het document eraan gehecht. Met `false` blijft de voetnoot achterwege.
- `footnote label` wat die voetnoot zegt. Het argument is de bijlage, met de subject als de tekst waaronder hij wordt aangeboden; standaard drukt de macro dat argument af en verder niets.
- `defs` de definities van het andere document, onder dat document ingeladen, zodat een definitie die eruit wordt aangehaald noemt waar hij vandaan komt.

`author` de auteur van het ingesloten bestand, die een PDF-viewer bij de bijlage toont.

`subject` de naam waarmee dit document het andere aanduidt. Het is waar `ref label` en `def label` op terugvallen, en het is de omschrijving van het ingesloten bestand.

`description` een langere omschrijving van het ingesloten bestand, die een viewer bij de bijlage toont.

Er worden nog twee velden aanvaard die niet door een document gezet horen te worden; zie paragraaf 2.9 voor wat dat voor een uitgave betekent. `referred` legt vast dat een artefact zijn voetnoot gehad heeft, wat `\documentfootnote` zet en waarmee een document dat het met de hand zet enkel een voetnoot onderdrukt die het nooit gezien heeft. `url` is gereserveerd: hij wordt opgeslagen, hij is met `\artifacturl` te lezen, en niets in deze bundel leest hem. Het was de bedoeling dat hij de vindplaats van het andere document zou noemen, in de voetnoot naast de bijlage, en dat is niet geïmplementeerd. De naam blijft daarvoor vrijgehouden.

<code>\documentlabel</code>	Een verwijzing of een definitie die uit een ander document komt noemt dat document,
<code>{\label}</code>	en de eerste van beide draagt een voetnoot met dat document erin ingesloten. Drie
<code>\documentfootnote</code>	commando's doen dat samen: <code>\documentlabel</code> ⁴ verwoordt de naam, <code>\documentfootnote</code>
<code>[\link text]</code>	plaatst de voetnoot, en <code>\documentattachment</code> sluit het bestand in en biedt het aan onder
<code>{\label}</code>	de tekst die hij meekrijgt. Ieder ander bestand wordt ingesloten met <code>\attachdocument</code> ,
<code>\documentattachment</code>	die geen eigen artefact nodig heeft.
<code>{\label}</code>	
<code>{\link text}</code>	Deze commando's kunnen ook met de hand aangeroepen worden. Neem bijvoorbeeld
<code>\attachdocument</code>	<code>\documentfootnote{example2}</code> , wat geeft: ⁵ '.
<code>[\description]</code>	
<code>{\filename}</code>	
<code>{\link text}</code>	

7.1 Welke link een bijlage opent

De optie `attachmentlink` bepaalt wat `\documentattachment` op de pagina zet. Het bestand zelf wordt hoe dan ook ingesloten en verschijnt in het bijlagenpaneel van de viewer, dat helemaal geen link nodig heeft; wat verschilt is wat er gebeurt waar de tekst staat.

`goto`, de standaard, maakt van die tekst een hyperlink met een embedded go-to action, die een bestand *binnen dit document* noemt. Dat is wat er te zeggen valt, en het is de ene actie die poppler niet implementeert. Gemeten in de gedeelde bibliotheek in plaats van gelezen in een handleiding: die noemt `GoTo`, `GoToR`, `Launch` en `Named` onder zijn acties en `GoToE` nergens, en `FileAttachment` onder zijn annotaties. poppler is de motor onder de meeste viewers op een vrij bureaublad, dus daar doet die link niets.

`attachmentlink=annotation` legt in plaats daarvan een file attachment annotatie over dezelfde tekst, waar Acrobat en poppler allebei gehoor aan geven. Hij laat zelf niets zien — de tekst blijft het enige op de pagina — en de annotatie is wat opent.

⁴Deze verwoordt de naam voor een verwijzing en niet voor een definitie, die een eigen `def label` heeft.

⁵Voorbeeld Twee

	goto	annotation
PDF-constructie	embedded go-to action	file attachment annotatie
opent in Acrobat	ja	ja
opent in poppler	nee	ja
in het bijlagenpaneel	ja	ja
structuurelement	Link	Annot
linktekst erin	ja	nee, ernaast
gemeten PDF/A-4f	slaagt	slaagt
gemeten PDF/UA-2	slaagt	slaagt

De twee viewerregels zijn wat de twee constructies zijn; de twee structuurregels zijn gemeten, en zij zijn de reden dat de standaard voor deze uitgave niet veranderd is. Een bijlage-annotatie is een markup-annotatie, en PDF/UA vraagt dat zo een in een Annot-structuurelement zit. De annotatieroute zet hem daarin, en de tekst waar hij naast staat blijft daarvan een broer in plaats van zijn inhoud, terwijl de route goto van die tekst juist de inhoud van zijn Link-element maakt. Gemeten aan hetzelfde document: goto geeft een Link met twee kinderen, de gemarkeerde inhoud van de tekst en de objectverwijzing; annotation geeft een Annot met één. Die twee op de annotatieroute samenbrengen is een klus op zich en zit niet in deze uitgave.

Neem een validator daarin niet op zijn woord, welke kant ook op. De veraPDF waarmee deze handleiding gemeten is meldde niets toen het Annot-element helemaal ontbrak en een oudere meldde het wel, dus de suite leest de structuurboom zelf uit het PDF-bestand in plaats van om een oordeel te vragen; paragraaf 13.2 bevat die controle.

Wat overblijft is een vraag over documenten en niet over PDF. Een verwijzing die een lezer in de loop van een zin tegenkomt is een ander aanbod dan een die ernaast staat, en goto is de route die het aanbod in de zin doet. Een document waarvan de lezers op een vrij bureaublad zitten en de bijlage vanaf de pagina horen te openen zegt attachment link=annotation en krijgt de andere ruil.

8 Handtekeningen

Een overeenkomst wordt ondertekend, en een PDF kan het veld dragen om dat in te doen. De module `regulatory-sign` tekent zo'n veld en legt er een handtekeningwidget overheen, zodat een lezer die het document opent in een viewer die kan ondertekenen het veld aangeboden krijgt waar de lijn staat.

De module komt uit `xdp-sign` van Xerdi's Documentation Project en wordt hier onderhouden, zodat een document dat een handtekening nodig heeft niet afhangt van een pakket dat in geen enkele distributie zit.

`\signaturefield` Tekent een handtekeningveld van $\langle breedte \rangle$ bij $\langle hoogte \rangle$. De $\langle naam \rangle$ is waaronder een viewer de handtekening bewaart en waar een ondertekentool om vraagt, dus een sprekende naam loont. De optionele $\langle tooltip \rangle$ zegt waar het veld voor is, en dat is ook wat een schermlezer voorleest.

$\{\langle width \rangle\}$ Het veld wordt via de kernel geplaatst wanneer `\DocumentMetadata` is gedeclareerd, en anders via de engine. Een document dat geen van beide biedt krijgt een waarschuwing en een getekend kader zonder veld: een lijn om op papier te tekenen, maar niets om in een viewer te ondertekenen.

$\{\langle height \rangle\}$

```
\article{Ondertekening}
\begin{paras}
  \item De opdrachtgever tekent hieronder.\
    \signaturefield[Handtekening van de opdrachtgever]{opdrachtgever}{6cm}{2cm}
\end{paras}
```

`\SignatureField` is de naam die dit commando in `xdp-sign` had en waar het nog steeds op luistert, zodat een document dat voor dat pakket geschreven is blijft werken wanneer het overkomt.

Een handtekening kostte het document zijn PDF/A-2b conformiteit en doet dat niet meer. Regel 6.3.3-1 vraagt van iedere annotatie die geen `Popup`, geen `Link` en niet van nul-formaat is een appearance dictionary, en een widget wordt dat ook gevraagd; veraPDF zei van een veld zonder “An annotation does not contain an appearance dictionary”. Ieder veld wijst nu naar één lege verschijning, die ze allemaal delen: de lijn om op te tekenen wordt op de pagina gezet en niet door de annotatie, dus een verschijning die iets tekende zou het dubbel tekenen, en een ondertekentool vervangt de normale verschijning door de zijne zodra het veld getekend wordt. Het geval staat in de tabel van paragraaf 13.3 en slaagt; het harnas maakt van een onverwacht geslaagd geval een fout, en zo heeft dit geval zich gemeld.

9 Externe bronnen

Een regulatorisch document haalt instrumenten aan die dit document niet zijn: een wet, een verordening van de Unie. `regulatory-sources` houdt de bronnen bij die een document aanhaalt en verwoordt de aanhalingen ervan, zodat hoe een aanhaling luidt een eigenschap is van de bron en van het document en niet van de zin waar hij toevallig in staat.

De module doet drie dingen. Hij *leest* bronnen, uit een bib-bestand (`\loadsources`) of uit het document zelf (`\newsources`), en legt ze naast de velden die hun type vereist. Hij *bewaart* ze, zodat `\srcfield` en `\src` type teruggeven wat er gedeclareerd is. En hij *verwoordt* een aanhaling ervan, in zijn geheel of tot op een bepaling, met `\src` — en daar komen de citeerregisters hieronder bij kijken.

Twee dingen staan met opzet elders. De registers zelf staan in de taaldefinitiebestanden van paragraaf 10, aangezien hoe een rechtsorde een aanhaling verwoordt dezelfde kennis is als wat die taal een lid noemt. En er komt geen regelgeving met deze bundel mee: een citeertitel verandert op zijn eigen klok en een pakket op CTAN niet.

`\loadsources` Leest $\langle bestand \rangle$.bib. De lezer is van deze module zelf, zodat een document voor een $\langle file \rangle$ handvol records geen tweede programma nodig heeft — en hij is **geen BibTeX-parser**. Hij leest een bestand regel voor regel en kent vier vormen, die elk een hele regel beslaan:

commentaar een lege regel, of een regel waarvan het eerste niet-blanco teken % is. Dat teken is van deze lezer en niet van BibTeX, dat helemaal geen commentaartekenen kent.

entry-kop $@\langle type \rangle \{ \langle key \rangle \}$, en verder niets op die regel. De sleutel bevat geen komma, geen spatie en geen accolade.

veld $\langle name \rangle = \{ \langle value \rangle \}$, en verder niets op die regel. De waarde staat tussen accolades, en de komma aan het eind is optioneel.

sluitaccolade $\}$ alleen op een regel, wat de entry afsluit.

Een regel die geen van de vier is stopt de draai, met het bestand, het regelnummer en de regel zelf erbij. Dat is de bedoeling van de beperking: een lezer die overslaat wat hij niet lezen kan laat een aanhaling leeg in een document dat er verder goed uitziet, en paragraaf 13.1 bevat de twee documenten die hem daaraan houden.

Dit wordt hier dus geweigerd, terwijl het allemaal gewoon BibTeX is: een waarde tussen dubbele aanhalingstekens (`title = "x"`); een kale waarde (`year = 2024`); een hele entry op één regel; een veld over meerdere regels; iets achter de sluitaccolade van een waarde op dezelfde regel; `@string`-afkortingen en samenvoeging met #; en `@comment`-blokken. Een waarde wordt bewaard zoals hij er staat en wordt nooit geëxpandeerd, dus een macro die in een bestand staat komt als die macro terug.

```
@regulation{bw6,
  citetitle = {Burgerlijk Wetboek},
  abbreviation = {BW},
  book = {6},
  bwbid = {BWBRO005289},
```



```

    valid = {2024-01-01},
}

```

`\newsourcetypealias` Declareert één bron in het document zelf, de route die helemaal geen bestand nodig heeft. Hij komt in dezelfde opslag terecht als een regel uit een bestand, dus verderop is het verschil niet te zien.

`\newsourcetypealias` De types en velden hebben Engelse namen, zoals de rest van deze bundel, en luisteren allemaal ook naar een Nederlandse: regeling en euhandeling voor de types, en citeertitel, afkorting, boek, geldig, soort, nummer, roepnaam, titel, pb en lidwoord voor de velden. Een bestand mag beide gebruiken, en een document dat ze terugleest ook: de Nederlandse naam wordt naar de Engelse herleid voordat er iets opgeslagen wordt. Nog een taal is een regel per naam.

`\newsourcetype` Declareert een type, de velden waar een entry van dat type niet zonder kan, en het register waarin zijn verwijzingen verwoord worden. Er zijn er twee gedeclareerd: `{\required fields}` regulation, dat een citetitle nodig heeft, en euact, dat een kind en een number nodig heeft. Een bron die er één mist stopt de draai, en dat is precies wat alles verderop toestaat een veld te lezen zonder te vragen of het er is.

`\srcfield` Lezen wat er gedeclareerd is. Een waarde is data en geen $\TeX$: hij komt terug zoals hij in het bestand staat.

`\srctype` Een aanhaling van regelgeving is een aanhaling van een *versie* ervan: hetzelfde artikel een jaar later gelezen is niet per se hetzelfde artikel, en de Juriconnect-standaard draagt de datum daarom in de verwijzing zelf.

`\ifsourceexists` Wat deze uitgave daar precies mee doet, zodat niets hier voor meer wordt gelezen dan het is. Zij **legt vast** per wanneer een document aanhaalt, met `\sourcedate` voor het hele document of met `date=\langle JJJJ-MM-DD \rangle` voor één aanhaling, wat ook naar datum luistert. Zij **waarschuwt** wanneer een document geen van beide zegt, en wanneer een document spreekt per een latere datum dan het veld `valid` van een bron die het aanhaalt. Zij drukt die datum **niet** af in een aanhaling, en zij bouwt **geen** Juriconnect-URI en geen link daarnaartoe. Een veld `bwbid` of `celex` wordt opgeslagen en is met `\srcfield` terug te lezen; niets in deze bundel maakt daar een verwijzing van. Kortom: regulatory 1.0 heeft de datumdiscipline die een Juriconnect-aanhaling vraagt en geen Juriconnect-uitvoer.

Een document dat geen van beide zegt krijgt dat eenmalig te horen, als waarschuwing en niet als fout: aanhalen zonder datum is gebrekkig, maar niet stuk, en een pakket dat niet aan een bestaand document toegevoegd kan worden zonder het te stoppen is een pakket dat niet toegevoegd wordt.

Iedere entry zegt wanneer hij voor het laatst naast de tekst gelegd is die hij beschrijft, in een veld `valid`. Spreekt het document per een latere datum, dan kan de entry niet voor zichzelf instaan en zegt dat, één keer per bron. Een entry die het veld weglaat krijgt dat te horen, want hij zou dan nooit als verouderd gemeld kunnen worden: “nog niet bekeken” en “bekeken en actueel” zouden dezelfde stilte zijn, en een wacht waarvan de uitschakelaar een ontbrekend veld is wordt per ongeluk uitgezet in plaats van

met opzet. Dat besluit heeft een eigen woord, `valid = unverified`, en dat blijft stil. Dat is wat een gedeeld bronbestand überhaupt veilig maakt om te gebruiken, waar het ook bijgehouden wordt: de identifiers erin verouderen nooit, de datums wel, en een document dat de gegevens voorbijloopt die het gekregen heeft wordt dat verteld in plaats van stilletjes bediend met wat de laatste wijziging aan dat bestand toevallig bevatte. Deze bundel levert zelf geen regelgeving mee en is daar ook niet de plek voor — een citeertitel verandert op zijn eigen klok en een pakket op CTAN niet — maar hij is wel de plek voor het gereedschap dat zegt wanneer die twee uit elkaar gelopen zijn.

`\srcref` Haalt een bron aan, in zijn geheel of tot op een van zijn bepalingen. Het optionele argument `[<provision>]` bevat de bepaling — `article`, `paragraph`, `point` en `subpoint`, die ook naar `artikel`, `{<key>}` `lid`, `onderdeel` en `onder luisteren` — en staat vóór de sleutel, zoals bij `\cite`, zodat een blokhaak in de zin ná een verwijzing er niet voor wordt aangezien.

```
\srcref[article=96, paragraph=2, point=c]{bw6}
    artikel 96, tweede lid, onderdeel c, van Boek 6 van het Burgerlijk Wetboek
\srcref[article=3.126, paragraph=1, point={c,d}]{wetib}
    artikel 3.126, eerste lid, onderdelen c en d, van de Wet inkomstenbelasting 2001
\srcref[article={162--164}]{bw6}
    artikelen 162 tot en met 164, van Boek 6 van het Burgerlijk Wetboek
\srcref*[article={162--164}]{bw6}
    art. 6:162-164 BW
\srcref*[article=96, paragraph=2, point=c]{bw6}
    art. 6:96 lid 2 onder c BW
\srcref{avg}
    de Algemene verordening gegevensbescherming
\srcref[article=6, paragraph=1, point=a]{avg}
    artikel 6, lid 1, punt a), van de Algemene verordening gegevensbescherming
\srcref[article=6, paragraph=1, point=a]{avgen}
    point (a) of Article 6(1) of the General Data Protection Regulation
```

De sleutels zijn in iedere regel dezelfde en de bewoording niet: wat verandert is de bron, en daarmee het register waarin hij wordt aangehaald. De laatste twee zijn dezelfde bepaling van dezelfde verordening, ingevoerd als `avg` met het Nederlandse register en als `avgen` met het Engelse — beide staan in codelijst 12. De Nederlandse sleutelnamen `artikel`, `lid`, `onderdeel` en `onder` zijn aliassen van deze en leveren woord voor woord dezelfde aanhaling op, wat paragraaf 13.2 terugleest. Twee dingen bepalen hoe dat eruit ziet, en geen van beide is de taal van het document.

Het eerste is het *register*, en dat hoort bij de bron: een verwijzing wordt verwoord zoals de rechtsorde van het instrument hem verwoordt. Een Nederlands contract dat een verordening van de Unie aanhaalt schrijft “lid 1, punt a)”, omdat die verordening dat zelf zo schrijft — geteld in de Nederlandse tekst van de Algemene verordening gegevensbescherming: punt a) zesendertig keer, onderdeel a) en sub a) geen enkele keer. Het register komt van het type van de entry en is per bron te overschrijven met een veld `register`.

Het tweede is het *systeem*: voluit of verkort. De lopende tekst van een document is voluit; een voetnoot is verkort, en een verwijzing die in de lopende tekst tussen haakjes

staat ook. De optie `sourcestyle` zet welke van de twee een document gebruikt, met `full` of `short`, en de ster vraagt om de andere. De voluit-vorm is de vorm die de Aanwijzingen voor de regelgeving voor regelingen zelf voorschrijven — hier een model en geen voorschrift, want die aanwijzingen binden ministeries en geen contracten — en de verkorte is die van de Leidraad voor juridische auteurs.

`\newsourcedeterminer` Zegt welk lidwoord een soort instrument in een register krijgt. Het is geen eigenschap
`{<register>}` van het register: in het Frans komt “de la directive” zestien keer voor tegenover “du règlement” zeven, dus één standaard zou vaker fout dan goed zijn, en “du directive” is geen
`{<kind>}` kwestie van stijl maar van grammatica. Het veld `determiner` van een entry overschrijft
`{<article>}` het voor het geval dat geen tabel kan kennen.

`\newsourceregister` Declareert een register: een stel sleutels voor de voluit-vorm en hetzelfde stel voor de
`{<name>}` verkorte. Een register is geen taal: hetzelfde document kan in meer dan één aanhalen,
`{<full>}` en doet dat zodra het naast een Nederlandse wet een verordening van de Unie noemt.
`{<short>}` Er worden er vijf meegeleverd, elk in het definitiebestand van de taal waar hij bij hoort en onder de sleutel waarmee hij hier genoemd wordt:

`dutch` een Nederlands nationaal instrument, aangehaald in het Nederlands. Voluit is het de vorm die de Aanwijzingen voor de regelgeving voor regelingen zelf voorschrijven; verkort die van de Leidraad voor juridische auteurs. Het is het register dat het type `regulation` krijgt wanneer een entry er geen noemt. Gedeclareerd in `regulatory-dutch.def`.

`dutch_eu` een handeling van de Unie, aangehaald in het Nederlands, wat een eigen register is en geen variant van het eerste. Het is het register dat het type `euact` standaard krijgt. Gedeclareerd in `regulatory-dutch.def`.

`german_eu` een handeling van de Unie, aangehaald in het Duits. Gedeclareerd in `regulatory-german.def`.

`french_eu` een handeling van de Unie, aangehaald in het Frans. Gedeclareerd in `regulatory-french.def`.

`english_eu` een handeling van de Unie, aangehaald in het Engels, en de enige van de vijf die andersom is opgebouwd. Gedeclareerd in `regulatory-english.def`.

Dat is de hele verzameling, en de suite legt die lijst naast de code: de registers die een draai declareert worden teruggelezen en met deze vijf vergeleken, zodat er één toevoegen, hernoemen of kwijtraken een test laat falen in plaats van alleen deze bladzijde. Twee van de vijf krijgt een document zonder erom te vragen, want het zijn de standaardwaarden van de twee meegeleverde types; om de andere drie wordt gevraagd met een veld `register` op de entry. Er is geen nationaal register voor het Engels, het Duits of het Frans — zie de laatste alinea van dit onderdeel.

De drie van de Unie zijn elk gemeten in de tekst van dezelfde verordening in die taal, en ze verschillen waar het uitmaakt: het Duits schrijft geen komma's en noemt het geletterde onderdeel een *Buchstabe*, het Frans schrijft de komma's en het sluithaakje maar noemt het een *point*. Bij die twee kort de ster niet de bewoording in maar grijpt hij

naar de abbreviation van het instrument, want die verordening kort geen enkel woord van zichzelf af: Abs. en par. komen er geen enkele keer in voor, tegenover Artikel vijfhonderdeen.

Een benaming laat zijn nummer nooit los: de verordening zet er een harde spatie tussen en een gewone tussen de onderdelen. In het Frans wordt paragrafe tweehonderdzesentachtig keer door een harde spatie gevolgd en vijf keer door een gewone; in het Nederlands artikel tweehonderdachtennegentig keer tegenover geen enkele.

Die Nederlandse telling komt uit diezelfde verordening, en een verordening van de Unie is geschreven in het register van de Unie: zij schrijft tweehonderdvijfentachtig keer lid 6 en geen enkele keer *zesde lid*. Wat naar het nationale register overdraagt is artikel 6, want daar staat de benaming in beide registers vóór het nummer. Waar het nationale register het nummer vooropzet, *zesde lid*, is niet gemeten wat de twee bindt, en schrijft deze bundel daar een gewone spatie.

Een verwijzing mag meer dan één bepaling noemen. Een komma maakt er een opsomming van en -- een reeks, en het register bepaalt hoe beide luiden: het draagt de voegwoorden en het meervoud van de benaming. Allebei zijn ze gemeten in dezelfde verordening, in elk van haar talen — “artikelen 101 en 102”, “artikelen 12 tot en met 15”, “punten c) en e)”, “Buchstaben c und e”, “paragraphes 1, 2 et 4” — en het verre eind van een reeks wordt anders geschreven dan het nabije, want het wetboek wordt aangehaald als “art. 6:162-164”, met zijn boek één keer genoemd.

Er is geen nationaal Duits of Frans register, en dat is met opzet: het Duitse federale recht schrijft § 5 Abs. 2 Nr. 3 BGB, een ander bouwsel en geen vertaling van dit, en er is hier niemand die de uitkomst kan toetsen. De registers zelf staan niet in de implementatie maar in het taaldefinitiebestand van hun taal, afgedrukt in paragraaf 10. Dat is één meting op één plek: wat zegt dat het Duits “Artikel 6 Absatz 1 Buchstabe a” schrijft in een aanhaling, is dezelfde lezing van dezelfde tekst die zegt dat het een lid een *Absatz* noemt. Het scheelt bovendien een tweede stelsel van namen: het register van het Nederlands heet dutch en de taal ook.

10 Taalondersteuning

Dit pakket begon in het Nederlands. Het Engels kwam als tweede, en de twee verwoorden een verwijzing zo verschillend — “artikel 2, tweede lid, onderdeel b” tegenover “Article 2(2), point (b)” — dat er een stel beslissingen van moest worden in plaats van een stel woorden. Dat is wat een derde taal mogelijk maakt, en het is ook waarom een taalbestand meer is dan een vertaling.

Er worden vier talen meegeleverd, en ze zijn niet even ver. Wat een taalbestand bevat zijn drie losse dingen, en het Duits en het Frans hebben er twee van de drie:

	Benamingen	Interne verwijzingen	Citeerregister
Nederlands	volledig	volledig	dutch, dutch_eu
Engels	volledig	volledig	english_eu
Duits	volledig	niet gemeten	german_eu
Frans	volledig	niet gemeten	french_eu

Het Nederlands en het Engels zijn compleet. Iedere benaming, ieder verwijsformaat en beide citeerregisters van het Nederlands zijn gemeten, en de bewoording van veertig aanhalingen wordt door de suite teruggelezen (paragraaf 13.2).

Het Duits en het Frans zijn ten dele gedekt, en het deel dat ontbreekt is zichtbaar. Wat deze bundel ervan heeft is gemeten in de Duitse en de Franse tekst van de Algemene verordening gegevensbescherming: de woorden waarmee een bepaling wordt aangeduid, en hoe een handeling van de Unie in die taal wordt aangehaald. Wat hij er niet van heeft is de verwoording van een *interne* verwijzing — de `\rref@setup` hieronder — die niet gemeten is, en die wordt overgelaten aan wie de uitkomst kan toetsen in plaats van dat ernaar geraden wordt. Dus:

- `\article` en `\para` dragen Duitse of Franse koppen, en `\gls` en `\printdefs` zijn in die taal benoemd.
- `\srcref` verwoordt een aanhaling van een handeling van de Unie correct, in het register van die taal. Een *nationaal* Duits of Frans instrument heeft hier helemaal geen register; zie hieronder.
- `\rref`, `\nref` en `\aref` vallen terug op de Engelse *opbouw* met Duitse of Franse *woorden*. Een Duits document leest daarom “Artikel 2(2), Buchstabe (b)”, wat Duitse woordenschat in een Engels bouwsel is. Het is zichtbaar fout in plaats van onzichtbaar fout, en er wordt niet voor gewaarschuwd, want de woorden staan er.
- Een verwijzing naar een *ander document* is in geen van beide talen te verwoorden, en dáár wordt wel voor gewaarschuwd. Het Duits verbuigt het woord dat een voegwoord regeert — “der Verordnung” maar “des Vertrags” — en het Frans versmelt het voorzetsel met een lidwoord dat het geslacht bepaalt, dus geen van beide kan de wending uit een onderwerp samenstellen. Beide declareren hun voegwoord leeg, en `\documentlabel` zegt dat dan één keer per artefact en vraagt om een `ref label` (paragraaf 7). Er een geven is de oplossing, en dat is een regel per document.

- Er is geen nationaal Duits of Frans citeerregister, met opzet: het Duitse federale recht schrijft § 5 Abs. 2 Nr. 3 BGB, een ander bouwsel en geen vertaling van dit. Een aanhaling van zo'n instrument met het type `regulation` wordt verwoord in het Nederlandse nationale register, want dat is de standaardwaarde van dat type, en dat is fout. Zolang zo'n register er niet is, zeg dat dan met een veld `register` in plaats van de standaardwaarde te laten staan.

Een taal waar deze bundel helemaal geen bestand voor heeft is een vijfde geval, en het mildste: alles valt terug op het Engels, en het document krijgt dat één keer te horen aan het begin, met de taal erbij. Er faalt in dat geval niets — de verwijzingen worden in het Engels verwoord, en dat is ook de enige manier waarop het opvalt.

Ondersteund betekent dus Nederlands en Engels. Het Duits en het Frans zijn bruikbaar voor wat er gemeten is en zijn niet dezelfde belofte, en deze handleiding zegt van iedere taal wat ze is in plaats van er vier te tellen.

`\RegulatoryLoadLanguage` Iedere ondersteunde taal heeft een eigen definitiebestand `regulatory-⟨taal⟩.def`, vergelijkbaar met de `fc-⟨taal⟩.def` bestanden van `fmtcount`. Aan het eind van de preamble wordt het definitiebestand geladen van iedere taal die `babel` of `polyglossia` geladen heeft, samen met `regulatory-english.def`, dat de standaardwaarden bevat waar iedere andere taal op voortbouwt. Een taal zonder definitiebestand valt terug op het Engels. Documenten die hun taal op een andere manier kiezen kunnen een definitiebestand expliciet laden met `\RegulatoryLoadLanguage{⟨taal⟩}`. Ondersteuning voor een nieuwe taal toevoegen komt neer op het schrijven van zo'n bestand; de vier meegeleverde staan afgedrukt in paragrafen 10.1 t/m 10.4.

`\rref@setup` `\rref@setup` declareert hoe `⟨taal⟩` een interne verwijzing verwoordt, en stelt zijn drie niveaus in één keer in. Het is **extension API**: hij is bedoeld om aangeroepen te worden, `{⟨lang⟩}` hij is bedoeld om vanuit een taaldefinitiebestand aangeroepen te worden, en hij wordt `{⟨article opts...⟩}` `{⟨para opts...⟩}` zoals hij is ondersteund voor de hele 1.x-reeks. `{⟨point opts...⟩}` Het apenstaartje in de naam is geen vergissing en geen voornemen tot hernoemen. Een definitiebestand wordt gelezen met het apenstaartje als letter — `\RegulatoryLoadLanguage` zet de categoriecode en herstelt hem daarna — en dat is dezelfde conventie waaronder de `.def` bestanden van de kernel zelf en de `fc-⟨taal⟩.def` bestanden van `fmtcount` geschreven zijn. Binnen zo'n bestand vraagt de naam nergens om. Een document dat een taal in zijn eigen preamble declareert zet de aanroep tussen `\makeatletter` en `\makeatother`, wat een apenstaartje overal in een preamble vraagt.

De andere drie argumenten nemen de sleutels hieronder, één stel per niveau:

`name` de benaming in kleine letters: 'artikel', 'lid', 'onderdeel'. De standaardwaarde van ieder niveau is de vertaling van zijn eigen naam, `artikel \GetTranslation{article}`, `lid \GetTranslation{paragraph}` en `onderdeel \GetTranslation{point}`.

`Name` dezelfde benaming met een beginhoofdletter, en de standaardwaarde is dezelfde vertaling met er een.

- `names, Names` de benaming van een verwijzing die meer dan één structuur noemt: “artikelen 1 en 2”, “onderdelen c en d”. Het is een eigen sleutel omdat niet de taal bepaalt of het meervoud gebruikt wordt maar de plaats van de benaming: het Nederlands schrijft “onderdelen c en d” maar houdt “het tweede en derde lid” enkelvoudig, omdat de benaming daar achter de rangtelwoorden staat.
- `ref format` een macro met één argument, het nummer zelf. Het Nederlands schrijft een lid als rangtelwoord, waarvoor `\ordinalstringnum` gebruikt is, en een onderdeel als letter. Een formaat dat een woord oplevert moet zowel `\Nref` als `\nref` bedienen, en daar is `\@ifrrefcap` voor: `\@ifrrefcap{\Ordinalstringnum{...}}{\ordinalstringnum{...}} ←`
- `label format` een macro met twee argumenten — de name en het resultaat van `ref format` — die de volgorde van die twee bepaalt. Een Nederlands artikel is `{#1 #2}`, een Nederlands lid `{#2 #1}`, en een Engels lid `\@gobble{#1}#2`⁶. De waarde is een macro en niet de tekst zelf, dus de eerste daarvan wordt meegegeven als `\mylabelformat`, gedefinieerd als: `\newcommand\mylabelformat[2]{#1 #2}`.
- `group conjunction` wat dit niveau aan het niveau erboven koppelt. In “artikel 1, eerste lid” heeft het artikel een `group conjunction` van `,~`.
- `group format` een macro met één argument, dat de niveaus boven dit niveau bevat. Een taal die ze apart zet — `[Article 1(6),] point (a)` — geeft het onderdeel `\rref@group@braced` mee, wat deze bundel voor het Engels deed tot de Europese stijlguides die vraag anders beslechtten.

Iedere sleutel kan per niveau verschillen. Lees bij het schrijven van een nieuwe taal eerst `regulatory-english.def`: dat geeft iedere waarde voluit op, en is de configuratie waar een taal zonder eigen configuratie op terugvalt.

Hieronder volgen de definitiebestanden van de vier meegeleverde talen. Ze vormen de bron waar een vertaling mee begint, dus staan ze hier en niet tussen de modules van paragraaf 14. De toelichting erin is, net als de broncode, in het Engels gesteld.

Elk van hen zegt in eigen woorden wat het bevat en wat niet, en daar komt de tabel aan het begin van dit onderdeel vandaan.

10.1 Engelse definities

English is the language every other one falls back on, so its definitions are stated in full rather than left to the initial values. Translating this bundle into another language comes down to copying this file to `regulatory-⟨language⟩.def` and translating the right hand side of every `\DeclareTranslation`; see paragraaf 10.2 for what a language that words its references differently needs on top of that.

⁶`\@gobble` neemt zijn argument en drukt niets af, en zo verdwijnt de benaming waar het nummer alleen staat.

A word on the third level. It is a *point*, the lettered subdivision of a paragraph, and not a subparagraph: in the terminology of the European institutions a subparagraph is the *alinea*, the unnumbered block of text, which this bundle does not number at all. The key used to be subparagraph and was renamed for that reason; a document that asked for that translation by name asks for point now.

```
\ProvidesRegulatoryLanguage{english}
[2026/09/10 1.0.0 English definitions for regulatory]

\DeclareTranslation{english}{article}{article}
\DeclareTranslation{english}{Article}{Article}
\DeclareTranslation{english}{articles}{articles}
\DeclareTranslation{english}{Articles}{Articles}
\DeclareTranslation{english}{paragraph}{paragraph}
\DeclareTranslation{english}{Paragraph}{Paragraph}
\DeclareTranslation{english}{paragraphs}{paragraphs}
\DeclareTranslation{english}{Paragraphs}{Paragraphs}
\DeclareTranslation{english}{point}{point}
\DeclareTranslation{english}{Point}{Point}
\DeclareTranslation{english}{points}{points}
\DeclareTranslation{english}{Points}{Points}
\DeclareTranslation{english}{to}{to}
\DeclareTranslation{english}{Definitions}{Definitions}
\DeclareTranslation{english}{of the}{of the}
```

The reference formats are only set up when regulatory-ref is loaded, since a document that only loads regulatory-struct has the translations above but none of the reference macros.

English compresses a reference the way the European style guides do: *Article 6(2), point (a)*. The article carries its name and its number, the paragraph follows in brackets with nothing in between, and the point is lettered and bracketed in turn, behind a comma. A group of points repeats neither the name nor the brackets of what came before: *Article 1(6), point (a), (b) and (d)*.

```
\regulatory@ifmodule{ref}{%
  \rref@setup{english}{
    group conjunction={ }
  }{
    group conjunction={,~}
  }{
    ref format=\rref@refformat@alphaparen,
    label format=\rref@label@prefix,
    group conjunction={,~},
    group format=\rref@refformat@noop
  }%
}
```

10.1.1 Engels citeerregister

How an act of the Union is cited in English, and the one register of the five that is arranged the other way round. Measured in the English text of the General Data Protection

Regulation, the same document the other registers were measured in: “point (a) of Article 6(1)” twenty-eight times, against nothing at all for “Article 6(1)(a)” and nothing for “Article 6(1), point (a)”, which are the two arrangements the continental registers use. The deepest level comes first, each is joined to the next with “of”, and the paragraph is written into the article between brackets rather than named — except where it stands without one, “point (a) of paragraph 1”, five times, which is why the paragraph keeps a word of its own as well. More than one paragraph on the same article keeps the brackets around each of them, “Article 6(1), (2) and (3)”, and by itself does not: “paragraphs 1 and 2”, twenty-one times.

A run is joined with “to” at every level — “Articles 12 to 15” fifteen times, “points (a) to (c)” twice, “paragraphs 1 to 3” once — and a list with “and”: “Articles 15 and 16”, “paragraphs 1, 2 and 3”. The designation stands before the run and takes its plural, as it does before a list. The instrument hangs on the end with the same “of” that joins the levels, so this register needs no separate word for it.

The non-breaking space between a designation and its number is a decision and not a measurement here: the English text writes it both ways, a hundred and sixty-seven against two hundred and twenty-seven. It is written non-breaking, as in the other registers, since nothing in the source speaks against it.

There is no measured shortened form. The English text of a regulation has no footnotes to shorten a citation in, and this bundle does not invent one: the shortened system is the same arrangement with the name of the instrument abbreviated, which is what `german_eu` and `french_eu` do for the same reason.

English gives a regulation no article and a treaty one: “of Regulation (EU) 2016/679” against “of the Treaty”, both measured in the same text. So the default of the register is nothing, and the two kinds that do take one say so.

```
\regulatory@ifmodule{sources}{%
\newsourceregister{english_eu}{
  articleone = Article,
  articlemany = Articles,
  articlenum = \regulatory@src@number@plain,
  paragraph = \regulatory@src@before,
  paragraphone = paragraph,
  paragraphmany = paragraphs,
  paragraphnum = \regulatory@src@num@plain,
  attachnum = \regulatory@src@num@parens,
  attach = \regulatory@src@attach@plain,
  point = \regulatory@src@before,
  pointone = point,
  pointmany = points,
  pointnum = \regulatory@src@num@parens,
  subpoint = \regulatory@src@before,
  subpointone = point,
  subpointmany = points,
  subpointnum = \regulatory@src@num@unmeasured,
  and = {-and-},
  to = {-to-},
  separator = {\space of\space},
```

```

assemble = \regulatory@src@assemble@backward,
determiner = {},
connective = {},
source = \regulatory@src@source@full@store
} {
  articleone = Article,
  articlemany = Articles,
  articlenum = \regulatory@src@number@plain,
  paragraph = \regulatory@src@before,
  paragraphone = paragraph,
  paragraphmany = paragraphs,
  paragraphnum = \regulatory@src@num@plain,
  attachnum = \regulatory@src@num@parens,
  attach = \regulatory@src@attach@plain,
  point = \regulatory@src@before,
  pointone = point,
  pointmany = points,
  pointnum = \regulatory@src@num@parens,
  subpoint = \regulatory@src@before,
  subpointone = point,
  subpointmany = points,
  subpointnum = \regulatory@src@num@unmeasured,
  and = {-and-},
  to = {-to-},
  separator = {\space of\space},
  assemble = \regulatory@src@assemble@backward,
  determiner = {},
  connective = {},
  source = \regulatory@src@source@short@store
}

```

English gives a regulation no article and a treaty one: “of Regulation (EU) 2016/679” against “of the Treaty”, both measured in the same text. So the default of the register is nothing, and the two kinds that do take one say so.

```

\newsourcedeterminer{english_eu}{treaty}{the}
\newsourcedeterminer{english_eu}{charter}{the}
} {}

```

10.2 Nederlandse definities

Dutch names an article ‘artikel’, a paragraph ‘lid’ and a point ‘onderdeel’. The translations dictionary translates ‘and’ and ‘see’ already, but its ‘to’ means ‘aan’, while a range of articles asks for ‘tot’.

```

\ProvidesRegulatoryLanguage{dutch}
[2026/09/10 1.0.0 Dutch definitions for regulatory]

\DeclareTranslation{dutch}{article}{artikel}
\DeclareTranslation{dutch}{Article}{Artikel}
\DeclareTranslation{dutch}{articles}{artikelen}

```

```

\DeclareTranslation{dutch}{Articles}{Artikelen}
\DeclareTranslation{dutch}{paragraph}{lid}
\DeclareTranslation{dutch}{Paragraph}{Lid}
\DeclareTranslation{dutch}{paragraphs}{leden}
\DeclareTranslation{dutch}{Paragraphs}{Leden}
\DeclareTranslation{dutch}{point}{onderdeel}
\DeclareTranslation{dutch}{Point}{Onderdeel}
\DeclareTranslation{dutch}{points}{onderdelen}
\DeclareTranslation{dutch}{Points}{Onderdelen}
\DeclareTranslation{dutch}{to}{tot}
\DeclareTranslation{dutch}{Definitions}{Definities}
% Composes, unlike German: a Dutch noun keeps its shape after `van'. It does assume a de-
word, so an
% artifact whose subject is a het-word gives a `ref label' of its own.
\DeclareTranslation{dutch}{of the}{van de}

```

Where English writes ‘Article 2(2), point (b)’, Dutch writes ‘artikel 2, tweede lid, onderdeel b’. So a paragraph is written as an ordinal followed by its designation, a point is lettered with its designation in front of it and without brackets, and all parts are joined with a comma.

An enumeration takes the plural of the designation, ‘de artikelen 162 tot en met 164’ and ‘onderdelen c en d’. A paragraph is the exception: its designation follows the ordinals rather than preceding them, and Dutch keeps it singular there, ‘het tweede en derde lid’. The Leidraad shows no such enumeration written out in full, so this is the idiom rather than a rule, which is why it is set here and not among the initial values.

```

\regulatory@ifmodule{ref}{%
  \rref@setup{dutch}{
    group conjunction={,~}
  }{
    ref format=\rref@refformat@ordinal,
    label format=\rref@label@postfix,
    names=\GetTranslation{paragraph},
    Names=\GetTranslation{Paragraph},
    group conjunction={,~},
    group format=\rref@refformat@noop
  }{
    ref format=\rref@refformat@alpha,
    label format=\rref@label@prefix,
    group conjunction={,~},
    group format=\rref@refformat@noop
  }%
}{}

```

10.2.1 Nederlandse citeerregisters

How a source is cited in Dutch, which is the same knowledge as the wording above and measured in the same texts, so it stands in the same file. Two registers, because Dutch serves two legal orders. `dutch` is how a Dutch instrument is cited: in full the form the Aanwijzingen voor de regelgeving lay down for regulations themselves, and shortened

the form of the Leidraad voor juridische auteurs. `dutch_eu` is how an act of the Union is cited in Dutch, which is a register of its own and not a variant of the first: the regulation writes lid 1, punt a) where the national register writes eerste lid, onderdeel a.

```
\regulatory@ifmodule{sources}{%
\newsourceregister{dutch}{
  articleone = artikel,
  articlemany = artikelen,
  articlenum = \regulatory@src@number@plain,
  paragraph = \regulatory@src@after,
  paragraphone = lid,
  paragraphmany = lid,
  paragraphnum = \regulatory@src@num@ordinal,
  point = \regulatory@src@before,
  pointone = onderdeel,
  pointmany = onderdelen,
  pointnum = \regulatory@src@num@plain,
  subpoint = \regulatory@src@before,
  subpointone = onder,
  subpointmany = onder,
  subpointnum = \regulatory@src@num@degree,
  and = {-en-},
  to = {-tot-en-met-},
  separator = {,~},
  determiner = de,
  connective = {van-},
  source = \regulatory@src@source@full@store
}{
  articleone = art.,
  articlemany = art.,
  articlenum = \regulatory@src@number@colon,
  paragraph = \regulatory@src@before,
  paragraphone = lid,
  paragraphmany = lid,
  paragraphnum = \regulatory@src@num@plain,
  point = \regulatory@src@before,
  pointone = onder,
  pointmany = onder,
  pointnum = \regulatory@src@num@plain,
  subpoint = \regulatory@src@before,
  subpointone = onder,
  subpointmany = onder,
  subpointnum = \regulatory@src@num@degree,
  and = {-en-},
  to = {-},
  separator = {-},
  determiner = de,
  connective = {},
  source = \regulatory@src@source@short@store
}
```

```

\newsourceregister{dutch_eu}{
    articleone = artikel,
    articlemany = artikelen,
    articlenum = \regulatory@src@number@plain,
    paragraph = \regulatory@src@before,
    paragraphone = lid,
    paragraphmany = leden,
    paragraphnum = \regulatory@src@num@plain,
    point = \regulatory@src@before,
    pointone = punt,
    pointmany = punten,
    pointnum = \regulatory@src@num@paren,
    subpoint = \regulatory@src@before,
    subpointone = onder,
    subpointmany = onder,
    subpointnum = \regulatory@src@num@unmeasured,
    and = {-en-},
    to = {-tot-en-met-},
    separator = {,~},
    determiner = de,
    connective = {van~},
    source = \regulatory@src@source@full@store
}
{
    articleone = art.,
    articlemany = art.,
    articlenum = \regulatory@src@number@plain,
    paragraph = \regulatory@src@before,
    paragraphone = lid,
    paragraphmany = leden,
    paragraphnum = \regulatory@src@num@plain,
    point = \regulatory@src@before,
    pointone = punt,
    pointmany = punten,
    pointnum = \regulatory@src@num@paren,
    subpoint = \regulatory@src@before,
    subpointone = onder,
    subpointmany = onder,
    subpointnum = \regulatory@src@num@unmeasured,
    and = {-en-},
    to = {-tot-en-met-},
    separator = {~},
    determiner = de,
    connective = {},
    source = \regulatory@src@source@short@store
}

\newsourcedeterminer{dutch}{verordening}{de}
\newsourcedeterminer{dutch}{richtlijn}{de}
\newsourcedeterminer{dutch}{wet}{de}

```

```

\newsourcedeterminer{dutch}{wetboek}{het}
\newsourcedeterminer{dutch}{besluit}{het}
\newsourcedeterminer{dutch}{verdrag}{het}
\newsourcedeterminer{dutch_eu}{verordening}{de}
\newsourcedeterminer{dutch_eu}{richtlijn}{de}
\newsourcedeterminer{dutch_eu}{besluit}{het}
\newsourcedeterminer{dutch_eu}{verdrag}{het}
}{}

```

10.3 Duitse definities

This file covers part of what a language file holds, and says which part. What is here was measured in the German text of the General Data Protection Regulation: the words a provision is called by, and the register an act of the Union is cited in. What is not here is how German words an *internal* reference — the `\rref@setup` of paragraaf 10 — which was not measured, and which is left to whoever can check the result rather than guessed at. A German document therefore gets German designations in the English arrangement, which is visible and wrong rather than invisible and wrong.

The connective before the name of a document is a different matter, and German is the language that decides it for all of them. It is a genitive, and the genitive reshapes the word it governs: counted in the same text, “der Verordnung” sixteen times and “der Richtlinie” twenty-nine, but “des Vertrags” and “des Beschlusses” — never *des Vertrag*. The ending is on the noun, so knowing the article would not be enough, and no rule over a connective and a subject in the nominative can produce the phrase. The connective is therefore declared empty here, which makes `\documentlabel` ask for a `ref label` instead of wording it wrongly in silence.

```

\ProvidesRegulatoryLanguage{german}
[2026/09/10 1.0.0 German definitions for regulatory]

\DeclareTranslation{german}{article}{Artikel}
\DeclareTranslation{german}{Article}{Artikel}
\DeclareTranslation{german}{articles}{Artikel}
\DeclareTranslation{german}{Articles}{Artikel}
\DeclareTranslation{german}{paragraph}{Absatz}
\DeclareTranslation{german}{Paragraph}{Absatz}
\DeclareTranslation{german}{paragraphs}{Absätze}
\DeclareTranslation{german}{Paragraphs}{Absätze}
\DeclareTranslation{german}{point}{Buchstabe}
\DeclareTranslation{german}{Point}{Buchstabe}
\DeclareTranslation{german}{points}{Buchstaben}
\DeclareTranslation{german}{Points}{Buchstaben}
\DeclareTranslation{german}{to}{bis}
\DeclareTranslation{german}{Definitions}{Begriffsbestimmungen}
\DeclareTranslation{german}{of the}{}

```

10.3.1 Duits citeerregister

How an act of the Union is cited in German: Artikel 6 Absatz 1 Buchstabe a, without commas between the parts and with the genitive before the name of the instrument.

There is no register for German federal law here. That law writes § 5 Abs. 2 Nr. 3 BGB, which is another construction and not a translation of this one, and it needs the local counterparts of the sources this bundle leans on. Whoever has those can write it in this file; what is missing is a convention of a legal order, not a handful of words.

```
\regulatory@ifmodule{sources}{%
\newsourceregister{german_eu}{
  articleone = Artikel,
  articlemany = Artikel,
  articlenum = \regulatory@src@number@plain,
  paragraph = \regulatory@src@before,
  paragraphone = Absatz,
  paragraphmany = Absätze,
  paragraphnum = \regulatory@src@num@plain,
  point = \regulatory@src@before,
  pointone = Buchstabe,
  pointmany = Buchstaben,
  pointnum = \regulatory@src@num@plain,
  subpoint = \regulatory@src@before,
  subpointone = Nummer,
  subpointmany = Nummer,
  subpointnum = \regulatory@src@num@unmeasured,
  and = {~und~},
  to = {~bis~},
  separator = {~},
  determiner = der,
  connective = {},
  source = \regulatory@src@source@full@store
}{
  articleone = Artikel,
  articlemany = Artikel,
  articlenum = \regulatory@src@number@plain,
  paragraph = \regulatory@src@before,
  paragraphone = Absatz,
  paragraphmany = Absätze,
  paragraphnum = \regulatory@src@num@plain,
  point = \regulatory@src@before,
  pointone = Buchstabe,
  pointmany = Buchstaben,
  pointnum = \regulatory@src@num@plain,
  subpoint = \regulatory@src@before,
  subpointone = Nummer,
  subpointmany = Nummer,
  subpointnum = \regulatory@src@num@unmeasured,
  and = {~und~},
  to = {~bis~},
```

```

separator = {~},
determiner = der,
connective = {},
source = \regulatory@src@source@short@store
}

\newssourcedeterminer{german_eu}{verordnung}{der}
\newssourcedeterminer{german_eu}{richtlinie}{der}
\newssourcedeterminer{german_eu}{empfehlung}{der}
\newssourcedeterminer{german_eu}{beschluss}{des}
\newssourcedeterminer{german_eu}{vertrag}{des}
}{}

```

10.4 Franse definities

Part of a language file, for the same reason as paragraaf 10.3: the designations and the citation register were measured in the French text of the General Data Protection Regulation, and the wording of an internal reference was not.

The connective is declared empty here as well, though for a milder reason than the German one. French does not reshape the noun, but it fuses the preposition with the article and picks by gender — “de la directive” against “du règlement”, counted sixteen against seven in that one text — so a single default is wrong more often than it is right. A document that refers to another one gives it a `ref label`, which is what the German case forces anyway.

```

\ProvidesRegulatoryLanguage{french}
[2026/09/10 1.0.0 French definitions for regulatory]

\DeclareTranslation{french}{article}{article}
\DeclareTranslation{french}{Article}{Article}
\DeclareTranslation{french}{articles}{articles}
\DeclareTranslation{french}{Articles}{Articles}
\DeclareTranslation{french}{paragraph}{paragraphe}
\DeclareTranslation{french}{Paragraph}{Paragraphe}
\DeclareTranslation{french}{paragraphs}{paragraphes}
\DeclareTranslation{french}{Paragraphs}{Paragraphes}
\DeclareTranslation{french}{point}{point}
\DeclareTranslation{french}{Point}{Point}
\DeclareTranslation{french}{points}{points}
\DeclareTranslation{french}{Points}{Points}
\DeclareTranslation{french}{to}{à}
\DeclareTranslation{french}{Definitions}{Définitions}
\DeclareTranslation{french}{of the}{}

```

10.4.1 Frans citeerregister

How an act of the Union is cited in French, measured in the French text of the same regulation: article 6, paragraphe 1, point a), with the commas the German register leaves out and the closing bracket the Dutch one also writes.

There is no register for French national law here, for the same reason there is none for German.

```
\regulatory@ifmodule{sources}{%
\newsourceregister{french_eu}{
  articleone = article,
  articlemany = articles,
  articlenum = \regulatory@src@number@plain,
  paragraph = \regulatory@src@before,
  paragraphone = paragraphe,
  paragraphmany = paragraphes,
  paragraphnum = \regulatory@src@num@plain,
  point = \regulatory@src@before,
  pointone = point,
  pointmany = points,
  pointnum = \regulatory@src@num@paren,
  subpoint = \regulatory@src@before,
  subpointone = point,
  subpointmany = points,
  subpointnum = \regulatory@src@num@unmeasured,
  and = {\~et~},
  to = {\~à~},
  separator = {,~},
  determiner = du,
  connective = {},
  source = \regulatory@src@source@full@store
}{
  articleone = article,
  articlemany = articles,
  articlenum = \regulatory@src@number@plain,
  paragraph = \regulatory@src@before,
  paragraphone = paragraphe,
  paragraphmany = paragraphes,
  paragraphnum = \regulatory@src@num@plain,
  point = \regulatory@src@before,
  pointone = point,
  pointmany = points,
  pointnum = \regulatory@src@num@paren,
  subpoint = \regulatory@src@before,
  subpointone = point,
  subpointmany = points,
  subpointnum = \regulatory@src@num@unmeasured,
  and = {\~et~},
  to = {\~à~},
  separator = {,~},
  determiner = du,
  connective = {},
  source = \regulatory@src@source@short@store
}
```

```
\newsourcedeterminer{french_eu}{règlement}{du}  
\newsourcedeterminer{french_eu}{traité}{du}  
\newsourcedeterminer{french_eu}{directive}{de~la}  
\newsourcedeterminer{french_eu}{décision}{de~la}  
\newsourcedeterminer{french_eu}{recommandation}{de~la}  
{}
```

11 Markdown

De `regulatory-md` module vertaalt de onderdelen van een Markdown-bron naar de structuren uit paragraaf 4. Deze wordt geladen zodra `markdown` dat is, ongeacht de volgorde van beide pakketten, en de pakketoptie `md` laadt `markdown` zelf. Die optie is de veilige weg, aangezien `markdown` vóór `enumitem` geladen moet worden om de lijsten van dit pakket heel te laten. Een Markdown-bron is daarmee beperkt tot de onderdelen die hier beschreven staan; hoofdstukken en de overige onderdelen van een documentklasse hebben geen tegenhanger. Kijk naar codelijst 8 voor een Markdown-voorbeeld en naar codelijst 7 hoe zo'n Markdown-bron uiteindelijk gebruikt kan worden vanuit $\text{\LaTeX}$ (paragraaf 13.4).

Een Markdown-bron bevat geen $\text{\TeX}$. De module werd geladen met de optie `hybrid` van `markdown`, die iedere backslash doorliet, en de voorbeelden van deze bundel schreven hun labels en verwijzingen op die manier; `markdown` heeft die optie afgeschaft. Zonder haar wordt een backslash afgedrukt in plaats van uitgevoerd, en een bron die er nog één bevat verliest zijn verwijzingen zonder dat er ook maar iets misgaat, dus een document dat uit een oudere bron komt is het waard één keer doorgelezen te worden. Alles wat die optie nodig had wordt hieronder in Markdown geschreven.

De Markdown-listings hieronder staan in beide handleidingen in het Nederlands, en dat is met opzet: het zijn de constructies uit `example.md`, een testfixture van deze bundel die door de suite wordt teruggelezen (paragraaf 13.2). Niets in dit onderdeel hangt af van de taal van de tekst.

11.1 Labels en verwijzingen

Een attribuut schrijft een label en een link leest er één. Een kop en een bracketed span nemen allebei een identifier tussen accolades, die het label wordt van het artikel of van het lid waarin hij staat:

```
# Quisque ullamcorper {#art:lorem}

1. [Lorem ipsum dolor sit amet.]{#lid:lorem}
2. [Sed do eiusmod tempor incididunt.]{#lid:lorem2}
```

Een verwijzing is een link waarvan het doel met `#` begint, en de drie vormen die een Markdown link kan aannemen zijn de drie verwijzingen uit paragraaf 6:

`<#label>` noemt de structuur waar hij naar wijst, zoals `\Aref` doet, en neemt meerdere labels tegelijk: `<#lid:a, lid:b>`.

`[](#label)` het nummer alleen, zoals `\rref` doet.

`[tekst](#label)` behoudt de tekst en maakt er een link van.

Naar een label van een ander document wordt op dezelfde manier verwezen, met de prefix die `\refdocument` eraan gaf: `<#ex2-lid:lorem>`.

Een bracketed span mag een verwijzing van de eerste vorm bevatten, maar geen tweede span en geen link met blokhaken; `markdown` laat de hele constructie dan staan zoals hij is. Voor een lid dat zowel een label draagt als een definitie aanhaalt is er de identifier van paragraaf 11.2.

11.2 Een eigen identifier

Een bracketed span mag geen tweede span bevatten en geen link met blokhaken, wat een beperking van Markdown zelf is en een ongelukkige voor een lid dat zowel een label draagt als een definitie aanhaalt. Deze bundel voegt daarom één regel toe aan de grammatica van de lezer, waarmee de identifier op zichzelf staat, vóór de tekst die hij labelt:

1. `{#lid:lorem}` Een [persoonsgegevens](#pg) wordt verwerkt, zie `<#lid:eerder>`.

Alleen `{#` opent zo'n identifier, dus een accolade elders in de tekst blijft ongemoeid. De regel staat in `regulatory-syntax.lua`, dat markdown zelf inleest, en dat bij de implementatie is opgenomen in paragraaf 14: het is Lua en geen $\text{\TeX}$, aangezien een renderer alleen bepaalt hoe een constructie die de lezer al kent gezet wordt, en deze eerst herkend moet worden.

11.3 Definities in Markdown

Een definitielijst declareert definities. De term draagt het label als bracketed span en de inhoud van het item wordt de beschrijving, wat hetzelfde is als wat `\newdefinition` doet (paragraaf 5):

```
[Onderhandse akte]{#onderhands}
```

: Een akte die zonder tussenkomst van een ambtenaar is opgemaakt.

De andere manier is het YAML blok waarmee een Markdown-bron mag beginnen, waar een definitie een record met benoemde velden is in plaats van een onderdeel van de tekst. Dat is de steviger van de twee, aangezien niets in de lopende tekst hem kan verstoren, en de aangewezen manier wanneer de definities los van de tekst worden bijgehouden:

```
---
definitions:
  - label: onderhands
    name: Onderhandse akte
    description: Een akte die zonder tussenkomst van een ambtenaar is opgemaakt.
---
```

Geen van beide drukt iets af op de plek waar hij staat. Waar de lijst verschijnt wordt bepaald door `\printdefs`, of vanuit de bron door een divisie, die de twee argumenten van dat commando als attributen neemt en de lijst na zijn eigen inhoud afdrukt:

```
::: {.definitions style=description widest="Onderhandse akte"}
```

De volgende begrippen worden in deze overeenkomst gebruikt.

...

Een divisie zonder inhoud wordt niet herkend door `markdown`, dus er hoort een regel tekst in. Wanneer er definities gedeclareerd worden en de lijst nooit wordt afgedrukt, zegt de module dat aan het eind van de draai.

Een definitie wordt aangehaald zoals naar al het andere verwezen wordt, met een link: `<#onderhands>` drukt de naam van de entry af en `[een onderhandse akte](#onderhands)` behoudt de formulering van de zin, beide met een link naar de definitielijst.

`\autocitedefs` Markeert iedere plek waar de naam van een definitie in de tekst daarna voorkomt, zonder dat de tekst dat hoeft aan te geven. Het is een bewuste keuze, en er valt iets af te wegen: de vergelijking is letterlijk, dus hoofdlettergevoelig, blind voor verbuiging — “persoonsgegevens” is niet “persoonsgegeven” — en hij ziet een term niet die door een regeleinde in tweeën is gebroken. Bovendien werkt hij alleen op wat na de aanroep omgezet wordt, dus de definities horen in een eigen bron, die als eerste omgezet wordt.

```
\markdownInput{definities.md}
\autocitedefs
\markdownInput{overeenkomst.md}
```

12 HTML

Dit pakket is voor PDF geschreven, zoals paragraaf 1 zegt, maar een regulatorisch document dat op een website terechtkomt is een gangbare genoeg wens. Die omzetting is het werk van `tex4ht`, aangestuurd met `make4ht`, dat een pakket configureert via een bestand `(pakket).4ht`. `tex4ht` levert er voor dit pakket geen, dus levert dit pakket zijn eigen mee als `tex/regulatory.4ht`, naast de pakketbestanden. Een document dat `regulatory` kan vinden, vindt die configuratie er meteen bij. Eén ding moet een document zelf doen: `\DocumentMetadata` *niet* declareren in een HTML-bouw. Die activeert de PDF-administratie waar paragraaf 1 om vraagt, wat de lijstcode van $\LaTeX$ binnenhaalt en `tex4ht` niets laat om een paragraaf onderdeel in om te zetten: gemeten aan codelijst 5 twaalf lijstelementen zonder die declaratie en geen enkele met. `tex4ht` definieert `\HCode` voordat het het document leest, en daar toetsen de voorbeelden op — zie codelijst 2.

Allebei worden ze hardop gezegd in plaats van aan de uitkomst overgelaten. Een bouw die omzet met actieve PDF-administratie krijgt te horen dat de lijstmaak verdwijnt, en een bouw waarin de configuratie van dit pakket niet gevonden is krijgt te horen dat een artikel en een lid lopende tekst worden zonder kop. Geen van beide stopt de omzetting, en geen van beide valt op in een uitkomst die je niet al weet te lezen: een kop die geen kop is ziet eruit als een vetgedrukte zin.

Of de omzetting zonder zo'n bestand überhaupt loopt hangt af van één regel in de preambule. De koppen van dit pakket lopen via `titlesec` zolang `\ParseLaTeXeHeading` niet beschikbaar is, en `tex4ht` vervangt juist de kopmachinerie waar `titlesec` zich aan hecht, waardoor die nergens meer op aan kan haken. Gemeten met `make4ht`:

met `\DocumentMetadata` is de nieuwe kopinterface beschikbaar, wordt `titlesec` in het geheel niet geladen en loopt de omzetting zonder ook maar één fout af.

zonder wordt `titlesec` geladen en stopt `make4ht` op `Package titlesec Error: No format for this command`.

paragraaf 1 vraagt al om `\DocumentMetadata` omwille van de bijlagen, dus een document dat iets bijvoegt staat aan de goede kant van die streep tegen de tijd dat het bij `make4ht` komt. Een document dat niets bijvoegt heeft geen reden die declaratie te dragen.

Een omzetting die afloopt is niet hetzelfde als een omzetting die iets betekent, en wat hier wegvalt meldt zich niet. codelijst 5 declareert `\DocumentMetadata` en zet in beide gevallen zonder fout om, wat niets zegt over wat eruit komt:

	zonder <code>regulatory.4ht</code>	met <code>regulatory.4ht</code>
<code>koppen <h<n>></code>	0	5
<code><section></code>	0	5
<code>ankers id='article.<n>'</code>	4	4

De ankers staan er in beide gevallen, want `hyperref` schrijft ze. Het is de structuur die wegvalt: zonder het configuratiebestand is een artikel een paragraaf met een vette passage erin, en voor een document waarvan de structuur de betekenis is is dat het verschil

tussen een tekst waar een lezer doorheen kan en een tekst die er alleen goed uitziet.

Het bestand is dus in beide gevallen nodig, om twee verschillende redenen: zonder \DocumentMetadata houdt het de omzetting overeind, en mét die regel is het het enige dat de structuur oplevert. Aangezien hij met het pakket meereist, is beide geregeld zodra regulatory geladen wordt.

13 Tests

De tests van dit pakket staan in de map `test` en beantwoorden twee verschillende vragen. `make test` zet ieder voorbeeld met iedere engine die dit pakket ondersteunt, te weten `lualatex` en `pdflatex`, en vraagt niets meer dan een $\text{\TeX}$ installatie. `make conformance` zet diezelfde documenten één keer per PDF-standaard en laat `veraPDF` er een oordeel over vellen, waarvoor Docker nodig is; de uitkomst daarvan wordt bewaard in het pakket zelf als paragraaf 13.3, aangezien de validator niet overal beschikbaar is waar dit pakket gebouwd wordt. Om diezelfde reden zit die niet in `make test`, en voegt hij zich daarbij met `make test WITH_CONFORMANCE=1`.

Iedere suite bouwt in een eigen map onder `test/out`: `out/lualatex` en `out/pdflatex` voor de engines, `out/html` voor de omzetting uit paragraaf 12 en `out/conformance` voor de standaarden. Ze overschrijven elkaar daardoor niet, en wat een suite opleverde blijft achteraf staan, wat de HTML van een draai tot iets maakt dat je kunt openen en bekijken. Dat ze niets delen is ook wat ze tegelijk laat draaien, wat `make test` doet; ieder van hen is los op te vragen als `make suite-lualatex`, `make suite-html` enzovoort. De bronnen worden in zo'n map gekopieerd in plaats van via `TEXINPUTS` gevonden: `\markdownInput` opent zijn bestand met Lua, dat `kpathsea` niet raadpleegt, dus dat bestand moet er echt staan.

Dat er een document uitkomt is de kleinste helft van wat een draai vaststelt. Vrijwel alles wat dit pakket verkeerd kan doen komt er evengoed als document uit — een verwijzing die in het verkeerde register geschreven is, een definitie die stilzwijgend niet aankwam, een label dat nergens heen wees. Iedere suite eindigt daarom in een reeks controles die de draai naleest, beschreven in paragraaf 13.2, en twee van de documenten staan er om te falen.

13.1 De testdocumenten

Er staan tien documenten in die map. Twee ervan bestaan in beide talen en één wordt omgezet in plaats van gezet, want waar het voor is wordt alleen onder `tex4ht` gelezen, zodat een suite elf bestanden per engine zet en er vier omzet. Ieder daarvan is een driver die de taal instelt, de artefacten declareert en de definities inlaadt, en die de inhoud overlaat aan een eigen bestand, zodat beide taalversies één tekst delen:

`example1-nl.tex` en `example1-en.tex`, die beide `example1.tex` lezen en de definities van `example1.bib` inladen.

`example2-nl.tex` en `example2-en.tex`, die op dezelfde manier `example2.tex` en `example2.bib` lezen. Ze voegen met `\newdefinition` een eigen definitie toe en zetten de lijst met de stijl `description`, waarmee beide routes uit paragraaf 5 die geen tweede programma nodig hebben gedekt zijn.

`md-example.tex` dat in plaats daarvan `example.md` leest met `\markdownInput`, en de definities van `example1.bib` inlaadt. Deze bestaat alleen in het Nederlands.

`sign-example.tex` dat de handtekeningvelden van paragraaf 8 plaatst en helemaal geen definities heeft. Het wordt ongecomprimeerd geschreven, zodat de

annotatie over een veld zonder PDF-gereedschap uit het PDF-bestand terug te lezen is. Het omkadert elk veld zelf: de module tekent een schrijflijn, en het kader laat op papier zien tot waar het vlak reikt dat een viewer aanbiedt.

`index-example.tex` dat de tweede definitieroute van paragraaf 5 neemt: `bib2gls=false`, zodat `\loadglsdefs` het bestand `index-defs.tex` met `\newglossaryentry` leest en `makeindex` het sorteren doet. Het is het enige document dat een ander programma dan `bib2gls` tussen de draaien nodig heeft, en tot het geschreven werd had die route geen enkele dekking.

`attach-example.tex` het enige document dat om `attachmentLink=annotation` vraagt (paragraaf 7.1), zodat die waarde er zonder dit document ongetest uit zou gaan. Het is met opzet getagd en ongecomprimeerd geschreven, zodat zowel de annotatie als haar plek in de structuurboom zonder PDF-gereedschap uit het PDF-bestand te lezen zijn.

`nolang-example.tex` dat helemaal geen taal noemt en omgezet wordt in plaats van gezet. Ieder ander document hier laadt `babel`, en een bijgehouden taal is wat `tracklang` een `glossaries-⟨taal⟩.ldf` laat laden en daarmee translator. Alles in `regulatory.4ht` dat naar een commando van dat pakket grijpt werkt daarom in ieder ander document en is in dit ene ongedefinieerd, en zo heeft een aanroep van `\providetranslation` in dat bestand het overleefd.

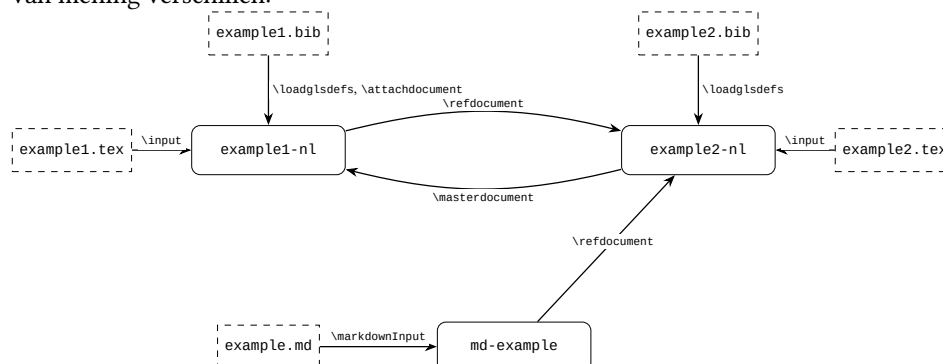
`date-example.tex` dat aanhaalt zonder te zeggen per wanneer, en daarna een bron aanhaalt waarvan de gegevens nagelopen zijn vóór de datum die het opgeeft. Het is het tweede van de twee documenten die horen te waarschuwen.

`source-example.tex` dat `sources.bib` inlaadt en aanhaalt wat daarin staat, in ieder register van paragraaf 9. Het declareert daarnaast twee bronnen in het document zelf, zodat beide routes de lezer in gedekt zijn.

`de-example.tex` het enige document dat bedoeld is om te *waarschuwen*. Het verwijst in het Duits naar twee andere documenten, waarvan er één zegt hoe hij genoemd moet worden en de ander niet, en het Duits kan dat niet uit een onderwerp alleen verwoorden. Het valt daarom buiten de controle die het pakket geen enkele waarschuwing toestaat, en heeft een eigen controle.

Er worden er nog twee gezet die juist moeten falen: `badsources.tex` laadt een bib-bestand waarin een verplicht veld ontbreekt, en `badsyntax.tex` één met een regel die de lezer niet aanvaardt. Het zijn geen voorbeelden van iets. Een lezer die overslaat wat hij niet lezen kan, laat een aanhaling leeg in een document dat er verder goed uitziet, dus de garantie die hij geeft is alleen zoveel waard als zijn weigering — en dat is wat deze twee meten. Wat de eerste drie tot een test maakt in plaats van drie losse documenten is de manier waarop ze naar elkaar wijzen, waar paragraaf 7 over gaat. `example1-nl` noemt `example2-`

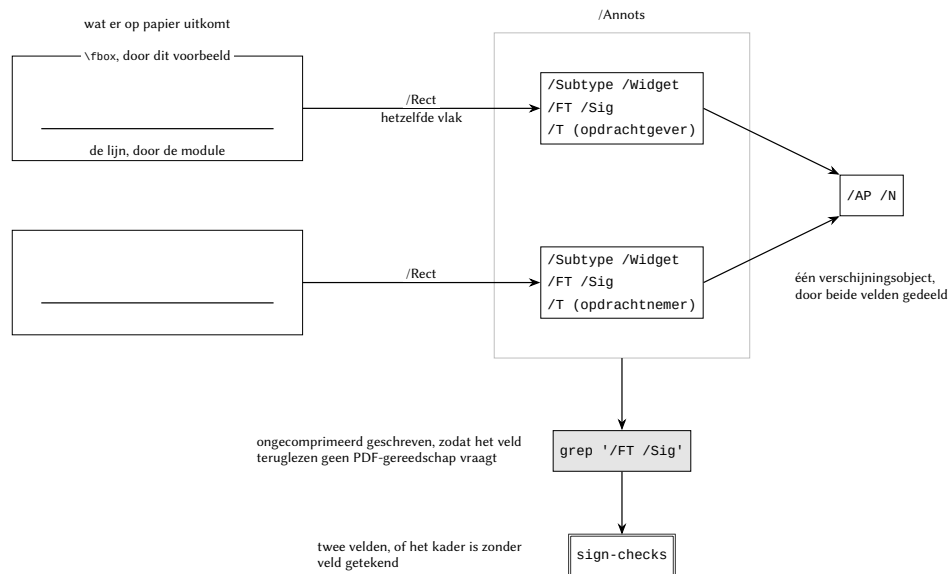
nl met `\refdocument`, zodat het naar diens artikelen kan verwijzen zonder het bij te voegen. `example2-nl` noemt `example1-nl` juist met `\masterdocument`, wat daarnaast diens definities inlaadt en het document zelf als bijlage hecht. Samen dekken ze daarmee een verwijzing in beide richtingen, en het paar wordt nog eens gedeclareerd door deze handleiding, en daarom kan de opsomming van paragraaf 13.4 naar hun artikelen verwijzen en hun PDF-bestanden insluiten. `example1-nl` hecht tenslotte met `\attachdocument` een bestand dat helemaal geen PDF is, het geval waar de PDF-standaarden het meest over van mening verschillen:



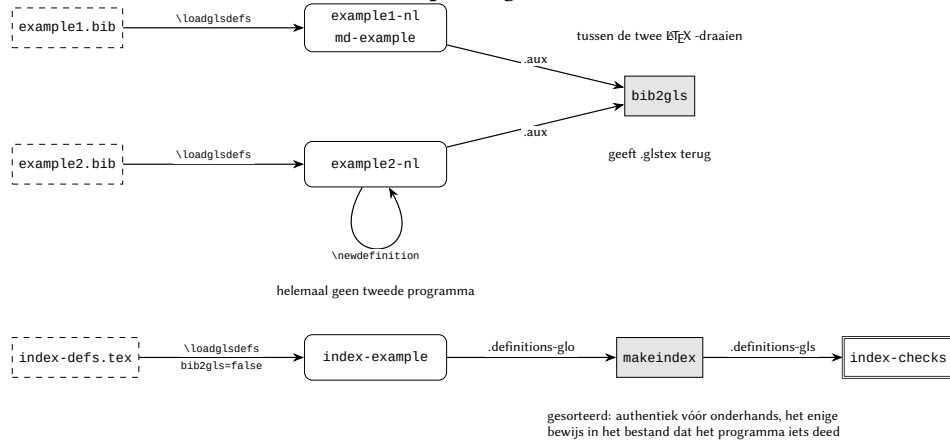
Een dichte kader is een document dat een PDF-bestand wordt, een gestreepte een bestand dat het inleest. Een pijl tussen twee documenten is een declaratie in de preamble van het document waar hij vandaan komt. `md-example` laadt daarnaast `example1.bib` in, wat omwille van de leesbaarheid niet getekend is.

De andere zes zijn ook getekend, elk op het ene waar hij voor staat. Ze houden de vormen van de tekening hierboven aan en voegen er drie toe: een grijze kader is een programma dat tussen twee $\text{\LaTeX}$ -draaien loopt, een kader met scherpe hoeken iets dat letterlijk uitgeschreven staat — sleutels in een document, een object in het PDF-bestand — en een dubbele kader de controle uit paragraaf 13.2 die de uitkomst naleest.

`sign-example` plaatst twee van de velden uit paragraaf 8. Wat het tekent en wat het draagt lopen uit elkaar zonder dat er iets verkeerd uit ziet: het kader en de lijn worden door $\text{\TeX}$ gezet en staan hoe dan ook op de pagina, terwijl de annotatie die het vlak tot een veld maakt op papier onzichtbaar is en pas blijkt wanneer iemand probeert te tekenen. Beide velden wijzen naar hetzelfde verschijningsobject, en daar is een lege verschijning voor — de lijn wordt door de pagina getekend, dus een verschijning die er een tekende zou hem dubbel tekenen.

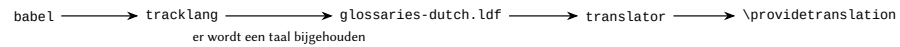


De wegen van paragraaf 5 verschillen in wat er tussen de twee $\text{\LaTeX}$ -draaien moet lopen, en dat is wat de documenten hieronder uit elkaar houdt. `index-example` is de enige op de tweede weg en de enige die om een ander programma dan `bib2gls` vraagt. Zijn controle leest niet de lijst maar de volgorde: het document declareert *Onderhandse* eerst en *Authentieke* daarna, dus een bestand dat andersom terugkomt is het enige bewijs in de hele draai dat `makeindex` überhaupt iets gedaan heeft.

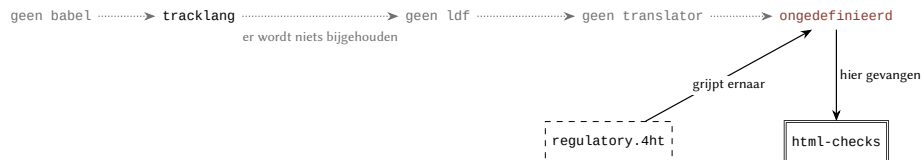


Een bijgehouden taal is wat `tracklang` een `glossaries-(taal).ldf` laat laden, en dat bestand is wat translator meebrengt. Ieder ander document hier laadt `babel`, dus een commando van dat pakket werkt in allemaal en is ongedefinieerd in het ene dat helemaal geen taal noemt, en zo heeft een aanroep van `\providetranslation` het overleefd in `regulatory.4ht`. De kopjes die dit document wél krijgt zijn Engels: de taalbestanden die deze bundel meelevert worden geladen wat het document ook spreekt.

ieder ander document

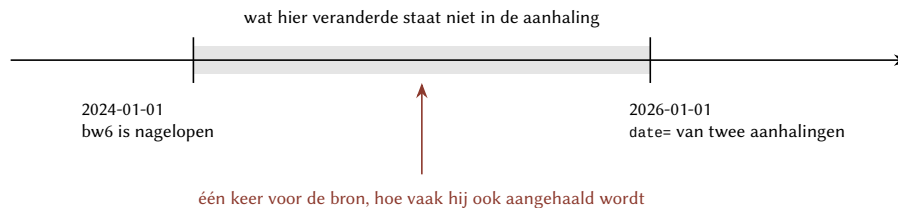


nolang-example



dat bestand wordt alleen onder tex4ht gelezen, dus
dit document wordt omgezet en nooit gezet

date-example hoort vier keer te waarschuwen, en zijn controle telt er van elk één en verder niets. Eén gaat over een bron die nagelopen is vóór de dag per welke dit document spreekt, en die wordt één keer gezegd hoe vaak die bron ook aangehaald wordt; één over een document dat nergens zegt per wanneer het aanhaalt; en twee over entries die helemaal niet naast een datum te leggen zijn. De vierde entry zegt dat met zoveel woorden en blijft stil, en daar is deze tekening voor: een wacht waarvan de uitschakelaar een ontbrekend veld is wordt per ongeluk uitgezet in plaats van met opzet.

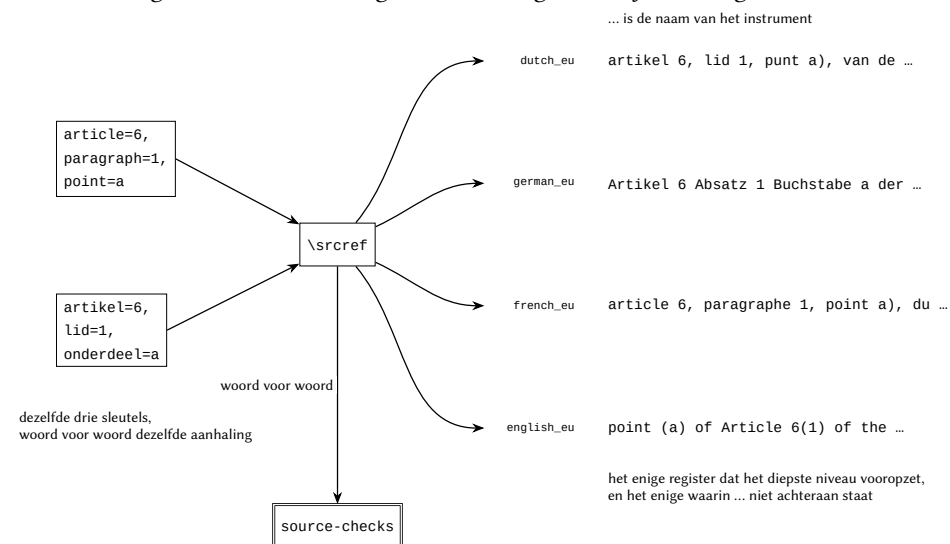


en wat het document zelf schrijft

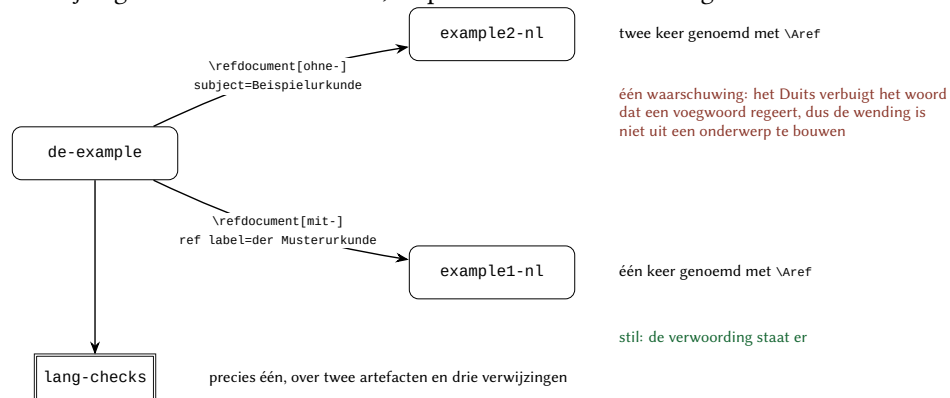
\srcref	geen date=, geen \sourcedate	het zegt nergens per wanneer het aanhaalt
zonder	geen veld valid	kan nooit verouderd gemeld worden
onbekeken	valid = unverified	een besluit, en besluiten zijn stil
scheef	valid = 1 januari 2024	noch JJJJ-MM-DD noch unverified
date-checks	vier waarschuwingen, van elk één, en verder niets	

Het register hoort bij de bron en niet bij de zin waar hij in staat, dus dezelfde drie sleutels komen er als vier verschillende aanhalingen uit. Het Engelse is het enige register dat het diepste niveau vooropzet, en het enige waarin de naam van het instrument niet op de

bepaling volgt. De Nederlandse sleutelnamen zijn aliases van de Engelse en horen dus woord voor woord dezelfde aanhaling op te leveren; verschilt er iets, dan is de alias niet wat hij zegt te zijn. `source-example` wordt zo voor veertig aanhalingen nagelezen, want een aanhaling in het verkeerde register faalt nergens — hij staat er gewoon verkeerd.



Het Duits verbuigt het woord dat een voegwoord regeert — *des Vertrags, des Beschlusses* — dus geen regel bouwt de wending uit een onderwerp in de nominatief. `de-example` noemt twee documenten, waarvan er één een eigen verwoording draagt en de ander niet, en verwijst twee keer naar degene zonder. Dat is wat één getal genoeg maakt: drie verwijzingen over twee artefacten, en precies één waarschuwing.



13.2 Wat de suite naleest

Iedere engine-suite eindigt in tien reeksen controles, die elk opschrijven wat ze geteld hebben en de bouw stoppen zodra het niet klopt. Ze zijn andersom geschreven dan een test die een uitkomst nakijkt: ieder van hen bestaat omdat er ooit iets faalde zonder te falen, en zijn taak is om diezelfde stilte de volgende keer hoorbaar te maken. Alle tien

zijn expres gebroken om te zien dat ze bijten — een controle die nooit gefaald heeft, heeft nog niets vastgesteld.

- `lang-checks` eist precies één waarschuwing over de twee documenten die `de-example` noemt, en verwijst naar één daarvan twee keer. Dat ene getal draagt drie beweringen tegelijk: degene zonder eigen verwoording zegt dat, degene met een verwoording niet, en wie het zegt zegt het één keer hoe vaak hij ook genoemd wordt. Een waarschuwing per verwijzing zou zijn eigen log verzuipen.
- `ref-checks` telt de verwijzingen die nergens heen wezen en de labels die twee keer gedefinieerd werden, over ieder document en niet alleen over dat waaraan gewerkt wordt: een label met een spatie erin wees nergens heen terwijl ieder document dat toen in de suite zat labels zonder spatie gebruikte.
- `log-checks` staat het pakket geen enkele waarschuwing toe. Wat dit pakket zegt over een document waarover het niets te klagen heeft is een defect, en een waarschuwing die niemand leest is een waarschuwing die niet geschreven is.
- `attach-checks` leest de annotatie van `attachmentLink=annotation` uit het PDF-bestand: dat hij geplaatst en op de pagina geregistreerd is, dat hij het bestand en een verschijning draagt, dat er geen `go-to-link` naast overleefd heeft, en dat hij in een `Annot-structuurelement` zit zoals de verwijzing ernaast in een `Link-element` zit. Dat laatste paar wordt hier gecontroleerd en niet aan veraPDF overgelaten, want de validator waarop deze suite gepind is meldt het ontbrekende element niet en een oudere deed dat wel: het object is het bewijs en het oordeel niet.
- `md-checks` leest het tussenbestand dat de Markdown-omzetting schrijft en eist dat er geen rauwe `TEX` in overleefde, en dat de syntaxuitbreiding van paragraaf 11.2 gevonden is. Een ontbrekende uitbreiding laat een identifier als tekst staan, wat leest als een typefout en niet als een kapotte bouw.
- `sign-checks` telt de handtekeningannotaties in het PDF-bestand zelf. Het kader onder een veld wordt door `TEX` getekend en komt er hoe dan ook uit; wat het tot een veld maakt is op papier onzichtbaar en blijkt pas wanneer iemand probeert te tekenen.
- `source-checks` leest terug wat de lezer van bib-bestanden gelezen heeft, veld voor veld, en de formulering van veertig aanhalingen woord voor woord. Een aanhaling in het verkeerde register faalt nergens; hij staat er gewoon verkeerd. Datzelfde doel zet daarna de twee documenten die moeten falen en stopt zodra een van beide toch slaagde.
- `index-checks` leest terug dat de geïndexeerde route genomen is, dat beide definities in het bestand geschreven zijn dat `makeindex` leest, en dat ze in een andere volgorde terugkomen dan waarin ze gedeclareerd zijn — het enige

bewijs in het resultaat dat het programma tussen de draaien iets gedaan heeft. De woordenlijstbestanden worden vóór de draai weggegooid: gemeten, met de `makeglossaries`-stap er expres uit slaagden de controles nog steeds, omdat het gesorteerde bestand van de vorige bouw er nog lag.

`date-checks` leest de twee dingen terug die een document mis kan hebben met de datum waarop een aanhaling spreekt: er niets over zeggen, en de gegevens van de aangehaalde bron voorbijlopen. De tweede wordt geteld en niet alleen gevonden, want hij hoort één keer per bron te komen en het voorbeeld haalt dezelfde bron twee keer aan.

`html-checks` telt wat de omzetting van paragraaf 12 moet overleven: secties, koppen, leden, definities uit beide routes en de twee lijststijlen. Die verdwijnen zonder foutmelding, want een verloren configuratie levert platte tekst op en geen fout. Hij leest ook terug dat het document dat geen taal noemt er nog steeds geen noemt, aan de woorden die het gebruikt: dat document is alleen het omzetten waard zolang het taallos is, en zijn structuur zou een toegevoegde taal overleven.

13.3 PDF-conformiteit

Een regulatorisch document wordt doorgaans gearhiveerd, dus is het de moeite waard te weten welke PDF-standaarden het kan dragen. De documenten hierboven worden voor ieder geval hieronder nog een keer gezet, waarbij er niets verandert behalve een `\DocumentMetadata` die op de commandline vóór het document gedeclareerd wordt, en `veraPDF` velt het oordeel. De kolom `expected` is wat dit pakket claimt: `pass` moet valideren, `xfail` faalt op een defect dat hieronder beschreven wordt, en `fail` is een geval dat juist *moet* falen — zonder zo'n geval kan het harnas niet aantonen dat het überhaupt iets meet.

Geval	Document	Standaard	expected	Uitkomst
a2b	md-example	2b	pass	PASS
a4	md-example	4	pass	PASS
a4f-ua2	example2-nl	4f	pass	PASS
a4f-ua2	example2-nl	ua2	pass	PASS
a3a-ua1	example2-nl	3a	pass	PASS
a3a-ua1	example2-nl	ua1	pass	PASS
a2a-ua1-tagged	md-example	2a	fail	FAIL (6.8-5)
a2a-ua1-tagged	md-example	ua1	pass	PASS
a2a-ua1-nocss	md-example	2a	pass	PASS
a2a-ua1-nocss	md-example	ua1	pass	PASS
a4-tagged	md-example	4	fail	FAIL (6.9-3)
a4-nocss	md-example	4	pass	PASS
a3b	example1-nl	3b	xfail	FAIL (6.8-4)
a2b-embedded-nonpdf	example1-nl	2b	fail	FAIL (6.8-5)
a4f-embedded-nonpdf	example1-nl	4f	pass	PASS
a4f-attachannot	example1-nl	4f	pass	PASS
a4f-ua2-attachannot	example2-nl	4f	pass	PASS
a4f-ua2-attachannot	example2-nl	ua2	pass	PASS
a2b-signed	sign-example	2b	pass	PASS

Gemeten met veraPDF 1.30.2, met de validator gepind op digest⁷.

Eén claim die een document geschreven met dit pakket niet kan maken. **PDF/A-1** verbiedt ingesloten bestanden categorisch, dus ieder document dat iets bijvoegt valt af.

PDF/A-2a en **PDF/A-4** stonden hier als tweede, en daar horen ze niet. Taggen hangt een tweetal HTML-bestanden als associated files aan de catalog, en beide standaarden eisen dat ieder ingesloten bestand zelf een PDF/A is — maar die twee bestanden zijn de hele hindernis, en tagpdf heeft een sleutel die ze weglaat: `\tagpdfsetup{attach-css=false}`. De tabel draagt beide helften als twee paren gevallen die in niets anders verschillen, op een document dat zelf niets bijvoegt: zonder de sleutel faalt PDF/A-2a op rule 6.8-5 en PDF/A-4 op rule 6.9-3, mét hem slagen ze allebei, en **PDF/UA-1** slaagt hoe dan ook. Wat je inlevert is waar die twee bestanden voor zijn — opmaakhints voor lijsten en voor de uitlijning van MathML — en niet de structuur waarop een lezer wordt voorgelezen.

Het verschil tussen de profielen is op dezelfde manier gemeten: hetzelfde document met dezelfde bijlage staat twee keer in de tabel, waar het PDF/A-2b faalt op regel 6.8-5 en PDF/A-4f haalt. Alles wat geen PDF is bepaalt dus het profiel, en dat is de moeite waard te weten vóórdat een document uitgegeven wordt in plaats van erna.

Eén geval staat op `xfail`. Het faalt op regel 6.8-4 van ISO 19005-3, op een bestandsspecificatie voor het externe document: die draagt `/AFRelationship /Unspecified`, heeft geen `/EF` en staat niet in `/AF`. Zij wordt geschreven voor de hyperlink naar dat document, door de PDF-administratie van $\LaTeX$ en niet door dit pakket, dus PDF/A-3 blijft buiten be-

⁷verapdf/cli@sha256:d5ee329657cf9bc4b2400392dd54c7d0a0ce9980ff6fa2da5590eebeec007cdb

reik voor een document dat naar de bepalingen van een ander verwijst, tot dat verderop opgelost is. Paragraaf 7 zegt dat waar een lezer de functie tegenkomt.

Het tweede document haalde PDF/A-3a op het moment dat de aanhaling van een definitie die elders staat geen hyperlink naar dat document meer werd. Die link wees naar een anker dat niet in dit bestand staat, dus hij ging sowieso nergens heen; hem in-trekken nam de bestandsspecificatie mee. Gemeten aan de twee documenten zoals ze er staan: `example1-nl` draagt elf `/GoToR`-acties en die ene specificatie zonder `/EF`; `example2-nl` draagt geen enkele `/GoToR`, en zijn ene specificatie voor het document dat het bijvoegt heeft wél een `/EF` en `/AFRelationship /Source`.

13.4 De testbestanden

De bronnen van de voorbeelden die de suite bouwt, hier afgedrukt en als PDF-bestand aan deze handleiding gehecht. Het is één stel fixtures en beide handleidingen drukken hetzelfde stel af, dus die met een eigen tekst staan hoe dan ook in het Nederlands; `example1-en.tex` en `example2-en.tex` zijn de Engelse drivers over dezelfde inhoud.

```
1 \begin{center}
2   \Huge \translation{Example One}{Voorbeeld Één}
3 \end{center}
4 \article{Definitiones}\label{art:defs}
5 \printdefs[labeling]{Nam dui}
6 \article{Quisque ullamcorper}\label{art:lorem}
7 \begin{paras}
8   \item \label{lid:lorem} \textfill
9   \item \label{lid:lorem2} \textfill
10  \begin{paras}
11    \item \label{sub:lorem} \textfill~\Aref{lid:lorem}.
12    \item \label{sub:lorem2} \textfill~\Aref{ex2-lid:lorem}.
13    \item \label{sub:lorem3} \textfill~\Aref{lid:lorem,lid:lorem2}.
14    \item \label{sub:lorem4} \textfill~\Aref{ex2-lid:lorem,ex2-lid:lorem2,ex2-lid:↵
      lorem3}.
15    \item \label{sub:lorem5} \textfill~\Aref{ex2-lid:lorem,ex2-lid:lorem2,ex2-lid:↵
      lorem3,ex2-lid:lorem5}.
16    \item \label{sub:lorem6} \textfill~\Gls{def1} ipsum dolor sit amed.
17    \item \label{sub:lorem7} \textfill~\Aref{ex2-sub:lorem,ex2-sub:lorem2,ex2-sub:↵
      lorem4}.
18    \item \label{sub:lorem8} \textfill~\Aref{ex2-lid:met spatie, ex2-lid:lorem}.
19    \item \label{sub:lorem9} \textfill~\Aref{art:defs,art:lorem}.
20  \end{paras}
21 \end{paras}
22 \article{Etiam ac leo}\label{art:lorem2}
23 \para{A risus tristique nonummy}
24 \textfill
25 \article{Nulla in ipsum}\label{art:lorem3}
26 \textfill~\translation{See the attached}{Zie de bijgevoegde} \attachdocument[↵
  \translation{Definitions of this document}{Definities van dit document}]{example1.↵
  bib}{example1.bib}.
```

Codelijst 5: `example1.tex`

Het resultaat van dit voorbeeld is `example1-nl.pdf`.

```
1 \begin{center}
2   \Huge \translation{Example Two}{Voorbeeld Twee}
3 \end{center}
4 \article{Pactum}\label{art:agreement}
5 \begin{paras}
6   \item \label{lid:lorem} \textfill~\Gls{def1}.
7   \item \label{lid:lorem2} \textfill~\Gls{def2}.
8   \item \label{lid:lorem3} \label{lid:met spatie} \textfill
```

```

9 \item \label{lid:lorem4} \textfill
10 \item \label{lid:lorem5} \textfill~\Gls{def4} ipsum dolor sit amed.
11 \item \label{lid:inline} \textfill~\Gls{inline} ipsum dolor sit amed.\\
12 \printdefs[description]{Onderhandse akte}
13 \begin{paras}
14     \item \label{sub:lorem} \textfill
15     \item \label{sub:lorem2} \textfill
16     \item \label{sub:lorem3} \textfill
17     \item \label{sub:lorem4} \textfill
18 \end{paras}
19 \end{paras}
20 \article{Suspendisse}
21 \para{Aliquam}
22 \textfill

```

Code lijst 6: example2.tex

Het resultaat van dit voorbeeld is example2-nl.pdf.

```

1 % PDF management has to be active for the attachments of regulatory-attachments.
2 % Declared here only when the commandline did not do it already, so that a
3 % conformance flavour can be declared in front of this file (see test/Makefile),
4 % and not at all under tex4ht: PDF management pulls in the latex-lab list code,
5 % which leaves tex4ht nothing to turn a paras item into. \HCode is defined by
6 % the tex4ht run before this file is read, which is what the test is for.
7 \ifdefined\HCode\else\IfDocumentMetadataTF{}\DocumentMetadata{}\fi
8 \documentclass[12pt,dutch]{article}
9 \usepackage{babel}
10 \usepackage[md,alldefs]{regulatory}
11 \usepackage{lipsum}
12
13 \newcounter{lip}
14 \setcounter{lip}{1}
15 \newcommand\textfill{\lipsum[\arabic{lip}]\stepcounter{lip}}
16
17 \refdocument[ex2-]{example2-nl}{
18     defs=example2,
19     ref label=van Voorbeeld Twee
20 }
21
22 % The definitions of this example are declared in the Markdown source itself,
23 % both in its metadata and as a definition list, so no bib file is loaded.
24
25 % PDF/UA wants a dc:title, which \title does not set.
26 \hypersetup{pdftitle={Voorbeeld Markdown}}
27
28 \begin{document}
29     \begin{center}
30         \Huge Voorbeeld Markdown
31     \end{center}
32
33     \markdownInput{example.md}
34
35 \end{document}

```

Codelijst 7: md-example.tex

Het resultaat van dit voorbeeld is md-example.pdf.

```

1 ---
2 definitions:
3   - label: onderhands
4     name: Onderhandse akte
5     description: >-
6       Een akte die zonder tussenkomst van een ambtenaar is opgemaakt,
7       gedeclareerd in de metadata van dit bestand.
8 ---
9
10 # Definitiones {#art:defs}
11
12 De begrippen in deze overeenkomst hebben de betekenis die hieronder aan
13 hen wordt toegekend.
14
15 [Lorem]{#def1}
16
17 : Een begrip dat in de definitielijst van deze bron is gedeclareerd en
18   dat elders met een span wordt aangehaald.
19
20 [Nam dui]{#def2}
21
22 : Een tweede begrip, dat laat zien dat de langste naam de uitlijning
23   van de lijst bepaalt.
24
25 ::: {.definitions widest="Onderhandse akte"}
26
27 De volgende begrippen worden in deze overeenkomst gebruikt.
28
29 :::
30
31 # Quisque ullamcorper {#art:lorem}
32
33 1. {#lid:lorem} Lorem ipsum dolor sit amet, consectetur adipiscing elit.
34 2. {#lid:lorem2} Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
35   1. {#sub:lorem} Ut enim ad minim veniam, zie <#lid:lorem>.
36   2. {#sub:lorem2} Quis nostrud exercitation, zie <#ex2-lid:lorem>.
37   3. {#sub:lorem3} Duis aute irure dolor, zie <#lid:lorem,lid:lorem2>.
38   4. {#sub:lorem4} Excepteur sint occaecat, zie <#ex2-lid:lorem,ex2-lid:lorem2,ex2- ↵
39     lid:lorem3>.
39   5. {#sub:lorem5} Cupidatat non proident, zie lid [](<#lid:lorem2> en <#art:defs>.
40   6. {#sub:lorem6} Sunt in culpa qui officia deserunt mollit anim id est laborum.
41   7. {#sub:lorem7} Nam dui ligula, zie [het eerste lid](<#lid:lorem> en <#def1>.
42   8. Fringilla a, euismod sodales, sollicitudin vel, wisi. Dit lid draagt geen label ↵
43   .
44 # Suspendisse {#art:susp}
45
46 Morbi in sem quis dui placerat ornare. Een <#def1> en een
47 [onderhandse akte](<#onderhands>) worden hier aangehaald; artikel-2 blijft

```

48 | bij elkaar staan.

Codelijst 8: example.md

Het resultaat van dit voorbeeld is md-example.pdf.

```

1
2 @entry{def1,
3   name = {Lorem},
4   description = {\textfill}
5 }
6
7 @entry{def2,
8   name = {Nam dui},
9   description = {\textfill}
10 }
11
12 @entry{def3,
13   name = {Nulla},
14   description = {\textfill}
15 }

```

Codelijst 9: example1.bib

Het resultaat van dit voorbeeld is example1-nl.pdf.

```

1
2 @entry{def4,
3   name = {Suspendisse},
4   description = {\textfill}
5 }

```

Codelijst 10: example2.bib

Het resultaat van dit voorbeeld is example2-nl.pdf.

```

1 % The signature fields are what this example is for, so the PDF is written
2 % uncompressed: the suite then reads the annotations out of it with grep and needs
3 % nothing beyond a TeX installation to do so.
4 \ifdefined\pdfvariable\pdfvariable compresslevel=0 \else\pdfcompresslevel=0 \fi
5 % Declared here only when the commandline did not do it already, so that a
6 % conformance flavour can be declared in front of this file.
7 \IfDocumentMetadataTF{}\DocumentMetadata{}}
8 \documentclass[12pt,dutch]{article}
9 \usepackage{babel}
10 \usepackage{regulatory}
11
12 \newcommand\translation[2]{#2}
13
14 % A field draws a line to write on, which says nothing about how far the area
15 % reaches that a viewer will offer. This example frames it, so that the sheet shows
16 % on paper what the annotation covers in a reader: \fboxsep is zero, which puts the
17 % rule exactly on the edge of the annotation rectangle.
18 \newcommand\signaturebox[4][\%
19   \begingroup
20   \setlength\fboxsep{0pt}%
21   \fbox{\signaturefield[#1]{#2}{#3}{#4}}%
22   \endgroup
23 }
24
25 % PDF/UA wants a dc:title, which \title does not set.
26 \hypersetup{pdftitle={Voorbeeld Ondertekening}}
27
28 \begin{document}
29   \begin{center}
30     \Huge Voorbeeld Ondertekening
31   \end{center}
32
33   \article{Ondertekening}\label{art:sign}
34   \begin{paras}
35     \item \label{lid:opdrachtgever} De opdrachtgever tekent hieronder.\
36       \signaturebox[Handtekening van de opdrachtgever]{opdrachtgever}{6cm}{2cm}
37     \item \label{lid:opdrachtnemer} De opdrachtnemer tekent hieronder.\
38       \signaturebox[Handtekening van de opdrachtnemer]{opdrachtnemer}{6cm}{2cm}
39   \end{paras}
40 \end{document}

```

Codelijst 11: sign-example.tex

Het resultaat van dit voorbeeld is sign-example.pdf.


```

1 \IfDocumentMetadataTF{}{\DocumentMetadata{}}
2 \documentclass[12pt,dutch]{article}
3 \usepackage{babel}
4 \usepackage{regulatory}
5
6 \newcommand\translation[2]{#2}
7
8 % Every source in this file was checked on 2024-01-01, and this document speaks as
9 % of that same day: nothing changed between the two.
10 \sourcedate{2024-01-01}
11 \loadsources{sources}
12 % The third route: a source that stands in the document itself, with no file.
13 \newsource{uavg}{regeling}{%
14     citeertitel = Uitvoeringswet Algemene verordening gegevensbescherming,
15     afkorting = UAVG,
16     lidwoord = de, geldig = 2024-01-01%
17 }
18 % The same regulation as a German and a French document cite it: the register
19 % belongs to the source, so the source carries its name in the language it is cited
20 % in. These two use the English field names and the rest the Dutch ones; both are
21 % meant to work.
22 \newsource{avgde}{euact}{%
23     kind = Verordnung, number = {(EU) 2016/679},
24     shortname = Datenschutz-Grundverordnung, abbreviation = DSGVO,
25     register = german_eu, valid = 2024-01-01%
26 }
27 \newsource{avgfr}{euact}{%
28     kind = Règlement, number = {(UE) 2016/679},
29     shortname = {règlement général sur la protection des données},
30     abbreviation = RGPD, register = french_eu, valid = 2024-01-01%
31 }
32 % And as an English document cites it, which is the one register that turns the
33 % parts around: the deepest level first.
34 \newsource{avgen}{euact}{%
35     kind = Regulation, number = {(EU) 2016/679},
36     shortname = General Data Protection Regulation, abbreviation = GDPR,
37     determiner = the, register = english_eu, valid = 2024-01-01%
38 }
39 % In English a treaty takes an article and a regulation does not, and that follows
40 % from the kind and not from the register.
41 \newsource{vweu}{regulation}{%
42     citetitel = Treaty on the Functioning of the European Union,
43     kind = Treaty, abbreviation = TFEU, register = english_eu, valid = 2024-01-01%
44 }
45 \newsource{wetib}{regeling}{%
46     citeertitel = Wet inkomstenbelasting 2001,
47     afkorting = Wet IB 2001,
48     lidwoord = de, geldig = 2024-01-01%
49 }

```

```

50
51 \hypersetup{pdftitle={Voorbeeld Bronnen}}
52
53 % What the suite reads. A field that was read in wrongly does not show in the PDF,
54 % so it is named here where an assertion can reach it.
55 \newcommand\report[2]{\typeout{SRC #1 #2 = \srcfield{#1}{#2}}}
56 % The same for a citation: it is built first and typeset afterwards, so what was
57 % built is what an assertion can read. A citation in the wrong register fails
58 % nowhere, it simply stands there wrong.
59 \makeatletter
60 \newcommand\citereport[2]{#1\typeout{CITE #2 = \regulatory@src@last}}
61 \makeatother
62
63 % Which registers this bundle ships. The manual names them one by one and says
64 % which legal order each of them words, so the list is a claim like any other: a
65 % register added or renamed without the manual following would go unnoticed, since
66 % no document cites a register that is not there. Written in the order they were
67 % declared, which is the order the language files are read in.
68 \ExplSyntaxOn
69 \NewDocumentCommand \registerreport { }
70 { \iow_term:e { REGISTERS--\seq_use:Nn \g__regulatory_src_registers_seq { ,~ } } }
71 \ExplSyntaxOff
72
73 \begin{document}
74   \begin{center}
75     \Huge Voorbeeld Bronnen
76   \end{center}
77
78   \registerreport
79   \typeout{SRC bw6 type = \srctype{bw6}}
80   \report{bw6}{citeertitel}
81   \report{bw6}{afkorting}
82   \report{bw6}{boek}
83   \report{bw6}{bwbid}
84   \report{bw6}{geldig}
85   \typeout{SRC avg type = \srctype{avg}}
86   \report{avg}{roepnaam}
87   \report{avg}{afkorting}
88   \report{avg}{pb}
89   \report{uavg}{citeertitel}
90   \typeout{SRC bw6 leeg = [\srcfield{bw6}{ditveldbestaatniet}]}
91   \ifsourceexists{avg}{\typeout{SRC avg bestaat}}{\typeout{SRC avg ontbreekt}}
92   \ifsourceexists{onbekend}{\typeout{SRC onbekend bestaat}}{\typeout{SRC onbekend ↵
    ontbreekt}}
93
94   \citereport{\srcref[artikel=96, lid=2, onderdeel=c]{bw6}}{nl-voluit}
95   \citereport{\srcref*[artikel=96, lid=2, onderdeel=c]{bw6}}{nl-verkort}
96   \citereport{\srcref[artikel=6, lid=1, onderdeel=a]{avg}}{eu-voluit}
97   \citereport{\srcref*[artikel=6, lid=1, onderdeel=a]{avg}}{eu-verkort}
98   \citereport{\srcref{avg}}{bron-voluit}

```

```

99 \citereport{\srcref*{avg}}{bron-verkort}
100 \citereport{\srcref[artikel=3.126, lid=1, onderdeel=d, onder=2]{wetib}}{vierde- ↵
niveau}
101 \citereport{\srcref[artikel=6, lid=1, onderdeel=a]{avgde}}{de-voluit}
102 \citereport{\srcref*[artikel=6, lid=1, onderdeel=a]{avgde}}{de-verkort}
103 \citereport{\srcref[artikel=6, lid=1, onderdeel=a]{avgfr}}{fr-voluit}
104 \citereport{\srcref*[artikel=6, lid=1, onderdeel=a]{avgfr}}{fr-verkort}
105 % The same two citations with the English key names. The Dutch ones are
106 % aliases of these, so both lines have to produce the same thing word for word;
107 % where anything differs, the alias is not what it says it is.
108 \citereport{\srcref[article=96, paragraph=2, point=c]{bw6}}{nl-engelse-sleutels}
109 \citereport{\srcref[article=3.126, paragraph=1, point=d, subpoint=2]{wetib}}{↵
vierde-niveau-engels}
110 % Lists and runs: the author writes what he means, the register decides how it
111 % reads.
112 \citereport{\srcref[artikel=3.126, lid=1, onderdeel={c,d}]{wetib}}{lijst-nl}
113 \citereport{\srcref[artikel={162--164}]{bw6}}{bereik-nl}
114 \citereport{\srcref*[artikel={162--164}]{bw6}}{bereik-verkort}
115 \citereport{\srcref[artikel=6, lid=1, onderdeel={c,e}]{avg}}{lijst-eu}
116 \citereport{\srcref[artikel=6, lid=1, onderdeel={c,e}]{avgde}}{lijst-de}
117 \citereport{\srcref[artikel=6, lid=1, onderdeel={a--c}]{avgde}}{bereik-de}
118 \citereport{\srcref[artikel=6, lid={1,2,4}]{avgfr}}{lijst-fr}
119 % English turns the parts around and writes the paragraph inside the article.
120 \citereport{\srcref[artikel=6, lid=1, onderdeel=a]{avgen}}{en-voluit}
121 \citereport{\srcref*[artikel=6, lid=1, onderdeel=a]{avgen}}{en-verkort}
122 \citereport{\srcref[artikel=6]{avgen}}{en-artikel}
123 \citereport{\srcref[lid=1, onderdeel=e]{avgen}}{en-lid-los}
124 \citereport{\srcref[artikel=58, lid=1, onderdeel={e,f}]{avgen}}{en-lijst}
125 \citereport{\srcref[artikel={12--15}]{avgen}}{en-bereik}
126 \citereport{\srcref[artikel=58, lid=2, onderdeel={a--c}]{avgen}}{en-bereik-punten}
127 \citereport{\srcref[lid={1--3}]{avgen}}{en-bereik-leden}
128 \citereport{\srcref[artikel=6, lid={1,2,3}]{avgen}}{en-leden}
129 \citereport{\srcref[artikel=6]{vweu}}{en-verdrag}
130
131 \article{Bronnen}\label{art:bronnen}
132 \begin{paras}
133 \item \label{lid:regeling} \srcfield{bw6}{citeertitel} (\srcfield{bw6}{↵
afkorting}),
134 Boek-\srcfield{bw6}{boek}.
135 \item \label{lid:euhandeling} \srcfield{avg}{soort}-\srcfield{avg}{nummer},
136 \srcfield{avg}{roepnaam} (\srcfield{avg}{afkorting}).
137 \item \label{lid:eigen} \srcfield{uavg}{citeertitel} (\srcfield{uavg}{afkorting ↵
}).
138 \end{paras}
139
140 \article{Verwijzingen}\label{art:verwijzingen}
141 \begin{paras}
142 \item \label{lid:voluit} Zoals bepaald in
143 \srcref[artikel=96, lid=2, onderdeel=c]{bw6}.
144 \item \label{lid:verkort} Zoals bepaald in

```

```

145         \srcref*[artikel=96, lid=2, onderdeel=c]{bw6}.
146         \item \label{lid:eu} Zoals bepaald in
147         \srcref[artikel=6, lid=1, onderdeel=a]{avg}.
148     \end{paras}
149 \end{document}

```

Codelijst 12: source-example.tex

Het resultaat van dit voorbeeld is source-example.pdf.

```

1 % The sources source-example cites. This file is the coverage of the reader as
2 % well: it uses all four forms the subset knows, this comment line and the empty
3 % line below it included.
4
5 @regeling{bw6,
6   citeertitel = {Burgerlijk Wetboek},
7   lidwoord = {het},
8   afkorting = {BW},
9   boek = {6},
10  bwbid = {BWR0005289},
11  geldig = {2024-01-01},
12 }
13
14 @euhandeling{avg,
15   soort = {Verordening},
16   nummer = {(EU) 2016/679},
17   titel = {betreffende de bescherming van natuurlijke personen in verband met de ↵
18           verwerking van persoonsgegevens},
19   roepnaam = {Algemene verordening gegevensbescherming},
20   afkorting = {AVG},
21   pb = {PbEU 2016, L 119/1},
22   geldig = {2024-01-01},
23 }

```

Codelijst 13: sources.bib

Het resultaat van dit voorbeeld is source-example.pdf.

14 Implementatie

De zeven modules van de bundel worden hieronder toegelicht, in de volgorde waarin ze geladen worden: de zes die `regulatory` laadt, en `regulatory-md`, die zich daarbij voegt zodra `markdown` geladen is. Daarna volgen de Markdown-syntaxuitbreiding en de `tex4ht`-configuratie, die geen modules zijn maar evengoed uit een eigen bron gegenereerd worden. Iedere module is geschreven als een `.dtx` bestand, waaruit zowel dit hoofdstuk als de module zelf gegenereerd wordt. De toelichting is, net als de broncode, in het Engels gesteld.

14.1 `regulatory-struct`

14.1.1 The bundle

The `regulatory` package itself is nothing more than a wrapper around the six modules that are always loaded. `regulatory-sign` is among them: it defines two commands and emits nothing until one of them is used, and a document that needs a signature field is exactly the kind this bundle is for. `regulatory-md` is the seventh and is not loaded here; a hook pulls it in as soon as `markdown` is. Every option is handed over to `regulatory-struct`, which administers the options for the whole bundle. Every other module requests its own prerequisites, so the order below merely documents the dependency chain.

```
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{regulatory-struct}}
\ProcessOptions\relax
\RequirePackage{regulatory-struct}
\RequirePackage{regulatory-defs}
\RequirePackage{regulatory-ref}
\RequirePackage{regulatory-attachments}
\RequirePackage{regulatory-sign}
\RequirePackage{regulatory-sources}
```

14.1.2 Pakketopties

Everything from here on belongs to `regulatory-struct`. The options are keys of the `regulatory` family of `l3keys`, which the kernel offers and which this bundle uses for every key it reads: the options here, the argument of `\srcref`, and the fields of an attachment. The registers of paragraaf 9 are the one thing that is not keys but a store, and those are property lists. `ifthen` rather than `xifthen`: measured, nothing here uses a command of the second that the first does not have, and `xifthen` drags in `calc`, which changes how every `\setlength` in the document is parsed. `xspace` stood here as well and is gone; it was called nowhere.

```
\RequirePackage{ifthen}
```

```
\ifregulatory@markdown Each switch option is stored in a \newif switch, which .legacy_if_set:n sets. The swit-
\ifregulatory@bibtogls ches are shared by every module, which is the reason for keeping them in this base
  \ifregulatory@legacy module. The nameinlink switch is declared with \newboolean, because it is queried with
  \ifregulatory@alldefs \ifthenelse in regulatory-ref as well; \newboolean builds a \newif switch, so the same
\ifregulatory@nameinlink property reaches it.
```

```

\newif\ifregulatory@markdown
\newif\ifregulatory@bibtogls
\newif\ifregulatory@legacy
\newif\ifregulatory@alldefs
\newboolean{regulatory@nameinlink}
\ExplSyntaxOn
\tl_new:N \regulatory@defstyle
\tl_new:N \regulatory@sourcestyle
\tl_new:N \regulatory@attachkind
\keys_define:nn { regulatory }
{
  md          .legacy_if_set:n = regulatory@markdown ,
  md          .default:n       = true ,
  md          .initial:n       = false ,
  bib2gls     .legacy_if_set:n = regulatory@bibtogls ,
  bib2gls     .default:n       = true ,
  bib2gls     .initial:n       = true ,
  legacy      .legacy_if_set:n = regulatory@legacy ,
  legacy      .default:n       = true ,
  legacy      .initial:n       = false ,
  alldefs     .legacy_if_set:n = regulatory@alldefs ,
  alldefs     .default:n       = true ,
  alldefs     .initial:n       = false ,
  nameinlink  .legacy_if_set:n = regulatory@nameinlink ,
  nameinlink  .default:n       = true ,
  nameinlink  .initial:n       = true ,
  defstyle    .tl_set:N        = \regulatory@defstyle ,
  defstyle    .initial:n       = alttree ,

```

The style a citation of an external source is written in is a choice and not a string, so that a value this bundle does not know is refused where it is given. It used to be stored as given, and a misspelling then reached every register lookup as a path that holds nothing: a citation came out with its numbers and without a single word, and nothing said so.

```

sourcestyle .choice: ,
sourcestyle / full .code:n = \tl_set:Nn \regulatory@sourcestyle { full } ,
sourcestyle / short .code:n = \tl_set:Nn \regulatory@sourcestyle { short } ,
sourcestyle / voluit .code:n = \tl_set:Nn \regulatory@sourcestyle { full } ,
sourcestyle / verkort .code:n = \tl_set:Nn \regulatory@sourcestyle { short } ,
sourcestyle .initial:n = full ,

```

How a link to an attachment is made is a choice for the same reason, and the two values are not a matter of taste. `goto` is the default and the semantically right one: an embedded go-to action names the file inside this document. It is also the one poppler does not implement, and that is the engine under most viewers on a free desktop, so there the link does nothing at all. `annotation` places a file attachment annotation over the same text instead, which both Acrobat and poppler do implement, at the price of an annotation that the A-standards want an appearance for. Measured, not read out of a manual: the shared library of poppler names `GoTo`, `GoToR`, `Launch` and `Named` among its actions and `GoToE` nowhere, and `FileAttachment` among its annotations.

```

attachmentlink .choice: ,

```

```

attachmentlink / goto .code:n =
  \tl_set:Nn \regulatory@attachkind { goto } ,
attachmentlink / annotation .code:n =
  \tl_set:Nn \regulatory@attachkind { annotation } ,
attachmentlink .initial:n = goto ,
bronstijl .meta:n = { sourcestyle = {#1} } ,
}
\ExplSyntaxOff

```

`\regulatory@defstyle` The two options that carry a value rather than a switch. `defstyle` holds the name of the style `\printdefs` reaches for when it is given no style of its own; `sourcestyle` holds full or short, which is the arrangement `\srcrcf` writes a citation in.

`\regulatorysetup` Sets the same keys after the package is loaded, and is what a `regulatory.cfg` writes. A local configuration file is read before the options given to the package are processed, so a document always wins from the directory it stands in.

```

\ExplSyntaxOn
\NewDocumentCommand \regulatorysetup { m }
{ \keys_set:nn { regulatory } { #1 } }
\ExplSyntaxOff

\InputIfFileExists{regulatory.cfg}{%
  \PackageInfo{regulatory}{Using local configuration file}%
}{%
  \PackageInfo{regulatory}{No local configuration file}%
}

\ProcessKeyOptions[regulatory]

```

14.1.3 Vereisten

Whenever markdown is loaded, the renderers of `regulatory-md` translate its constructs into the structures of this module. The hook fires whatever the order of both packages is, since it is executed right away for a markdown that is loaded already.

It cannot fire in the middle of this module, though, and the `md` option below makes it try: `regulatory-md` needs `regulatory-defs`, which loads `glossaries`, and `glossaries` decides whether its entries become hyperlinks by looking for `hyperref`, which this module only loads a few lines further down. A document with the `md` option therefore used to get definitions without a single link, and nothing said so. The load is postponed to the end of this file instead, where every prerequisite is in place.

```

\newif\ifregulatory@md@pending
\newif\ifregulatory@ready
\AddToHook{package/markdown/after}{%
  \ifregulatory@ready
    \RequirePackage{regulatory-md}%
  \else
    \regulatory@md@pendingtrue
  \fi
}

```

The `md` option comes down to loading markdown itself. This has to happen before `enumitem` is loaded, otherwise markdown breaks the list definitions further down, which is the reason for loading it here rather than leaving it to the document. The options markdown is to be given are not stated here but in `regulatory-md`, which sets them with `\markdownSetup`: a document that loads markdown by itself then gets the same conversion as one that used the `md` option.

```
\ifregulatory@markdown
\RequirePackage{markdown}
\fi
```

The remaining prerequisites of the whole bundle are loaded here, in this exact order. The `xr` option of `zref` is only used by `regulatory-attachments`, but has to be loaded before `hyperref`, which is loaded by this module. `glossaries` is not among them: it belongs to `regulatory-defs`, so a document that only wants the structures does not carry it.

`xr-hyper` used to stand beside it and is gone. Nothing in this bundle calls `\externaldocument`; the references to another document are `\zexternaldocument` of `zref-xr`, and the links they produce come from `zref-hyperref`. Measured on the same example built both ways: eleven `/GoToR` actions, eighteen link annotations and two file specifications, with the package and without it. A document that wants `\externaldocument` loads `xr-hyper` itself, which is where that choice belongs.

```
\RequirePackage{xcolor}
\RequirePackage{enumitem}
\RequirePackage{fmtcount}
\RequirePackage{scrextend}
\RequirePackage[user,counter,hyperref,xr]{zref}
\RequirePackage{zref-xr,zref-user,zref-clever,zref-hyperref}
\RequirePackage{hyperref}
\RequirePackage{translations}
```

`\regulatory@ifmodule` The language definition files of `regulatory-ref` configure whatever module happens to be loaded. This test tells them which parts to skip. Its first argument is the module name without the `regulatory-` prefix.

```
\newcommand\regulatory@ifmodule[3]{%
  \@ifundefined{ver@regulatory-#1.sty}{#3}{#2}%
}
```

`\GetTranslation` English is the fallback of every translation used by this bundle. The languages themselves are declared in the language definition files of `regulatory-ref`.

```
\DeclareTranslationFallback{article}{article}
\DeclareTranslationFallback{Article}{Article}
\DeclareTranslationFallback{articles}{articles}
\DeclareTranslationFallback{Articles}{Articles}
\DeclareTranslationFallback{paragraph}{paragraph}
\DeclareTranslationFallback{Paragraph}{Paragraph}
\DeclareTranslationFallback{paragraphs}{paragraphs}
\DeclareTranslationFallback{Paragraphs}{Paragraphs}
\DeclareTranslationFallback{point}{point}
```



```

\DeclareTranslationFallback{Point}{Point}
\DeclareTranslationFallback{points}{points}
\DeclareTranslationFallback{Points}{Points}
\DeclareTranslationFallback{Definitions}{Definitions}

```

14.1.4 Kleuren

All colors default to black, so they only have to be redefined, e.g. with `\colorlet`, whenever headings should stand out.

```

\providecolor{artlabel}{rgb}{0,0,0}
\providecolor{arttitle}{rgb}{0,0,0}
\providecolor{paralabel}{rgb}{0,0,0}
\providecolor{paratitle}{rgb}{0,0,0}
\providecolor{sublabel}{rgb}{0,0,0}
\providecolor{subtitle}{rgb}{0,0,0}

```

14.1.5 Koppen

`\article` Articles and paragraphs are ordinary counters, where the paragraph counter is reset by `\para` the article counter.

```

\newcounter{article}
\newcounter{para}
\counterwithin{para}{article}

\renewcommand{\thearticle}{\arabic{article}}
\renewcommand{\thepara}{\arabic{para}}

```

The headings themselves are provided by `titlesec`. As soon as the new $\LaTeX$ heading interface is available, `\ParseLaTeXeHeading` is defined and both headings are declared as heading instances instead, since `titlesec` and the new interface exclude each other.

```

\@ifundefined{ParseLaTeXeHeading}{%
  \RequirePackage{titlesec}%
  \titleclass{\article}[0]{straight}%
  \titleformat{\article}{\normalfont\color{arttitle}}
    {\bfseries\color{artlabel}\GetTranslation{Article} \thearticle.\quad}{1em}{}%
  \titlespacing*{\article}{0pt}{3.5ex plus 1ex minus .2ex}{5pt}%
  \titleclass{\para}{straight}[\article]%
  \titleformat{\para}{\normalfont\color{paratitle}}
    {\normalfont\color{paralabel}\thearticle.\thepara.\quad}{1em}{}%
  \titlespacing*{\para}{0pt}{3.5ex plus 1ex minus .2ex}{5pt}%
}%
\DeclareInstance{heading}{regulatory-article}{display}{
  name           = article,
  level          = 1,
  heading-decls  = \normalfont\color{arttitle},
  prefix         = \GetTranslation{Article},
  prefix-decls   = \bfseries\color{artlabel},
  number-decls   = \bfseries\color{artlabel},
  number-format  = \thearticle.,
}

```

```

        headformat-instance = regulatory-article,
        before-vspace       = 3.5ex plus 1ex minus .2ex,
        after-vspace        = 5pt,
    }%
\DeclareInstance{headformat}{regulatory-article}{hang}{
    heading-indent    = 0pt,
    number-title-sep = 2em,
}%
\DeclareInstance{heading}{regulatory-para}{display}{
    name              = para,
    parent-name       = article,
    level             = 2,
    heading-decls      = \normalfont\color{paratitle},
    number-decls       = \normalfont\color{paralabel},
    number-format      = \thearticle.\thepara.,
    headformat-instance = regulatory-para,
    before-vspace      = 3.5ex plus 1ex minus .2ex,
    after-vspace       = 5pt,
}%
\DeclareInstance{headformat}{regulatory-para}{hang}{
    heading-indent    = 0pt,
    number-title-sep = 2em,
}%
\DeclareDocumentCommand\article{som}{%
    \ParseLaTeXeHeading{regulatory-article}{#1}{#2}{#3}%
}%
\DeclareDocumentCommand\para{som}{%
    \ParseLaTeXeHeading{regulatory-para}{#1}{#2}{#3}%
}%
}

```

Both headings take part in the table of contents at the level of subsections and subsubsections.

```

\newcommand{\l@article}{\@dottedtocline{1}{1.5em}{2.3em}}
\def\toclevel@article{2}
\newcommand{\l@para}{\@dottedtocline{1}{3.8em}{2.3em}}
\def\toclevel@para{3}

```

14.1.6 Lijsten

`provisions` (*env.*) Provisions are a flat enumeration within a section, whereas paragraphs and points are
`paras` (*env.*) the two levels of the `paras` environment. The label of the first level repeats the article number, and the second level is enumerated alphabetically with `\abalphnum` of `fmtcount`, so it isn't limited to 26 points.

```

\newcounter{provisions}
\newlist{provisions}{enumerate}{1}
\setlist[provisions,1]{label=\arabic*.,align=left,itemsep=0pt}
\counterwithin{provisions}{section}

```

```

\newlist{paras}{enumerate}{2}
\@ifpackageloaded{latex-lab-enumeritem}{%
  \setlist[paras,1]{label=\thearticle.\arabic*. ,align=left,itemsep=0pt}%
  \setlist[paras,2]{label=\abalphnum{\arabic*}},itemsep=0pt}%
}{%
  \setlist[paras,1]{label=\thearticle.\arabic*. ,ref=\arabic*,align=left,itemsep=0pt}%
  \setlist[paras,2]{label=\abalphnum{\arabic*}},ref=\arabic*,itemsep=0pt}%
}
\counterwithin{parasi}{article}
\counterwithin{parasii}{parasi}

```

The list counters of `enumeritem` are numbered per list, while references have to address them as part of the document structure. Therefore both representations are rewritten right before every item.

```

\AddToHook{cmd/item/before}{%
  \@ifundefined{c@parasi}{}{%
    \renewcommand\theparasi{\thearticle.\arabic{parasi}}%
    \renewcommand\theparasii{\theparasi\abalphnum{\arabic{parasii}}}%
  }%
}

```

14.1.7 Verwijseigenschappen

Every label records the article, paragraph and point it appears in, which is what `regulatory-ref` builds its references upon. A paragraph is either a `\para` heading or the first level of the `paras` environment, so both are stored in the same property.

```

\zref@newprop{article}[-1]{\number\value{article}}
\zref@newprop{para}[-1]{\ifnum\value{para}>0\number\value{para}\else\number\value{parasi}\fi}
\zref@newprop{sub}[-1]{\number\value{parasii}}
\zref@addprops{main}{article,para,sub}

```

Every `\label` records them without anything further being asked of it: `\zref@addprops` puts the properties on the main list, and `zref-clever` labels with that list from the `label` hook, which it installs itself unless its `labelhook` option says otherwise.

This package used to hook `\zlabel` onto `\label` once more, through `\RegisterPostLabelHook` of `xassocnt`. That wrote a second and identical `\zref@newlabel` for every label, which the next run reads back as a label that is defined twice: every document built with this package carried a dozen or more ‘multiply defined’ warnings and nothing else went wrong, which is exactly why it took this long to notice.

References made with `\zceref` of `zref-clever` get the designations of these structures, so that both ways of referring stay consistent. The counters `parasi` and `parasii` are the list variants of a paragraph and a point.

```

\zcRefTypeSetup{article}{%
  Name-sg = \GetTranslation{Article},
  name-sg = \GetTranslation{article},
  Name-pl = \GetTranslation{Articles},
  name-pl = \GetTranslation{articles}
}

```

```

\zcRefTypeSetup{para}{%
  Name-sg = \GetTranslation{Paragraph},
  name-sg = \GetTranslation{paragraph},
  Name-pl = \GetTranslation{Paragraphs},
  name-pl = \GetTranslation{paragraphs}
}
\zcRefTypeSetup{parasi}{%
  Name-sg = \GetTranslation{Paragraph},
  name-sg = \GetTranslation{paragraph},
  Name-pl = \GetTranslation{Paragraphs},
  name-pl = \GetTranslation{paragraphs}
}
\zcRefTypeSetup{parasii}{%
  Name-sg = \GetTranslation{Point},
  name-sg = \GetTranslation{point},
  Name-pl = \GetTranslation{Points},
  name-pl = \GetTranslation{points}
}

```

14.1.8 Taalbestanden

Every language supported by this bundle has its own definition file `regulatory-⟨language⟩.def`, comparable to the `fc-⟨language⟩.def` files of `fmtcount`. Such a file declares the translations of the bundle and, whenever `regulatory-ref` is loaded, the reference formats of that language. The loading mechanism lives in this module, so a document that only loads `regulatory-struct` is translated as well.

`\ProvidesRegulatoryLanguage` Every language file identifies itself with this macro, just like `\ProvidesFile` does.

```

\newcommand\ProvidesRegulatoryLanguage[1]{%
  \ProvidesFile{regulatory-#1.def}%
}

```

`\RegulatoryLoadLanguage` Loads the definition file of `⟨language⟩`, unless it is already loaded. A language without a definition file is silently accepted, since `\regulatory@load@languages` offers every language known to `babel`, most of which this bundle does not support. Definition files are read at the end of the preamble, where the at sign is no letter, so its category code is set and restored around them. The tilde as well: `ℒTEX` makes it an ordinary space while it reads a package file, and a language file is largely a set of decisions about spacing. That is not a detail — measured, a tilde that quietly became a space took every non-breaking space in a citation register with it.

```

\newcommand\RegulatoryLoadLanguage[1]{%
  \@ifundefined{ver@regulatory-#1.def}{%
    \edef\regulatory@restorat{%
      \catcode`\noexpand\@=\the\catcode`\@ \relax
      \catcode`\noexpand\~=\the\catcode`\~ \relax}%
    \makeatletter
    \catcode`\-=13 \relax
    \InputIfFileExists{regulatory-#1.def}{%

```

```

\PackageInfo{regulatory}{Loaded language definitions for `#1'}%
}{%
\PackageInfo{regulatory}{No language definitions for `#1'}%
}%
\regulatory@restoreat
}{}%
}

```

`\regulatory@load@languages` English is always loaded, as it holds the defaults every other language builds upon. The languages of the document itself are only known once `babel` or `polyglossia` is loaded, which is why this runs at the very end of the preamble. Documents using another language selection mechanism can call `\RegulatoryLoadLanguage` in their preamble instead.

Every language this bundle ships is read whatever the document says, and not only the ones it speaks. A language file holds two things: how that language words a reference, and how a legal order writing in it words a citation. The second is a property of the source and not of the document — a Dutch contract cites a German regulation in German — so a register that were only read when the document spoke its language would be missing exactly where it is needed. Reading a language a document does not speak costs nothing that shows: everything in such a file is keyed by the name of its language, so it defines what nobody asks for.

Three places are asked, because no single one of them is right. `\bbl@loaded` is the list `babel` keeps, and it is empty under the `ini` based loading that `provide=*` selects; `\xpg@loaded` is the same list for `polyglossia`; and `\language` is the one language that is current, which is correct in every case but names only one. Measured on the same document with only the `babel` line differing: the classic form gave “Artikel 1, eerste lid” and the `ini` form gave “Article 1(1)”, because the list was empty and Dutch was therefore never loaded. Nothing failed, and the reference was simply worded in another language than the document.

```

\ExplSyntaxOn
\clist_new:N \g_regulatory_languages_clist

\cs_new_protected:Npn \regulatory@collect@languages
{
  \clist_gset:Nx \g_regulatory_languages_clist
  {
    \cs_if_exist_use:CF { bbl@loaded } { } ,
    \cs_if_exist_use:CF { xpg@loaded } { } ,
    \cs_if_exist_use:CF { language } { }
  }
  \clist_gremove_duplicates:N \g_regulatory_languages_clist
}

\clist_const:Nn \c_regulatory_shipped_clist { english , dutch , german , french }

\cs_new_protected:Npn \regulatory@load@languages
{
  \regulatory@collect@languages
}

```

```

\clist_map_inline:Nn \c_regulatory_shipped_clist
{ \RegulatoryLoadLanguage { ##1 } }
\clist_map_inline:Nn \g_regulatory_languages_clist
{ \RegulatoryLoadLanguage { ##1 } }
}
\ExplSyntaxOff
\AddToHook{begindocument/before}{\regulatory@load@languages}

```

14.1.9 Wat de run niet kan verhelpen

`\regulatory@checkenvironment` Three things this bundle cannot do anything about, and which show up in the result rather than in the log. Each of them was found the hard way, by reading an output that looked finished, so each is said out loud once at the beginning of the document.

A language without a definition file is accepted where it is loaded, since `babel` may well be asked for a language that is only used for a quotation. What it costs is only visible in a reference: the wording falls back on English, so a Dutch document that never declared Dutch reads “Article 1(1)” in the middle of its own sentences. The languages are listed rather than counted, since the one that is missing is the point. They are the same three sources the loader reads, which matters more than it looks: a check that asked `\bbl@loaded` alone would share the blind spot of the thing it is meant to watch, and stay silent in exactly the case it exists for.

Under `tex4ht`, the configuration of this bundle is what turns an article into a section with a heading. Without it the conversion still succeeds and produces neither: a heading becomes a paragraph of text and nothing says so. The test is for the configuration itself, not for the file, so a copy that shadows the one of the package fails it just as loudly as a missing one.

Also under `tex4ht`: PDF management pulls in the list code of `latex-lab`, which leaves `tex4ht` nothing to turn a `paras` item into, and the items come out as running text. A document that is converted as well as `typeset` declares its `\DocumentMetadata` conditionally, which is what the examples of this bundle do.

```

\ExplSyntaxOn
\cs_new:Npn \regulatory@languages { \clist_use:Nn \g_regulatory_languages_clist { , } }
\ExplSyntaxOff

\newcommand\regulatory@checkenvironment{%
  \def\regulatory@missinglangs{}%
  \@for\regulatory@lang:=\regulatory@languages\do{%
    \ifundefined{ver@regulatory-\regulatory@lang.def}{%
      \edef\regulatory@missinglangs{%
        \regulatory@missinglangs
        \ifx\regulatory@missinglangs\empty\else, \fi
        \regulatory@lang}%
    }{}%
  }%
  \ifx\regulatory@missinglangs\empty\else%
    \PackageWarning{regulatory}{%

```

```

        No definitions for the language(s) \regulatory@missinglangs;\MessageBreak
        references in them are worded in English. Reported}%
\fi%
\ifdefined\HCode%
  \@ifundefined{a:regarticle}{%
    \PackageWarning{regulatory}{%
      regulatory.4ht was not found, so an article and a\MessageBreak
      paragraph convert to running text without a heading.\MessageBreak
      Reported}%
    }{}%
  \IfDocumentMetadataTF{%
    \PackageWarning{regulatory}{%
      PDF management is active under tex4ht, which leaves it\MessageBreak
      nothing to turn a paras item into. Declare\MessageBreak
      \string\DocumentMetadata\space only when \string\HCode\space is undefined.\MessageBreak
      Reported}%
    }{}%
  }{}%
\fi%
}
\AddToHook{begindocument/end}{\regulatory@checkenvironment}

```

14.1.10 Uitgestelde modules

Everything this module provides is now in place, so `regulatory-md` can be loaded for the markdown that was loaded above, or for one the document loaded before this module.

```

\regulatory@readytrue
\ifregulatory@md@pending
\RequirePackage{regulatory-md}
\fi

```

14.2 regulatory-defs

14.2.1 Vereisten

`glossaries-extra` is loaded whichever way the definitions are fed in. Only the `record` option depends on `bib2gls`, since that is what hands the recording over to it; everything else this module uses – `\printunsrtglossary`, `\glsdefpostlink`, the `almtree` style – has to be there for the routes that do without `bib2gls` as well.

```

\RequirePackage{regulatory-struct}
\RequirePackage[sanitize=none,nopostdot]{glossaries}
\ifregulatory@bibtogls
\RequirePackage[record,style=almtree]{glossaries-extra}
\else
\RequirePackage[style=almtree]{glossaries-extra}
\fi

```

14.2.2 De definitielijst

Definitions of the document itself and definitions of other documents are kept apart in two glossary types. The externals type is added by regulatory-attachments.

```
\newglossary*{definitions}{Definitions}
```

Definition lists are typeset as part of the running text, so the section heading of the glossary is dropped altogether. Whenever the list is the first thing in the document, it gets an article heading of its own.

```
\newboolean{regulatory@defslisted}
\setboolean{regulatory@defslisted}{false}
```

Whether the list was printed at all is counted rather than switched, since a counter is global: the list may well be printed from inside a group — a Markdown conversion is one — and a switch set there would be back to false by the end of the document, which is where the question is asked.

```
\newcounter{regulatory@defsprinted}
\newboolean{regulatory@indexeddefs}
\setboolean{regulatory@indexeddefs}{false}
```

```
\renewcommand{\glossarysection}[2][\@gobble{#1}\@gobble{#2}]{%}
```

A listing that opens under a heading opens a line of its own first, and this pulls that line back. How much there is to pull back belongs to the style and not to the preamble: the alttree that glossaries brings leaves such a line and the two list styles below leave none, so a correction of one line over all three put the first entry of a list on the line of the heading itself. A style that needs no correction says so.

```
\newcommand\regulatory@defsrise{-\baselineskip}
\setglossarypreamble[definitions]{%
  \ifnum\value{article}=0%
    \article{\GetTranslation{Definitions}}\label{art:definitions}%
    \vspace*{\regulatory@defsrise}%
  \else%
    \ifnum\value{parasi}=0%
      \vspace*{\regulatory@defsrise}%
    \fi%
  \fi%
}
```

14.2.3 Lijststijlen

regulatory-labeling	A definition list is a term with its meaning, which is what a description list is for. Two of
regulatory-description	them are offered next to the alttree style glossaries brings: one built on the labeling environment of scrextend, which aligns every description on the widest name given to \printdefs, and one on the description environment of the class, which leaves the alignment to the class. Both put the anchor on the name, so a \gls link lands on the term rather than beside it.

Which one to reach for is a question of where the document goes. The alttree style is the narrowest of the three in HTML: it is written in terms of boxes and widths and

comes out as one flat paragraph, while both list styles carry the term and its meaning as separate elements.

```
\newcommand\regulatory@defstyle@body[1]{%
  \renewcommand*\glossaryheader{}{}%
  \renewcommand*\glsgroupheading[1]{}%
  \renewcommand*\glsgroupskip{}{}%
  \renewcommand*\glossentry[2]{%
    \item[\glstarget{##1}{\glossentryname{##1}}]%
    \glossentrydesc{##1}\glspostdescription}%
  \renewcommand*\subglossentry[3]{%
    \glossentrydesc{##2}\glspostdescription}%
}

\newglossarystyle{regulatory-labeling}{%
  \renewcommand\regulatory@defsrise{\z@}%
  \renewenvironment{theglossary}%
    {\begin{labeling}{\@glswidestname}}%
    {\end{labeling}}%
  \regulatory@defstyle@body{}%
}

\newglossarystyle{regulatory-description}{%
  \renewcommand\regulatory@defsrise{\z@}%
  \renewenvironment{theglossary}%
    {\begin{description}}%
    {\end{description}}%
  \regulatory@defstyle@body{}%
}
```

`\describe` Prints the description of a single definition and marks it as the anchor of that definition, so hyperlinks of `\gls` end up at the right spot.

```
\newcommand\describe[1]{\setboolean{regulatory@defslisted}{true}\glstarget{#1}{\glsentrydesc{#1}}}
```

`\printdefs` Prints the whole definition list, aligned on the width of $\langle width\ of\ text \rangle$. Within a paragraph the list is wrapped in a minipage, to keep it from being broken across pages halfway an enumeration.

The width is read before it is set, since `\glssetwidest` stores it in the very macro one would reach for to name the widest name already known, and setting that macro to itself is a loop the run does not come back from.

The optional argument picks the style. A name for which a `regulatory-` style exists is taken to mean that one, so `labeling` and `description` reach the two above; anything else is handed to glossaries as it stands, which leaves every style it knows available. Without the argument the `defstyle` option decides.

Which of the two printing commands applies depends on how the entries arrived rather than on the `bib2gls` option. `\printglossary` needs a sorted and indexed glossary, which only `\loadglsentries` produces; `bib2gls` and `\newdefinition` both hand over entries that

are already in the order they should be printed in, and `\printunsrtglossary` prints those without an external tool having to run at all.

```
\newcommand\printdefs[2][\regulatory@defstyle]{%
  \renewcommand*\glsnamefont[1]{\textmd{##1}}%
  \setboolean{regulatory@defslisted}{true}%
  \stepcounter{regulatory@defsprinted}%
  \protected@edef\regulatory@thiswidest{#2}%
  \expandafter\glssetwidest\expandafter{\regulatory@thiswidest}%
  \edef\regulatory@thisstyle{%
    \ifcsname @glsstyle@regulatory-#1\endcsname regulatory-#1\else#1\fi}%
  \ifnum\value{parasi}>0%
    \def\@wrapper##1{\begin{minipage}{\linewidth}##1\end{minipage}}%
  \else%
    \def\@wrapper##1{##1}%
  \fi%
  \@wrapper{%
    \begingroup%
    \setglossarystyle{\regulatory@thisstyle}%
    \ifthenelse{\boolean{regulatory@indexeddefs}}{%
      \printglossary[type=definitions,title={},nonumberlist=true]%
    }{%
      \printunsrtglossary[type=definitions,title={},nonumberlist=true]%
    }%
    \endgroup%
  }%
}
```

`\loadglsdefs` Loads the definitions of `\src` under the definitions type. With `bib2gls` the `alldefs` option is the difference between listing every entry of the resource and only the used ones.

```
\edef\@regulatory@bib@selection{}
\ifregulatory@alldefs
\edef\@regulatory@bib@selection{,selection=all}
\fi

\newcommand\loadglsdefs[1]{%
  \setboolean{regulatory@defslisted}{true}%
  \ifregulatory@bibtogls%
    \GlsXtrLoadResources[type=definitions,src={#1},sort={nl-NL},category={defs}\@regulatory@bib@selection}%
  \else%
    \setboolean{regulatory@indexeddefs}{true}%
    \loadglsentries[definitions]{#1}%
  \fi%
}
```

`\newdefinition` Declares one definition in the document itself, under the definitions type and the `defs` category, so that it lands in the same list as the ones read from a file. This is the route that needs no second file and no second program: the entries are printed in the order

they are declared in.

```
\newcommand\newdefinition[3]{%
  \setboolean{regulatory@defslisted}{true}%
  \newglossaryentry{#1}{type=definitions,category=defs,name={#2},description={#3}}%
}
```

Only the indexed route asks for a glossary to be made, which is what draws in `makeindex` or `makeglossaries`; declaring the entries in the document or reading them with `bib2gls` does not. With the `alldefs` option every unused definition is added at the end of the document, so that it shows up in the list as well.

The kernel hook rather than `\AtEndPreamble` of `etoolbox`, which is what this was and which cost a `xpatch` in the prerequisites for one call. Both fire in the same place; the hook is the one that is there without asking for it.

```
\AddToHook{begindocument/before}{%
  \ifthenelse{\boolean{regulatory@indexeddefs}}{\makeglossaries}{}%
}
\AtEndDocument{%
  \ifregulatory@alldefs\glsaddallunused\fi%
}
```

14.3 regulatory-ref

14.3.1 Vereisten

The structures of `regulatory-struct` hold the counters and the reference properties every reference is built upon, so this module loads it first. `xstring` is used to take apart the comma separated label lists of `\aref`.

Options are administered by `regulatory-struct` for the whole bundle, so they are handed over, just like `regulatory` does. Whenever that module is loaded already, its options are settled and there is nothing left to hand over.

```
\@ifpackageloaded{regulatory-struct}{}{%
  \DeclareOption*{\PassOptionsToPackage{CurrentOption}{regulatory-struct}}%
  \ProcessOptions\relax
}
\RequirePackage{regulatory-struct}
\RequirePackage{xstring}
```

The range conjunction and the conjunction of an enumeration are looked up with translations as well, so that they follow the language selected at that very spot in the document.

```
\DeclareTranslationFallback{and}{and}
\DeclareTranslationFallback{to}{to}
```

14.3.2 Schakelaars

`\hashchar` Hyperlinks to another document are composed by hand, for which the hash character is needed with its ordinary category code.

```
{\catcode\#=12 \gdef\hashchar{#}}
\def\rref@linkpr{\href{\rref@current@fullurl}}
```

`\@ifrrefcap` A capitalized reference is requested by the `\Rref`, `\Nref` and `\Aref` variants. Only the first designation printed may be capitalized, hence the switch resets itself as soon as it produced something.

```
\newboolean{rref@cap}
```

A designation that is not built from a language file never passes through here — an article is referred to by its number alone — so the request would otherwise survive the reference that made it and capitalize the next one instead. Every reference ends by withdrawing it.

```
\newcommand\rref@capreset{\setboolean{rref@cap}{false}}
\def\@ifrrefcap#1#2{%
  \ifthenelse{\boolean{rref@cap}}{%
    #1%
    \ifthenelse{\equal{#1}{}}{}{}%
    \setboolean{rref@cap}{false}%
  }%
}{%
  #2%
}%
}
```

`\@ifnameinlink` Reads the `nameinlink` option of `regulatory-struct`, which decides whether the designation is part of the hyperlink or only the number is.

```
\def\@ifnameinlink#1#2{%
  \ifthenelse{\boolean{regulatory@nameinlink}}{#1}{#2}%
}
```

14.3.3 Conjunction

`\setmiddleconjunction` The conjunctions join the labels of an enumeration, of which the last one is joined differently, and a range of three or more consecutive labels is joined with the range conjunction instead.

`\setlastconjunction`

`\setrangeconjunction`

```
\setconjunction
\newcommand\rref@middleconjunction{, }
\newcommand\rref@lastconjunction{ \GetTranslation{and} }
\newcommand\rref@rangeconjunction{ \GetTranslation{to} }

\newcommand\setmiddleconjunction[1]{\renewcommand\rref@middleconjunction{#1}}
\newcommand\setlastconjunction[1]{\renewcommand\rref@lastconjunction{#1}}
\newcommand\setrangeconjunction[1]{\renewcommand\rref@rangeconjunction{#1}}

\newcommand\setconjunction[3]{%
  \setmiddleconjunction{#1}%
  \setlastconjunction{#2}%
  \setrangeconjunction{#3}%
}
```

`\setjuncto` Legacy documents join the last label with ‘junctis’, which the `legacy` option switches on
`\unsetjuncto` right away.

```
\newcommand\setjuncto{%
\setlastconjunction{ jo.\ }%
}

\newcommand\unsetjuncto{%
\setlastconjunction{ \GetTranslation{and} }%
}

\ifregulatory@legacy
\setjuncto
\fi
```

14.3.4 Verwijsformaten

These are the formats the language definition files pick from. A `ref` format turns the number of a reference into its representation, a `label` format decides on the order of the designation and the number, and a `group` format wraps the parent parts of a reference. A point is lettered and put in brackets, (a), which is the form the European style guides write it in and the one the English configuration reaches for.

```
\newcommand\rref@reformat@noop[1]{#1}
\newcommand\rref@reformat@parenthesis[1]{(#1)}
\newcommand\rref@reformat@ordinal[1]{\@ifrrefcap{\Ordinalstringnum{#1}}{\ordinalstringnum{#1}}}
\newcommand\rref@reformat@alpha[1]{\@ifrrefcap{\ABAlphnum{#1}}{\abalphnum{#1}}}
\newcommand\rref@reformat@alphaparen[1]{(\abalphnum{#1})}
\newcommand\rref@label@prefix[2]{#1 #2}
\newcommand\rref@label@postfix[2]{#2 #1}
\newcommand\rref@label@refonly[2]{\@gobble{#1}#2}
\newcommand\rref@group@braced[1]{\{[{}#1{]-}}
```

14.3.5 Taalconfiguratie

Every language gets its own designations per type of reference, of which the initial values are the English ones, so a language definition file only has to state the differences. They are kept in one property list, keyed by language, type and key; the keys a definition file may set are listed, so that a misspelling is refused where it is written rather than read back as nothing wherever it is used.

```
\ExplSyntaxOn
\prop_new:N \g__regulatory_rref_prop
\cs_generate_variant:Nn \prop_item:Nn { Ne }
\cs_generate_variant:Nn \prop_gput:Nnn { Nen }
\cs_generate_variant:Nn \prop_get:NnNF { NeNF }

\clist_const:Nn \c__regulatory_rref_keys_clist
{
  name , Name , names , Names ,
  ref-format , label-format , group-conjunction , group-format
}
```

```

}

\cs_new_protected:Npn \__regulatory_rref_put:nnnn #1#2#3#4
{ \prop_gput:Nen \g__regulatory_rref_prop { #1 / #2 / #3 } { #4 } }

\cs_new_protected:Npn \__regulatory_rref_set:nnnn #1#2#3#4
{
  \clist_if_in:NnTF \c__regulatory_rref_keys_clist { #3 }
    { \__regulatory_rref_put:nnnn { #1 } { #2 } { #3 } { #4 } }
    { \msg_error:nnnnn { regulatory-ref } { unknown-key } { #1 } { #2 } { #3 } }
}

\cs_new_protected:Npn \__regulatory_rref_bare:nnn #1#2#3
{ \msg_error:nnnnn { regulatory-ref } { key-without-value } { #1 } { #2 } { #3 } }

\msg_new:nnn { regulatory-ref } { unknown-key }
{
  The~#2~designations~of~`#1'~were~given~a~key~`#3',~which~this~package~
  does~not~know.~\
  The~keys~are:~\clist_use:Nn \c__regulatory_rref_keys_clist { ,~ }.
}

\msg_new:nnn { regulatory-ref } { key-without-value }
{ The~key~`#3'~of~the~#2~designations~of~`#1'~was~given~no~value. }

\cs_new_protected:Npn \rref@setup@lang #1#2#3
{
  \keyval_parse:nnn
    { \__regulatory_rref_bare:nnn { #1 } { #2 } }
    { \__regulatory_rref_set:nnnn { #1 } { #2 } }
    { #3 }
}
\ExplSyntaxOff

```

```

\rref@init@lang@article
\rref@init@lang@para
\rref@init@lang@sub
\newcommand\rref@init@lang@article[1]{%
  \rref@setup@lang{#1}{article}{%
    name=\GetTranslation{article},
    Name=\GetTranslation{Article},
    names=\GetTranslation{articles},
    Names=\GetTranslation{Articles},
    ref format=\rref@refformat@noop,
    label format=\rref@label@prefix,
    group conjunction={},
    group format=\rref@refformat@noop
  }%
}
\newcommand\rref@init@lang@para[1]{%
  \rref@setup@lang{#1}{para}{%

```

```

        name=\GetTranslation{paragraph},
        Name=\GetTranslation{Paragraph},
        names=\GetTranslation{paragraphs},
        Names=\GetTranslation{Paragraphs},
        ref format=\rref@refformat@parenthesis,
        label format=\rref@label@refonly,
        group conjunction={,},
        group format=\rref@refformat@noop
    }%
}
\newcommand\rref@init@lang@sub[1]{%
    \rref@setup@lang{#1}{sub}{%
        name=\GetTranslation{point},
        Name=\GetTranslation{Point},
        names=\GetTranslation{points},
        Names=\GetTranslation{Points},
        ref format=\rref@refformat@ordinal,
        label format=\rref@label@postfix,
        group conjunction={,},
        group format=\rref@group@braced
    }%
}
}

```

`\rref@setup` Declares *⟨language⟩* and configures its three types at once. The declaration itself is recorded, so that `\rref@lang` can tell a supported language from an unsupported one.

This is extension API and not an internal: it is what a language definition file calls, and it is supported as it stands for the whole of 1.x. The at sign is deliberate. A definition file is read with the at sign made a letter — `\RegulatoryLoadLanguage` sets the category code and restores it afterwards — which is the convention the kernel’s own .def files and the `fc-⟨language⟩.def` files of `fmtcount` are written under, and inside such a file the name asks for nothing. A document that declares a language in its own preamble puts the call between `\makeatletter` and `\makeatother`. The format commands it is given — `\rref@refformat@` and `\rref@label@` below, and `\rref@group@braced` — are part of the same interface.

```

\newcommand\rref@setup[4]{%
    \expandafter\gdef\csname rref@declared@#1\endcsname{}%
    \rref@init@lang@article{#1}
    \rref@init@lang@para{#1}
    \rref@init@lang@sub{#1}
    \rref@setup@lang{#1}{article}{#2}
    \rref@setup@lang{#1}{para}{#3}
    \rref@setup@lang{#1}{sub}{#4}
}

```

`\rref@lang` The language a reference is formatted in. Whenever the current language has no configuration of its own, the English one is used, which is always loaded. Note that this expands within `\csname`, so it must not produce a single space.

```

\def\rref@lang{%
  \ifcsname languagename\endcsname%
    \ifcsname rref@declared@\languagename\endcsname\languagename\else english\fi%
  \else english\fi%
}

```

`\rref@get` Reading a single key of the current, or of an explicitly given, language. Where `\rref@get` prints the value, `\rref@set` stores it in the macro of its last argument.

A designation that stands in front of more than one number is written in the plural: the Leidraad voor juridische auteurs writes “de artikelen 162 tot en met 164” and “onderdelen c en d”. The plural is a key of its own per type and per language, since it is not the language that decides it but the position: Dutch keeps “lid” singular where the designation follows the ordinals, and says so in its own definition file rather than here.

```

\ExplSyntaxOn
\NewDocumentCommand \rref@set { O{\rref@lang} m m m }
{
  \prop_get:NenF \g__regulatory_rref_prop { #1 / #2 / #3 } #4
  { \cs_set_eq:NN #4 \relax }
}

\NewExpandableDocumentCommand \rref@get { O{\rref@lang} m m }
{ \prop_item:Ne \g__regulatory_rref_prop { #1 / #2 / #3 } }
\ExplSyntaxOff

\newif\ifrref@plural
\newcommand\rref@name[2][\rref@lang]{%
  \ifrref@plural%
    \ifrrefcap{\rref@get[#1]{#2}{Names}}{\rref@get[#1]{#2}{names}}%
  \else%
    \ifrrefcap{\rref@get[#1]{#2}{Name}}{\rref@get[#1]{#2}{name}}%
  \fi%
}
\def\rref@article@name{\rref@name{article}}
\def\rref@para@name{\rref@name{para}}
\def\rref@sub@name{\rref@name{sub}}

```

The formats of the current type applied to a number. A link format is a label format of which either the whole label or only the number is a hyperlink, depending on the `nameinlink` option.

```

\newcommand\rref@refformat[2][\rref@current@type]{\rref@set{#1}{ref format}{\@@@refformat}\@@@refformat{#2}}
\newcommand\rref@labelformat[2][\rref@current@type]{\rref@set{#1}{label format}{\@@@labelformat}\@@@labelformat{#2}}
\newcommand\rref@linkformat[2][\rref@current@type]{%
  \@@ifnameinlink{%
    \rref@linkpr{\rref@labelformat[#1]{\rref@refformat[#1]{#2}}}%
  }{%
    \rref@labelformat[#1]{\rref@linkpr{\rref@refformat[#1]{#2}}}%
  }%
}
\newcommand\rref@reflink[2][\rref@current@type]{%

```



```

\rrref@linkpr{\rrref@reformat[#1]{#2}}%
}
\newcommand\rrref@groupformat[2][\rrref@current@type]{%
\rrref@set{#1}{group format}{\@@groupformat}\@@groupformat{#2}%
}

```

14.3.6 Verwijsmacro's

`\rrref` Refers with the number only. Whenever the label is not one of the structures of this bundle, the reference is handed over to `zref`, which knows how to refer to sections and the like.

```

\def\rrref{\@ifstar\rrref@star\rrref@nostar}
\def\rrref@star#1{%
\rrref@setcurrent{#1}%
\ifnum\rrref@current@value>0%
\rrref@reformat{\rrref@current@value}%
\else%
\zref{#1}%
\fi%
\rrref@capreset%
}
\def\rrref@nostar#1{%
\rrref@setcurrent{#1}%
\ifnum\rrref@current@value>0%
\rrref@linkpr{\rrref@reformat{\rrref@current@value}}%
\else%
\zceref[noname]{#1}%
\fi%
\rrref@capreset%
}
\def\Rref{\@ifstar\Rref@star\Rref@nostar}
\def\Rref@star#1{%
\setboolean{rref@cap}{true}%
\rrref*{#1}%
}
\def\Rref@nostar#1{%
\setboolean{rref@cap}{true}%
\rrref{#1}%
}

```

`\nref` Refers with the designation and the number, in the order the language prescribes.

```

\Nref \def\nref{\@ifstar\nref@star\nref@nostar}
\def\nref@star#1{%
\rrref@setcurrent{#1}%
\ifnum\rrref@current@value>0%
\rrref@labelformat{\rrref@reformat{\rrref@current@value}}%
\else%
\zceref*{#1}%
\fi%
}

```

```

\rrref@capreset%
}
\def\Nref@nostar#1{%
\rrref@setcurrent{#1}%
\ifnum\rrref@current@value>0%
\rrref@linkformat{\rrref@current@value}%
\else%
\zcref{#1}%
\fi%
\rrref@capreset%
}
\def\Nref{\@ifstar\Nref@star\Nref@nostar}
\def\Nref@star#1{%
\setboolean{rrref@cap}{true}%
\Nref*{#1}%
}
\def\Nref@nostar#1{%
\setboolean{rrref@cap}{true}%
\Nref{#1}%
}

```

14.3.7 De huidige verwijzing

`\rrref@setcurrent` Reads all properties of $\langle label \rangle$ recorded by `regulatory-struct` and works out which type of reference it is.

```

\def\rrref@setcurrent#1{%
\def\rrref@current{#1}%
\def\rrref@current@counter{\zref@extract{#1}{counter}}%
\def\rrref@current@article{\zref@extract{#1}{article}}%
\def\rrref@current@para{\zref@extract{#1}{para}}%
\def\rrref@current@sub{\zref@extract{#1}{sub}}%
\rrref@settype%
\rrref@setlink%
}

```

The counter of a label tells the type. Both the `\para` heading and the first level of the `paras` environment count as a paragraph, and anything else is left to `zref`, which is what the value of -1 stands for.

```

\def\rrref@settype{%
\ifthenelse{\equal{article}{\rrref@current@counter}}{%
\def\rrref@current@type{article}%
\def\rrref@current@value{\rrref@current@article}%
\def\rrref@current@fullformat{}}%
}%
\ifthenelse{\equal{para}{\rrref@current@counter}\or\equal{parasi}{\rrref@current@counter}}{%
\def\rrref@current@type{para}%
\def\rrref@current@value{\rrref@current@para}%
}%
\ifthenelse{\equal{parasii}{\rrref@current@counter}}{%

```

```

\def\rref@current@type{sub}%
\def\rref@current@value{\rref@current@sub}%
}%
\def\rref@current@type{}%
\def\rref@current@value{-1}%
}%
}%
}%
}

```

The anchor of a label within the current document, or the anchor within the URL of another document, which regulatory-attachments records with \refdocument and \masterdocument.

```

\def\rref@setlink{%
\def\rref@current@anchor{\zref@extract{\rref@current}{anchor}}%
\def\rref@current@url{\zref@extractdefault{\rref@current}{url}{}}%
\def\rref@current@fullurl{\rref@current@url\hashchar\rref@current@anchor}%
}

```

\rref@number The number of another label, of the type of the current reference. The sorting and the compaction of an enumeration compare labels with it.

```

\newcommand\rref@number[1]{%
\zref@extractdefault{#1}{\rref@current@type}{-1}%
}

```

14.3.8 Volledige verwijzingen

\aref@postlink@setup Whatever follows a complete reference, which is nothing as long as the label belongs to this document. regulatory-attachments redefines this to mention the document a label of another document belongs to.

```

\newcommand\aref@postlink@setup[1]{%
\def\@aref@postlink{}%
}

```

\aref@setcurrent A complete reference states all parent parts of the label as well, so a paragraph is prefixed with its article, and a point with both its article and its paragraph. Each of them is joined with the group conjunction of its own type.

```

\newcommand\aref@setcurrent[1]{%
\def\aref@current@reflink{\rref@reflink[\rref@current@type]{\rref@current@value}}%
\aref@postlink@setup{#1}%
\ifthenelse{\equal{\rref@current@type}{article}}{%
\def\@aref@prefix{}%
\def\@aref@postfix{}%
}%
\ifthenelse{\equal{\rref@current@type}{para}}{%
\def\@aref@prefix{%
\rref@labelformat[article]{\rref@current@article}\rref@get{article}{group conjunction}%
}%
}

```

```

\def\@aref@postfix{}%
}{%
\ifthenelse{\equal{\rref@current@type}{sub}}{%
\def\@aref@prefix{%
\rref@labelformat[article]{\rref@refformat[article]{\rref@current@article}}\rref@get{article}
\rref@labelformat[para]{\rref@refformat[para]{\rref@current@para}}\rref@get{para}{group con
}%
\def\@aref@postfix{}%
}{%
\def\@aref@prefix{%
\def\@aref@postfix{}%
}%
}%
}

```

`\rref@labellist` Reads an enumeration of labels into `\rref@labels`. A space is what a writer naturally puts after a comma, and it used to be fatal in a way that said nothing: the number of a label with a leading space cannot be extracted, the default of that extraction is -1 , and -1 is the terminator of the enumeration, so the list ended silently at the first space and every label behind it was dropped.

The list is therefore split on the commas and each label is trimmed on its own. Taking every space out of the whole list would be shorter and wrong: a label may perfectly well have a space inside it, and a document that writes `\aref{lid:meerwerk niet akkoord}` would then be looking for a label nobody ever declared, which is exactly the kind of silent miss this is meant to end.

```

\ExplSyntaxOn
\tl_new:N \rref@labels
\seq_new:N \__regulatory_ref_labels_seq

\cs_new_protected:Npn \__regulatory_ref_append:n #1
{
\tl_put_right:Nn \rref@labels { #1 }
}

\cs_new_protected:Npn \rref@labellist #1
{
\seq_set_split:Nnn \__regulatory_ref_labels_seq { , } { #1 }
\tl_clear:N \rref@labels
\seq_map_inline:Nn \__regulatory_ref_labels_seq
{
\tl_if_empty:NF \rref@labels { \tl_put_right:Nn \rref@labels { , } }
\tl_trim_spaces_apply:nN { ##1 } \__regulatory_ref_append:n
}
}
\ExplSyntaxOff

```

`\aref` A complete reference of one or more labels. All labels of an enumeration are of the same
`\Aref`

type, so the first one determines the prefix and the designation of the whole group.

```

\newcommand{\aref}[1]{%
  \rref@labellist{#1}%
  \def\@@arefsetup##1##2{%
    \rref@setcurrent{##1}%
    \aref@setcurrent{##1}%
    \rref@groupformat{\@aref@prefix}%
    ##2%
    \@aref@postfix\@aref@postlink%
  }%
  \expandafter\aref@dispatch\expandafter{\rref@labels}%
  \rref@capreset%
}

\newcommand\aref@dispatch[1]{%
  \IfSubStr{#1}{,}{,%}{%
    \StrBefore[1]{#1}{,}{\firstarg}%
    \@@arefsetup{\firstarg}{%
      \rref@pluraltrue\rref@labelformat{\aref@group{#1}}\rref@pluralfalse%
    }%
  }{%
    \@@arefsetup{#1}{\nref{#1}}%
  }%
}

\newcommand{\Aref}[1]{%
  \setboolean{rref@cap}{true}%
  \aref{#1}%
}

```

14.3.9 Opsommingen

An enumeration of labels is first sorted and then compacted, so that three or more consecutive numbers become a range. Both walk the comma separated list with a terminator at the end.

```

\def\listterminator{-1}

\newcommand\aref@group[1]{%
  \bubblesort{#1}%
  \expandafter\begincompaction\sortedlist,\listterminator,\relax%
}

```

`\begincompaction` The compaction keeps track of the first label of the current run and of the last consecutive one found so far. As soon as the run is broken, it is printed as a single reference, as a pair joined with a conjunction, or as a range, after which the remainder of the list is compacted the same way.

```

\def\begincompaction#1,#2\relax{%
  \def\startlist{#1}%

```

```

\def\currentendlist{#1}%
\findendlist#2\relax%
}
\def\findendlist#1,#2\relax{%
\ifnum\numexpr\rref@number{\currentendlist}+1\relax=\rref@number{#1}\relax%
\def\currentendlist{#1}%
\findendlist#2\relax%
\else%
\ifnum\rref@number{\startlist}=\rref@number{\currentendlist}\relax%
\ignorespaces\rref{\startlist}\unskip%
\else%
\ifnum\numexpr\rref@number{\startlist}+1\relax=\rref@number{\currentendlist}\relax%
\ignorespaces\rref{\startlist}\unskip%
\ifnum\rref@number{#1}=\listterminator\relax%
\rref@lastconjunction%
\else%
\rref@middleconjunction%
\fi\ignorespaces%
\rref{\currentendlist}\unskip%
\else%
\ignorespaces\rref{\startlist}\unskip\rref@rangeconjunction\ignorespaces%
\rref{\currentendlist}\unskip%
\fi%
\fi%
\ifnum\rref@number{#1}=\listterminator\else%
\IfStrEq{#2}{\listterminator,}{\rref@lastconjunction}{\rref@middleconjunction}\begincompaction#1,#2
\fi%
\fi%
}

```

\bubblesort Sorting is done on the number of the type of the current reference, by swapping the first two labels that are out of order and starting over with the result.

```

\newcommand\bubblesort[1]{\def\sortedlist{}\sortlist#1,\listterminator,\relax}
\def\sortlist#1,#2,#3\relax{%
\rref@setcurrent{#1}%
\aref@setcurrent{#1}%
\ifnum\rref@number{#2}=\listterminator\relax%
\edef\sortedlist{\sortedlist#1}%
\else%
\ifnum\rref@number{#1}<\rref@number{#2}\relax%
\edef\sortedlist{\sortedlist#1,}%
\sortlist#2,#3\relax%
\else%
\let\tmp\sortedlist
\def\sortedlist{}%
\expandafter\sortlist\tmp#2,#1,#3\relax%
\fi%
\fi%
}

```

14.4 regulatory-attachments

14.4.1 Vereisten

A reference to another document is a reference first, so this module builds on `regulatory-ref`, which loads `regulatory-struct` in turn. The definitions of another document land in a glossary of their own, which is why `regulatory-defs` is needed as well. Both `xr-hyper` and `zref-xr` are already loaded by `regulatory-struct`, since they have to be loaded before `hyperref`. Options are handed over to `regulatory-struct`, which administers them for the whole bundle.

```
\@ifpackageloaded{regulatory-struct}{}{%  
  \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{regulatory-struct}}%  
  \ProcessOptions\relax  
}  
\RequirePackage{regulatory-ref}  
\RequirePackage{regulatory-defs}  
  
\DeclareTranslationFallback{of the}{of the}  
\DeclareTranslationFallback{see}{see}
```

Definitions of other documents are kept in a glossary type of their own, so they don't get mixed up with the definitions of the document itself.

```
\newglossary*{externals}{Externals}
```

14.4.2 Artefacten

`\newartifact` An artifact holds everything known about another document: where its file is, how it is referred to, and what a PDF viewer should say about it once it is attached. The artifacts of a document live in one property list, keyed by *name* and field, and the fields a document may give are listed rather than left open: a field this package does not know is a misspelling of one that it does, and it used to be stored under the wrong name and read back as nothing.

Two of the fields are not for a document to set, and they are held apart from the ones that are. They are accepted, because a document written against an earlier release may give one and stopping such a document would be the worse of the two errors, and they are left out of the list the error message offers, because that list is read as an invitation.

`referred` is set by `\documentfootnote` to record that an artifact has had its footnote, and a document that sets it by hand suppresses a footnote it never saw. `url` is reserved: it is stored and readable with `\artifacturl`, and nothing in this bundle reads it. It was meant to name where the document may be found, in the footnote beside the attachment, and that is not implemented. It is kept rather than removed so that the name stays free for exactly that, and it is documented as reserved rather than as an option, since an option that does nothing is a promise.

```
\newcommand\artifact@footnote@noop[1]{#1}  
\newcommand\artifact@def@label@default[1]{~(\GetTranslation{see} #1)}  
  
\ExplSyntaxOn
```

```

\prop_new:N \g__regulatory_artifact_prop
\cs_generate_variant:Nn \prop_item:Nn { Ne }
\cs_generate_variant:Nn \prop_gput:Nnn { Nen }
\cs_generate_variant:Nn \prop_get:NnNF { NeNF }
\cs_generate_variant:Nn \prop_if_in:NnTF { NeTF }

\clist_const:Nn \c__regulatory_artifact_keys_clist
{
  name , filename , path , ref~label , def~label , footnote , footnote~label ,
  defs , author , subject , description
}

\clist_const:Nn \c__regulatory_artifact_reserved_clist { url , referred }

\cs_new_protected:Npn \__regulatory_artifact_put:nnn #1#2#3
{ \prop_gput:Nen \g__regulatory_artifact_prop { #1 / #2 } { #3 } }

\cs_new_protected:Npn \__regulatory_artifact_set:nnn #1#2#3
{
  \clist_if_in:NnTF \c__regulatory_artifact_keys_clist { #2 }
  { \__regulatory_artifact_put:nnn { #1 } { #2 } { #3 } }
  {
    \clist_if_in:NnTF \c__regulatory_artifact_reserved_clist { #2 }
    { \__regulatory_artifact_put:nnn { #1 } { #2 } { #3 } }
    {
      \msg_error:nnnn { regulatory-attachments } { unknown-field }
      { #1 } { #2 }
    }
  }
}

\cs_new_protected:Npn \__regulatory_artifact_bare:nn #1#2
{ \msg_error:nnnn { regulatory-attachments } { field-without-value } { #1 } { #2 } }

\msg_new:nnn { regulatory-attachments } { unknown-field }
{
  The~artifact~`#1'~was~given~a~field~`#2',~which~this~package~does~not~know.~\\
  The~fields~are:~\clist_use:Nn \c__regulatory_artifact_keys_clist { ,~ }.
}

\msg_new:nnn { regulatory-attachments } { field-without-value }
{ The~field~`#2'~of~artifact~`#1'~was~given~no~value. }

\NewDocumentCommand \newartifact { m m }
{
  \__regulatory_artifact_put:nnn { #1 } { name } { #1 }
  \__regulatory_artifact_put:nnn { #1 } { filename } { #1.pdf }
  \__regulatory_artifact_put:nnn { #1 } { path } { ./ }
  \__regulatory_artifact_put:nnn { #1 } { ref~label } { }
  \__regulatory_artifact_put:nnn { #1 } { def~label } { \artifact@def@label@default }
}

```



```

\__regulatory_artifact_put:nnn { #1 } { footnote } { true }
\__regulatory_artifact_put:nnn { #1 } { footnote-label } { \artifact@footnote@noop }
\__regulatory_artifact_put:nnn { #1 } { url } { }
\__regulatory_artifact_put:nnn { #1 } { referred } { false }
\__regulatory_artifact_put:nnn { #1 } { defs } { }
\__regulatory_artifact_put:nnn { #1 } { author } { <author> }
\__regulatory_artifact_put:nnn { #1 } { subject } { <subject> }
\__regulatory_artifact_put:nnn { #1 } { description } { <description> }
\keyval_parse:nnn
{ \__regulatory_artifact_bare:nn { #1 } }
{ \__regulatory_artifact_set:nnn { #1 } }
{ #2 }
}

\cs_new_protected:Npn \artifact@ifknown #1
{ \prop_if_in:NeTF \g__regulatory_artifact_prop { #1 / name } }

\cs_new_protected:Npn \artifact@ifuncomposable #1
{ \prop_if_in:NeTF \g__regulatory_artifact_prop { #1 / uncomposable } }

\cs_new_protected:Npn \artifact@setuncomposable #1
{ \__regulatory_artifact_put:nnn { #1 } { uncomposable } { true } }

\cs_new_protected:Npn \artifact@setreferred #1
{ \__regulatory_artifact_put:nnn { #1 } { referred } { true } }
\ExplSyntaxOff

```

The accessors of an artifact, of which the full name and the full filename are the ones prefixed with the path of the artifact.

```

\ExplSyntaxOn
\cs_new:Npn \@aget #1#2 { \prop_item:Ne \g__regulatory_artifact_prop { #1 / #2 } }
\cs_new_protected:Npn \artifact@setfootnotelabel #1
{
  \prop_get:NeNF \g__regulatory_artifact_prop { #1 / footnote-label }
  \artifactfootnotelabel
  { \cs_set_eq:NN \artifactfootnotelabel \artifact@footnote@noop }
}
\ExplSyntaxOff
\newcommand\artifactname[1]{\@aget{#1}{name}}
\newcommand\artifactfullname[1]{\@aget{#1}{path}\@aget{#1}{name}}
\newcommand\artifactfilename[1]{\@aget{#1}{filename}}
\newcommand\artifactfullfilename[1]{\@aget{#1}{path}\@aget{#1}{filename}}
\newcommand\artifactfootnote[1]{\@aget{#1}{footnote}}

```

The reader of the reserved `url` field. It gives back what was stored and nothing else reads it; see `\newartifact` above.

```

\newcommand\artifacturl[1]{\@aget{#1}{url}}
\newcommand\artifactreferred[1]{\@aget{#1}{referred}}
\newcommand\artifactdefs[1]{\@aget{#1}{defs}}

```

```

\newcommand\artifactauthor[1]{\@aget{#1}{author}}
\newcommand\artifactssubject[1]{\@aget{#1}{subject}}
\newcommand\artifactdescription[1]{\@aget{#1}{description}}

```

14.4.3 Bestanden invoegen

\regulatory@embed Embeds the file of #1 under the name #2, as an object named #3 with description #4. The file is registered as an associated file of the document with relationship /Source, and it is added to the embedded files of the catalog, so that PDF viewers list it as an attachment.

```

\ExplSyntaxOn
\str_new:N \l__regulatory_desc_str
\str_new:N \l__regulatory_ext_str

```

PDF/A-3 requires the MIME type of every embedded file, and prescribes application/octet-stream whenever it is unknown. l3pdf looks the type up by extension and only warns when it finds none, so the prescribed fallback is filled in here.

```

\cs_new_protected:Npn \__regulatory_mimetype:n #1
{
  \file_parse_full_name:nnn {#1}
  \l_tmpa_str \l_tmpb_str \l__regulatory_ext_str
  \prop_if_in:NVF \g_pdfffile_mimetypes_prop \l__regulatory_ext_str
  {
    \pdfdict_put:nne {\l_pdfffile}{Subtype}
    { \pdf_name_from_unicode:e:n {application/octet-stream} }
  }
}

\cs_new_protected:Npn \__regulatory_embed:nnnn #1#2#3#4
{
  \group_begin:
  \pdfdict_put:nnn {\l_pdfffile/Filespec}{AFRelationship}{/Source}
  \__regulatory_mimetype:n {#1}
  \tl_if_empty:nF {#4}
  {
    \pdf_string_from_unicode:nnN {utf16/string}{#4}\l__regulatory_desc_str
    \pdfdict_put:nne {\l_pdfffile/Filespec}{Desc}{\l__regulatory_desc_str}
  }
  \pdfffile_embed_file:nnn {#1}{#2}{regulatory/attach/#3}
  \pdfmanagement_add:nne {Catalog/Names/EmbeddedFiles}
  {#3}{\pdf_object_ref:n{regulatory/attach/#3}}
  \pdfmanagement_add:nne {Catalog}{AF}
  {\pdf_object_ref:n{regulatory/attach/#3}}
  \group_end:
}

```

The description arrives expanded, all four arguments alike. The string conversion of l3pdf assumes the purifying step has already been done, and says so in its own source, so a description handed over unexpanded is written into the PDF as the tokens it is made of:

every attachment used to carry `\artifactdescription{<label>}` as its description rather than the description itself, which is what a viewer shows beside the file in its attachment pane.

```
\cs_generate_variant:Nn \__regulatory_embed:nnnn {eeee}
\cs_new_protected:Npn \regulatory@embed #1#2#3#4
{ \__regulatory_embed:eeee {#1}{#2}{#3}{#4} }
```

`\regulatory@attach@annot` The other route of attachmentlink: a file attachment annotation of exactly the size of #2, with #2 drawn over it, so that the text is what the reader sees and the annotation is what the reader opens. `\pdfannot_box:nnnn` advances by nothing of its own, so the text that follows lands exactly where the annotation is. The appearance is the empty one, since the page already draws everything there is to see; an appearance there has to be, because the A-standards ask one of every annotation that is not a Popup, a Link or of zero size. The `/Contents` is the description the file was embedded with, which is what a screen reader announces.

```
\box_new:N \l__regulatory_attach_box
\tl_new:N \l__regulatory_attach_desc_tl
\str_new:N \l__regulatory_attach_desc_str
\cs_new:Npn \regulatory@attach@appearance { }
\bool_lazy_and:nnT
{ \cs_if_exist_p:N \pdfmanagement_if_active_p: }
{ \cs_if_exist_p:N \pdf_object_new:n }
{
  \pdfmanagement_if_active:T
  {
    \pdf_object_new:n { regulatory/attach/blank }
    \AddToHook { shipout/firstpage }
    {
      \pdf_object_write:nnn { regulatory/attach/blank } { stream }
      { { /Type~/XObject /Subtype~/Form /BBox~[0~0~1~1] } { } }
    }
    \cs_gset:Npn \regulatory@attach@appearance
    { /AP~<<~/N~\pdf_object_ref:n { regulatory/attach/blank }~>> }
  }
}
\cs_new_protected:Npn \regulatory@attach@place #1#2
{
  \pdfannot_box:nnne
  { \box_wd:N \l__regulatory_attach_box }
  { \box_ht:N \l__regulatory_attach_box }
  { \box_dp:N \l__regulatory_attach_box }
  {
    /Subtype~/FileAttachment
    /FS~\pdf_object_ref:n { regulatory/attach/ #1 }
    /F~4
    \tl_if_empty:NF \l__regulatory_attach_desc_tl
    { /Contents~\l__regulatory_attach_desc_str }
    \regulatory@attach@appearance
  }
}
```

```

        #2
    }
}
\cs_new_protected:Npn \regulatory@attach@annot #1#2
{
    \tl_set:Nx \l__regulatory_attach_desc_tl
    { \cs_if_exist_use:cF { regulatory@desc@ #1 } { } }
    \pdf_string_from_unicode:nVN {utf16/string}
    \l__regulatory_attach_desc_tl \l__regulatory_attach_desc_str
    \hbox_set:Nn \l__regulatory_attach_box { #2 }
    \mode_leave_vertical:
    \cs_if_exist:NTF \tag_if_active:TF
    { \tag_if_active:TF { \regulatory@attach@tagged {#1} }
      { \regulatory@attach@place {#1} { } } }
    { \regulatory@attach@place {#1} { } }
    \box_use:N \l__regulatory_attach_box
}
\cs_new_protected:Npn \regulatory@attach@tagged #1
{
    \tag_mc_end_push:
    \tag_struct_begin:n { tag=Annot }
    \regulatory@attach@place {#1}
    { /StructParent~\tag_struct_parent_int: }
    \exp_args:Nee
    \tag_struct_insert_annot:nn
    { \pdfannot_ref_last: } { \tag_struct_parent_int: }
    \tag_struct_end:
    \tag_mc_begin_pop:n {}
}

```

`\regulatory@attachmentlink` Turns #2 into whatever `attachmentlink` says: a link that opens the embedded file named #1 in a new window, or the annotation above.

For the link, the destination is the first page of the target, and it is not optional: an embedded go-to action without a `/D` is not one, and a viewer that reads it has nowhere to go. Not every viewer goes there even so — `GoToE` is the one file-opening action poppler does not implement, so in the viewers built on it the file is reachable through the attachment pane and not through this link. That is what the other route is for.

The link also carries the file in its `/AF`. An action is something a consumer has to perform to learn what it points at; an associated file is something it can read. This route names the file indirectly, by a key in the name tree of the document, so the entry says declaratively what the action says only when it is followed. The other route needs none: a file attachment annotation names its file in `/FS` already, and an `/AF` beside it would say the same thing twice.

```

\NewDocumentCommand \regulatory@attachmentlink { m m }
{
    \str_if_eq:VnTF \regulatory@attachkind { annotation }
    { \regulatory@attach@annot {#1} {#2} }
    {

```

```

\pdfannot_link_begin:nnw {user}
{
  /Subtype/Link/F~4
  /AF~[~\pdf_object_ref:n { regulatory/attach/ #1 }~]
  /A<</S/GoToE/D~[0~/Fit]/NewWindow-true/T<</R/C/N~(#1)>>>>
}
#2
\pdfannot_link_end:n {user}
}
}

```

`\regulatory@ifembed` Embedding a file is only possible while the PDF management of $\text{\LaTeX}$ is active, which a document activates with `\DocumentMetadata` in front of its `\documentclass`. Without it, the file embedding commands of `l3pdf` do not even exist, so every attachment falls back to its plain text and the document is told once what it is missing.

```

\cs_if_exist:NTF \pdffile_embed_file:nnn
{ \cs_new_eq:NN \regulatory@ifembed \use_i:nn }
{ \cs_new_eq:NN \regulatory@ifembed \use_ii:nn }
\ExplSyntaxOff

\regulatory@ifembed{}{%
  \PackageWarning{regulatory}{%
    Attachments need the PDF management of LaTeX.\MessageBreak
    Put \string\DocumentMetadata{} before \string\documentclass\MessageBreak
    to activate it. Attachments are typeset as plain text%
  }%
}

```

`\regulatory@embedonce` A document referred to more than once is embedded only the first time, of which the object name is kept for all following links.

```

\newcommand\regulatory@embedonce[1]{%
  \expandafter\ifx\csname regulatory@embedded@#1\endcsname\relax
    \regulatory@embed
    {\artifactfullfilename{#1}}%
    {\artifactfilename{#1}}%
    {regulatory-#1}%
    {\artifactdescription{#1}}%
  \expandafter\xdef\csname regulatory@desc@regulatory-#1\endcsname
    {\artifactdescription{#1}}%
  \expandafter\xdef\csname regulatory@embedded@#1\endcsname{regulatory-#1}%
  \fi
}

```

`\documentattachment` Attaches the file of artifact $\langle label \rangle$ and prints $\langle link\ text \rangle$ as the link to it. A missing file is no reason to fail; its link text is printed verbatim instead.

```

\newcommand\documentattachment[2]{%
  \def\tmpfile{\artifactfullfilename{#1}}%
  \regulatory@ifembed{%

```

```

\IfFileExists{\@tmpfile}{%
  \regulatory@embedonce{#1}%
  \expandafter\expandafter\expandafter\regulatory@attachmentlink
  \expandafter\expandafter\expandafter
  {\csname regulatory@embedded@#1\endcsname}{#2}%
}%
{\texttt{#2}}%
}{\texttt{#2}}%
}

```

`\attachdocument` Attaches any file by its *filename*, without an artifact of its own, with *description* as the description of the attachment.

```

\newcommand\attachdocument[3][]{%
  \regulatory@ifembed{%
    \IfFileExists{#2}{%
      \expandafter\ifx\csname regulatory@plain@#2\endcsname\relax
        \regulatory@embed {#2}{#2}{regulatory-#2}{#1}%
        \expandafter\xdef\csname regulatory@desc@regulatory-#2\endcsname{#1}%
        \expandafter\xdef\csname regulatory@plain@#2\endcsname{regulatory-#2}%
      \fi
      \expandafter\expandafter\expandafter\regulatory@attachmentlink
      \expandafter\expandafter\expandafter
      {\csname regulatory@plain@#2\endcsname}{#3}%
    }%
    {\texttt{#3}}%
  }{\texttt{#3}}%
}

```

14.4.4 Bronvermelding

`\documentlabel` How a reference mentions the document it belongs to, which defaults to the subject of the artifact with a connective before it: “of the Example”, “van de Voorbeeldakte”.

That default only works in a language that leaves a noun alone. German does not, and it is the case that settles the design. Its connective is a genitive that reshapes the word it governs: measured in the German text of the General Data Protection Regulation, “der Verordnung” sixteen times and “der Richtlinie” twenty-nine, but “des Vertrags” and “des Beschlusses” — not *des Vertrag*. So the ending belongs to the noun and not to the connective, and no rule over a connective and a subject in the nominative can produce the phrase. Handing the artifact a determiner would not do it either; the determiner is the half that is easy.

What is left is to state the phrase, which `ref label` does and always did. A language that cannot compose one therefore declares its connective empty, and a reference that then has no `ref label` to fall back on says so once per artifact rather than quietly wording it in another language.

```

\newcommand\documentlabel[1]{%
  \def\@@label{\@aget{#1}{ref label}}%
  \ifthenelse{\equal{\@@label}{}}{%

```

```

\edef\@@of{\GetTranslation{of the}}%
\ifx\@@of\@empty
  \artifact@warnuncomposable{#1}%
  \artifactssubject{#1}%
\else
  \@@of\space\artifactssubject{#1}%
\fi
}{\@@label}%
}

\newcommand\artifact@warnuncomposable[1]{%
  \artifact@ifuncomposable{#1}{}{%
    \artifact@setuncomposable{#1}%
    \PackageWarning{regulatory}{%
      A reference to `#1' is worded in \language\space by naming its\MessageBreak
      subject alone: that language inflects the noun a connective\MessageBreak
      governs, so this bundle cannot build the phrase. Give the\MessageBreak
      artifact a `ref label' with the phrase in it. Reported}%
    }%
  }
}

```

`\documentfootnote` Places the footnote of the first occurrence of a reference or a definition of another document, with the document itself attached to it. The artifact is marked as referred, so that following occurrences don't repeat the footnote.

```

\newcommand\documentfootnote[2][{}]{%
  \artifact@ifknown{#2}{%
    \artifact@setreferred{#2}%
    \artifact@setfootnotelabel{#2}%
    \ifthenelse{\equal{#1}{} }{%
      \def\artifact@description{\artifactssubject{#2}}%
    }{%
      \def\artifact@description{#1}%
    }%
    \footnote{\artifactfootnotelabel{\documentattachment{#2}{\artifact@description}}}%
  }{\PackageWarning{regulatory}{Artifact does not exists '#2'}\footnote{DOCUMENT NOT FOUND '#2'}}
}

```

`\aref@postlink@setup` This is where a complete reference of regulatory-ref gets its source mentioned. Labels of the document itself carry no externaldocument property, in which case nothing is added at all.

```

\renewcommand\aref@postlink@setup[1]{%
  \zref@ifrefcontainsprop{#1}{externaldocument}{%
    \filename@parse{\expanded{\zref@extract{#1}{externaldocument}}}%
    \edef\@aref@filelabel{\expanded{\filename@base}}%
    \def\@aref@postlink{%
      , \documentlabel{\@aref@filelabel}%
      \ifthenelse{\equal{\artifactfootnote{\@aref@filelabel}}{true}}{\and\equal{\artifactreferred{\@aref@filelabel}}{\documentfootnote{\@aref@filelabel}}}%
    }{}%
  }
}

```

```

    }%
  }{%
    \def\@aref@postlink{%
  }%
}

```

14.4.5 Documenten koppelen

`\loadextdefs` Loads the definitions of $\langle src \rangle$ under the externals type and the category $\langle category \rangle$, which is the name of the artifact they belong to. Every first occurrence of such a definition mentions its source, unless the document was already referred to.

```

\newcommand\loadextdefs[2][externals]{%
  \ifregulatory@bibtogls%
    \GlsXtrLoadResources[type=externals,src={#2},sort={nl-NL},category={#1}]%
  \else%
    \loadglsentries[externals]{#2}%
  \fi%
  \glsdefpostlink{#1}{%
    \ifthenelse{\equal{\artifactreferred{#1}}{true}}{}{%
      \@aget{#1}{def label}{%
        \artifactsubject{#1}%
        \documentfootnote{#1}%
      }%
    }%
  }%
}

```

`\regulatory@extdefs@nohyper` A definition of another document is printed in that other document, not in this one, so the anchor its citation would point at is not here. `glossaries` links to it all the same, and the result is a link that goes nowhere: measured on the examples, the PDF of the second one holds a link to `glo: def1` and no destination by that name, and the HTML of it holds an `href` to a fragment that lives in the other file.

The hyperlink is therefore withdrawn for the externals type, in the hook `glossaries` offers for exactly this, after the options of a citation are set and before it is typeset. The citation itself stays, and so does the mention of the document it comes from, which is what tells a reader where to look — the same thing `regulatory-html` does for a reference to a provision of another document.

```

\def\regulatory@externalstype{externals}
\renewcommand*\glslinkpostsetkeys{%
  \edef\regulatory@thisdeftype{\glsentrytype\glslabel}%
  \ifx\regulatory@thisdeftype\regulatory@externalstype%
    \setkeys{glslink}{hyper=false}%
  \fi%
}

```

`\refdocument` Declares another regulatory document to refer to, of which the labels are read with `\xexternaldocument` of `zref-xr` and prefixed with $\langle prefix \rangle$.


```

\newcommand\refdocument[3][ext-]{%
  \newartifact{#2}{footnote=false,#3}%
  \zexternaldocument[#1]{\artifactfullname{#2}}[#2]%
}

```

`\masterdocument` Like `\refdocument`, but with the definitions of the other document loaded as well and with the document itself attached at the first occurrence of a reference or a definition. With `bib2gls`, the hyperlinks of those definitions are aimed at the other document.

```

\newcommand\masterdocument[3][ext-]{%
  \newartifact{#2}{#3}%
  \zexternaldocument[#1]{\artifactfullname{#2}}[#2]%
  \ifthenelse{\equal{\artifactdefs{#2}}{}}{%
    \PackageWarning{regulatory}{No definitions given to the master document. \the\gls commands will therefore
  }{%
    \loadextdefs[#2]{\artifactdefs{#2}}%
    \ifregulatory@bib2gls%
      \glssetcategoryattribute{#2}{targeturl}{\artifactfilename{#2}}%
      \glssetcategoryattribute{#2}{targetname}{\glolinkprefix\glslabel}%
    \fi%
  }%
}

```

14.5 regulatory-md

14.5.1 Vereisten

This module is loaded by `regulatory-struct` as soon as `markdown` is loaded, which is what the `md` option comes down to. `regulatory-defs` comes with it, because a Markdown definition list is aligned on the widest name glossaries-extra knows of. Both prerequisites are therefore normally settled already; they are stated for the document that loads this module on its own.

```

\RequirePackage{regulatory-struct}
\RequirePackage{regulatory-defs}
\RequirePackage{markdown}

```

14.5.2 Conversieopties

The renderers below only see what the reader was told to look for, so the options they depend on are set here rather than left to the document or to the `md` option of `regulatory-struct`. A document that loads `markdown` by itself then gets the same conversion as one that asked for it through the option.

`markdown` forwards its package options to `\markdownSetup` and reads them at conversion time rather than at load time, so setting them here is the same thing as naming them when the package is loaded, only later.

`extension` names the file that adds the standalone identifier of paragraph 11.2 to the grammar of the reader. It is the singular of `extensions`, which appends rather than replaces, so an extension of the document's own is left in place. `markdown` looks the file

up with `kpathsea` and stops with an error when it is not there, which is the one way round this bundle can fail loudly.

`hybrid` is off. It made every backslash in a Markdown source reach $\TeX$, which is how the examples of this bundle used to write a label or a reference; `markdown` soft-deprecated it and names `rawAttribute` among its replacements. What it costs to leave it on is not a warning but a wrong document: with `hybrid` off, raw $\TeX$ is typeset verbatim, so a source that still contains it prints its own control sequences and loses every reference without a single error. The constructs that used to need it each have a Markdown form of their own below.

```
\markdownSetup{
  extension = regulatory-syntax.lua,
  hybrid = false,
  hashEnumerators,
  headerAttributes,
  bracketedSpans,
  relativeReferences,
  definitionLists,
  fencedDivs,
  jekyllData,
  tightLists,
}
```

14.5.3 Structuren

A renderer tells `markdown` how to typeset one construct of a Markdown source. Only the constructs that have a counterpart in this bundle are redirected; everything else keeps the rendering of the `markdown` defaults.

`\markdownRendererHeadingOne` A first level heading is an article. Deeper headings are left alone: a regulatory document numbers its paragraphs as list items, not as headings.

An article is labelled with the attribute syntax of `headerAttributes`, `# Heading {#art:x}`. The `markdown` defaults theme turns such an identifier into a `\label` of its own, after the heading renderer has run, so the label lands on the article and carries its reference properties without this module having to do anything.

```
\markdownSetup{
  renderers = {
    headingOne = {\article{#1}},
  }
}
```

`\markdownRendererOLBegin` An ordered list is a `paras` environment, and its numbering is composed by this bundle rather than taken from the Markdown source. Tight and loose lists are rendered the same way, since the spacing of `paras` is set by `enumitem`.

The item carries no label of its own. It used to be given one made from the number in the source, `m1:⟨number⟩`, but `markdown` hands a renderer nothing except that number: no nesting depth, no list identity. The number restarts in every list, so `m1:1` existed once per list and per article, and a reference to it silently resolved to whichever item was declared last. Worse, such a key is positional: inserting a paragraph moves every reference after

it to another provision without anything failing. An item that is referred to is labelled by hand, with the bracketed span below.

```
olBegin = {\begin{paras}},
olEnd = {\end{paras}},
olBeginTight = {\begin{paras}},
olEndTight = {\end{paras}},
olItemWithNumber = {\item{}},
```

`\markdownRendererLink` Every reference of a Markdown source arrives here: `relativeReferences` lets an autolink hold something that is not a web address, and a link whose target starts with # is a reference to a label of this document or of another one declared with `\refdocument`.

The three shapes a Markdown link can take are enough to reach the whole reference family without any syntax of our own, and each of them says what it means:

`<#label>` an autolink, where the text equals the target, becomes `\Aref:` the reference names the structures it points at, and several labels may be given at once, `<#lid:a,lid:b>`.

`[](#label)` a link with no text becomes `\rref:` the bare number of the structure.

`[text](#label)` a link with text keeps that text and turns it into a hyperlink.

An attribute writes a label and a link reads one, which is the whole of it: `{#lid:a}` on a heading or a span is where a provision is named, and the three shapes above are how it is referred to. A definition is referred to in exactly the same way. Its label is not a label of the document but an entry of the glossary, so the target is looked up there first: a reference to one is a citation, hyperlinked to the definition list, which is what `\gls` and `\glslink` produce.

The target arrives in the third argument, which `markdown` escapes far less than the first two: braces, backslashes and line ends only, so a label comes through exactly as it was written. Anything that is not a reference is handed back to the `markdown` prototype, which is what turns a real web address into a link.

```
link = {\regulatory@md@link{#1}{#2}{#3}{#4}},
```

`\markdownRendererDlItem` A definition list declares definitions. The term carries the label as a bracketed span, `[Term]{#label}`, and the body of the item becomes the description, so that one construct produces an entry that both prints in the list and can be cited with `\gls` elsewhere in the document.

The list itself typesets nothing where it stands. Where the definitions appear is decided by `\printdefs`, or from the source by the division below, which keeps the declaration and the placement apart in the same way the file-based routes of `paragraaf 5` do.

```
dlBegin = {},
dlEnd = {},
dlBeginTight = {},
dlEndTight = {},
dlItem = {\regulatory@md@dlitem{#1}},
dlItemEnd = {\regulatory@md@dlitemend},
},
```

`\markdownRendererTilde` Without hybrid a tilde no longer reaches $\TeX$ as the non-breaking space it is there; the default prototype prints the character itself. A legal text is full of them — “artikel 2” keeps its number on the same line as its noun — so the prototype is set back to what an author typing a tilde means by it.

```

    rendererPrototypes = {
      tilde = {\~},
    },
  }

```

`\markdownRendererRegulatoryLabel` The renderer the syntax extension calls. An identifier that stands on its own is a label, and nothing else: it is written where it stands, so within a `paras` item it names that item. The renderer is declared rather than set through `\markdownSetup`, which only accepts the names `markdown` knows.

```
\newcommand\markdownRendererRegulatoryLabel[1]{\label{#1}}
```

14.5.4 Verwijzingen

`\regulatory@md@link` Decides which of the three shapes above a link has. The test is on the target: a leading `#` makes it a reference, and what remains is the label list, handed to the reference family as it stands. The labels are trimmed, since `<#a, b>` is the natural thing to write and a space would otherwise end the list at the first item.

```

\newcommand\regulatory@md@full[1]{%
  \ifglstryexists{#1}{\gls{#1}}{\Aref{#1}}%
}
\newcommand\regulatory@md@bare[1]{%
  \ifglstryexists{#1}{\gls{#1}}{\rref{#1}}%
}
\newcommand\regulatory@md@named[2]{%
  \ifglstryexists{#1}{\glslink{#1}{#2}}{\hyperref[1]{#2}}%
}

\ExplSyntaxOn
\str_new:N \l__regulatory_md_target_str
\tl_new:N \l__regulatory_md_labels_tl
\seq_new:N \l__regulatory_md_labels_seq

\cs_new_protected:Npn \__regulatory_md_labels:n #1
{
  \seq_set_split:Nnn \l__regulatory_md_labels_seq { , } { #1 }
  \seq_set_map_x:Nnn \l__regulatory_md_labels_seq \l__regulatory_md_labels_seq
    { \tl_trim_spaces:n { ##1 } }
  \tl_set:Nx \l__regulatory_md_labels_tl
    { \seq_use:Nn \l__regulatory_md_labels_seq { , } }
}

\cs_new_protected:Npn \regulatory@md@link #1#2#3#4
{

```

```

\str_set:Nn \__regulatory_md_target_str { #3 }
\str_if_eq:eeTF { \str_range:Nnn \__regulatory_md_target_str { 1 } { 1 } }
{ \c_hash_str }
{
  \__regulatory_md_labels:n
  { \str_range:Nnn \__regulatory_md_target_str { 2 } { -1 } }
  \tl_if_blank:nTF { #1 }
  { \exp_args:NV \regulatory@md@bare \__regulatory_md_labels_tl }
  {
    \str_if_eq:nnTF { #1 } { #2 }
    { \exp_args:NV \regulatory@md@full \__regulatory_md_labels_tl }
    { \exp_args:NV \regulatory@md@named \__regulatory_md_labels_tl { #1 } }
  }
}
{ \markdownRendererLinkPrototype { #1 } { #2 } { #3 } { #4 } }
}
\ExplSyntaxOff

```

14.5.5 Definities

`\regulatory@md@dlitem` Collects one item of a definition list. The term is executed into a box that is thrown away, with the attribute renderers replaced, so that the identifier of its bracketed span is captured instead of becoming a label of its own and the name is captured instead of being typeset. Executing it is what makes the capture work: the renderers of markdown are not expandable, so reading the term as text would store the calls themselves rather than what they carry. A term that turns out to have no span is read as text after all, since then it holds nothing but the name.

A term without an identifier still becomes a definition, but one that cannot be cited by name, so it is reported.

```

\ExplSyntaxOn
\cs_new_protected:Npn \regulatory@md@declare #1#2#3
{
  \use:x
  {
    \exp_not:N \newdefinition { #1 }
    { \exp_not:N #2 } { \exp_not:N #3 }
  }
}
\ExplSyntaxOff

\newcommand\regulatory@md@deflabel{}
\newcommand\regulatory@md@defname{}
\newcommand\regulatory@md@defbody{}
\newcounter{regulatory@md@defs}
\newcounter{regulatory@md@declared}

\newcommand\regulatory@md@dlitem[1]{%
  \gdef\regulatory@md@deflabel{}%

```

```

\gdef\regulatory@md@defname{}}%
\gdef\regulatory@md@defbody{}}%
\begingroup%
  \def\markdownRendererBracketedSpanAttributeContextBegin{}}%
  \def\markdownRendererBracketedSpanAttributeContextEnd{}}%
  \def\markdownRendererAttributeIdentifier##1{\gdef\regulatory@md@deflabel{##1}}%
  \def\markdownRendererAttributeClassName##1{}}%
  \def\markdownRendererAttributeKeyValue##1##2{}}%
  \def\markdownRendererBracketedSpan##1{\gdef\regulatory@md@defname{##1}}%
  \setbox\z@\hbox{##1}%
\endgroup%
\ifx\regulatory@md@defname\@empty%
  \protected@xdef\regulatory@md@defname{##1}%
\fi%
\ifx\regulatory@md@deflabel\@empty%
  \stepcounter{regulatory@md@defs}%
  \xdef\regulatory@md@deflabel{regulatory@md@arabic{regulatory@md@defs}}%
  \PackageWarning{regulatory-md}{%
    Definition without an identifier; give the term a bracketed\MessageBreak
    span with one, so that it can be cited with \string\gls. Reported}%
\fi%
}

```

`\markdownRendererDlDefinitionBegin` The description of a definition is not handed to a renderer as an argument: it is streamed between `\markdownRendererDlDefinitionBegin` and `\markdownRendererDlDefinitionEnd`, which are both ordinary tokens in the converted file. Making the first of the two a delimited macro therefore hands the whole body over as an argument. A term may carry more than one description, which are collected in the order they are written.

```

\long\def\markdownRendererDlDefinitionBegin#1\markdownRendererDlDefinitionEnd{%
  \ifx\regulatory@md@defbody\@empty%
    \protected@xdef\regulatory@md@defbody{##1}%
  \else%
    \protected@xdef\regulatory@md@defbody{\regulatory@md@defbody\space##1}%
  \fi%
}

```

`\regulatory@md@dlitemend` Declares what was collected. `\newdefinition` is the same route a document takes when it declares a definition by hand, so a Markdown definition list and a `\newdefinition` in the preamble produce entries that are indistinguishable afterwards.

```

\newcommand\regulatory@md@dlitemend{%
  \stepcounter{regulatory@md@declared}%
  \regulatory@md@declare\regulatory@md@deflabel\regulatory@md@defname\regulatory@md@defbody%
}

```

`endererFencedDivAttributeContextBegin` A division marks where the definitions are printed, `::: {.definitions}`. The class is what selects it; a style and a widest attribute may be given along with it, which are the two arguments of `\printdefs`. The list is printed after the content of the division, so that a sentence introducing it can be written inside.

markdown does not recognise an empty division, so a division that is to print nothing but the list still needs a line of text in it.

```

\newcommand\regulatory@md@divdefs{false}
\newcommand\regulatory@md@divstyle{\regulatory@defstyle}
\newcommand\regulatory@md@divwidest{}

\renewcommand\markdownRendererFencedDivAttributeContextBegin{%
  \def\regulatory@md@divdefs{false}%
  \def\regulatory@md@divstyle{\regulatory@defstyle}%
  \def\regulatory@md@divwidest{}%
  \def\markdownRendererAttributeClassName##1{%
    \def\regulatory@md@class{##1}%
    \def\regulatory@md@definitionsclass{definitions}%
    \ifx\regulatory@md@class\regulatory@md@definitionsclass%
      \def\regulatory@md@divdefs{true}%
    \fi%
  }%
  \def\markdownRendererAttributeKeyValue##1##2{%
    \def\regulatory@md@key{##1}%
    \def\regulatory@md@stylekey{style}%
    \def\regulatory@md@widestkey{widest}%
    \ifx\regulatory@md@key\regulatory@md@stylekey\def\regulatory@md@divstyle{##2}\fi%
    \ifx\regulatory@md@key\regulatory@md@widestkey\def\regulatory@md@divwidest{##2}\fi%
  }%
}

\renewcommand\markdownRendererFencedDivAttributeContextEnd{%
  \def\regulatory@md@true{true}%
  \ifx\regulatory@md@divdefs\regulatory@md@true%
    \par%
    \printdefs[\regulatory@md@divstyle]{\regulatory@md@divwidest}%
  \fi%
}

```

14.5.6 Definities in de metadata

\regulatory@md@yaml The other way to declare a definition is the `YAML` block a Markdown source may open with, where a definition is a record with named fields rather than a construct of the text. It is the sturdier of the two: nothing in the prose can perturb it, and any field glossaries knows could be added to it. What it costs is that the definitions are no longer where the reader of the source expects them.

The fields of one record do not arrive in the order they are written but alphabetically, so they are collected and the entry is declared when the record ends. That is a prototype rather than a renderer: the renderer of a mapping end is what pops the address stack of markdown itself, so it is appended to instead of replaced.

```

\newcommand\regulatory@md@yamllabel{}
\newcommand\regulatory@md@yamlname{}

```

```

\newcommand\regulatory@md@yamldesc{}

\markdownSetup{
  jekyllDataRenderers = {
    /definitions/*/label = {\gdef\regulatory@md@yamllabel{#1}},
    /definitions/*/name = {\gdef\regulatory@md@yamlname{#1}},
    /definitions/*/description = {\gdef\regulatory@md@yamldesc{#1}},
  },
  rendererPrototypes = {
    jekyllDataMappingEnd + = {\regulatory@md@yamlflush},
  },
}

\newcommand\regulatory@md@yamlflush{%
  \ifx\regulatory@md@yamllabel\empty\else%
    \stepcounter{regulatory@md@declared}%
    \regulatory@md@declare\regulatory@md@yamllabel\regulatory@md@yamlname\regulatory@md@yamldesc%
    \gdef\regulatory@md@yamllabel{}%
    \gdef\regulatory@md@yamlname{}%
    \gdef\regulatory@md@yamldesc{}%
  \fi%
}

```

14.5.7 Een definitie aanhalen

`\autocite` A definition is cited the same way anything else is referred to, with a bracketed span: `[verwerking]{#vw}` prints that text and links it to the entry. That is explicit, and it is the documented way.

`\autocite` is the other one, for a document that would rather keep its prose free of markup: it hands the names of the definitions declared so far to the acronyms machinery of `markdown`, which then marks up every occurrence of such a name in the text that follows and turns it into a `\gls`. It is opt-in for good reason. The matching is literal: case sensitive, blind to inflection — “persoonsgegevens” is not “persoonsgegeven” — and it does not see a term that a line break has split. It also only affects text converted after the call, so the definitions have to be declared in a file of their own, converted first.

```

\ExplSyntaxOn
\prop_new:N \g__regulatory_md_names_prop
\tl_new:N \l__regulatory_md_name_tl

\cs_new_protected:Npn \regulatory@md@register #1
{
  \tl_set:Nx \l__regulatory_md_name_tl { \glsentryname { #1 } }
  \prop_gput:Nvn \g__regulatory_md_names_prop \l__regulatory_md_name_tl { #1 }
  \tl_gput_right:Nx \markdownOptionAcronyms
    { , { \l__regulatory_md_name_tl } }
}

\cs_new_protected:Npn \regulatory@md@cite #1

```



```

{
  \prop_get:NnNTF \g__regulatory_md_names_prop { #1 } \l__regulatory_md_name_tl
  { \exp_args:NV \gls \l__regulatory_md_name_tl }
  { #1 }
}
\ExplSyntaxOff

\newcommand\autocitedefs{%
  \forglsentries[definitions]\regulatory@md@entry{%
    \expandafter\regulatory@md@register\expandafter{\regulatory@md@entry}%
  }%
  \renewcommand\markdownRendererAcronymPrototype[1]{\regulatory@md@cite{##1}}%
}

```

\regulatory@md@nodefs A definition list that is never printed is the one mistake this module cannot catch while it happens: the entries are declared, the conversion succeeds, and the definitions article of the document is simply empty. The end of the document is the first moment at which that is knowable, so it is said there.

```

\AtEndDocument{%
  \ifnum\value{regulatory@md@declared}>0%
    \ifnum\value{regulatory@defspainted}=0%
      \PackageWarning{regulatory-md}{%
        Definitions were declared but never printed;\MessageBreak
        call \string\printdefs\space or mark the place with a\MessageBreak
        definitions division. Reported}%
      \fi%
    \fi%
  }

```

14.6 regulatory-sign

This module comes from xdp-sign of Xerdi's Documentation Project, where it was written for the agreement class of that bundle. A signature belongs to the same kind of document as the rest of this package, and a document that needs one should not have to reach for a package that is in no distribution, so it is maintained here.

14.6.1 Vereisten

A signature field is a widget annotation, which needs hyperref for the form machinery it belongs to. It is loaded by regulatory-struct, which is stated here for the document that loads this module on its own. Options for hyperref have to be passed before either is loaded, for instance with \PassOptionsToPackage{<hidelinks>}{<hyperref>}.

```

\RequirePackage{regulatory-struct}
\RequirePackage{hyperref}

```

14.6.2 De annotatie

\regulatory@sign@annot Draws a box of <width> by <height> with a rule at the top and at the bottom, and puts a

PDF annotation of exactly that rectangle over it, so that a viewer offers the field where the box is.

There are two ways to place one, and which is available depends on the document rather than on the engine. With PDF management active — which is what `\DocumentMetadata` turns on, and what `regulatory-attachments` needs anyway — the annotation goes through `\pdfannot_box:nnnn` of the kernel, which knows about the page it lands on and about the standards the document declares. Without it, the engine primitive is the only route: `\pdfextension annot` under `LuaTeX` and current `pdfTeX`, `\pdfannot` under an older one. Measured: the kernel command exists only under `\DocumentMetadata`, so the test is for the command and not for the engine.

A document that has neither gets a warning and a drawn box without a field, which is a signature line to write on but not one to sign in a viewer. That is worth saying out loud: a missing annotation is invisible on paper and the difference only shows when someone tries to sign.

```
\newcommand*\regulatory@sign@annot[3]{%
  \begingroup
    \setlength{\dimen0}{#1}%
    \setlength{\dimen1}{#2}%
    \leavevmode
    \hbox to \dimen0{%
      \vbox to \dimen1{%
        \hrule height0.4pt\relax
        \vfill
        \hrule height0.4pt\relax
      }\hss
    }%
    \kern-\dimen0
    \raisebox{0pt}[0pt][0pt]{%
      \hbox to 0pt{%
        \hskip0pt
        \regulatory@sign@place{#3}%
        \hss
      }%
    }%
    \kern\dimen0
  \endgroup
}
```

`\regulatory@sign@place` Emits the annotation itself, by whichever of the three routes this document has.

```
\ExplSyntaxOn
\cs_new_protected:Npn \regulatory@sign@place #1
{
  \cs_if_exist:NTF \pdfannot_box:nnnn
  { \pdfannot_box:nnnn { \dimen0 } { \dimen1 } { 0pt } { #1 } }
  {
    \cs_if_exist:NTF \pdfextension
    { \tex_pdfextension:D annot ~ width ~ \dimen0 ~
```

```

        height ~ \dimen1 ~ depth ~ 0pt ~ { #1 } }
    {
        \cs_if_exist:NTF \pdfannot
        { \pdfannot ~ width ~ \dimen0 ~
          height ~ \dimen1 ~ depth ~ 0pt ~ { #1 } }
        {
            \msg_warning:nn { regulatory-sign } { no-annotation }
        }
    }
}

}

\msg_new:nnn { regulatory-sign } { no-annotation }
{
    This-engine-offers-no-way-to-place-an-annotation,~so-the-signature~\
    field-is-a-drawn-box-and-nothing-more.~Declare~\iow_char:N\\DocumentMetadata-
    to-let-the~kernel-place-it.
}
\ExplSyntaxOff

```

14.6.3 Handtekeningvelden

`\regulatory@sign@appearance` An annotation has to say what it looks like. ISO 19005 puts it as a requirement rather than a nicety: rule 6.3.3-1 of both PDF/A-2 and PDF/A-3 asks every annotation that is not a Popup, a Link or of zero size to carry an appearance dictionary, and veraPDF says of a field without one “An annotation does not contain an appearance dictionary”. A signature field is a widget, so it is asked as well, and a document with one used to lose its conformance over it.

What it points at is deliberately empty. The line to sign on is drawn on the page by `\regulatory@sign@annot`, not by the annotation, so an appearance that drew anything would draw it twice; and a signing tool replaces the normal appearance of the field with its own the moment the field is signed. One object serves every field: the bounding box of an appearance is mapped onto the rectangle of the annotation, so a stream that draws nothing needs no size of its own.

Only where the PDF management of $\text{\LaTeX}$ is active, which a document that cares about PDF/A has declared anyway, since the standards need it for much more than this. The test is for the management and not for the command: under `tex4ht` the object commands exist and their backend does not, so asking whether they are defined answers yes and then fails on the first call.

```

\ExplSyntaxOn
\cs_new:Npn \regulatory@sign@appearance { }
\bool_lazy_and:nnT
{ \cs_if_exist_p:N \pdfmanagement_if_active_p: }
{ \cs_if_exist_p:N \pdf_object_new:n }
{
    \pdfmanagement_if_active:T
    {

```

```

\pdf_object_new:n { regulatory/sign/blank }
\AddToHook { shipout/firstpage }
{
  \pdf_object_write:nnn { regulatory/sign/blank } { stream }
  { { /Type /XObject /Subtype /Form /BBox [0~0~1~1] } { } }
}
\cs_gset:Npn \regulatory@sign@appearance
{ /AP~<<~/N~\pdf_object_ref:n { regulatory/sign/blank }~>> }
}
}
\ExplSyntaxOff

```

`\signaturefield` Draws a signature field of $\langle width \rangle$ by $\langle height \rangle$ under the name $\langle name \rangle$, which is the name a viewer stores the signature under and the one a signing tool asks for. The optional $\langle tooltip \rangle$ is what a reader is told the field is for, which is also what a screen reader announces, so it is worth giving.

```

\newcommand\signaturefield[4][]{%
  \def\regulatory@sign@tooltip{#1}%
  \regulatory@sign@annot{#3}{#4}{%
    /Subtype /Widget
    /FT /Sig
    /T (#2)
    \ifx\regulatory@sign@tooltip\empty\else /TU (#1)\fi
    /BS << /W 0 >>
    /F 4
    /DA (/Helv 0 Tf 0 g)
    \regulatory@sign@appearance
  }%
}

```

`\SignatureField` The name this command had in `xdp-sign`, kept so that a document written for that package keeps working when it is moved over. New documents use the lowercase name, which is the one the rest of this bundle is written in.

```

\let\SignatureField\signaturefield
\let\SignatureAnnotation\regulatory@sign@annot

```

14.7 regulatory-sources

A regulatory document cites instruments that are not this document: a statute, a regulation of the Union. This module holds the ones a document cites, so that the wording of a citation is a property of the document rather than of the sentence it happens to stand in. What it does *not* do is decide the wording; that is the next module. This one reads and holds.

14.7.1 Vereisten

```

\RequirePackage{regulatory-struct}

```

14.7.2 Bronnen van een bron

Sources are declared in a bib file, which is the format such records are already kept in, and read by this module rather than by biblatex or bib2gls. Neither would be wrong; both would put an external program between a document and a handful of records. The test suite of this bundle promises a T_EX installation and nothing else, the conversion to HTML runs the same documents, and a document rarely cites more than a few instruments. So the reading is done here, and the price of that is that this module owns a parser.

A parser that is owned has to say what it accepts. This one accepts the shape a bib file is written in when a person writes it, and nothing further:

an entry opens with `@⟨type⟩{⟨key⟩}`, on a line of its own;

a field is `⟨name⟩ = {⟨value⟩}`, on a line of its own;

the entry closes with `}` on a line of its own;

a line that is empty, or that starts with `%`, is a comment.

A line that is none of these is an error naming the file and the line, rather than a line that is skipped. That is the whole reason a parser may be owned at all: the failure of a reader has to be loud, since a record that was not read is a citation that comes out empty and a document that looks finished.

```
\ExplSyntaxOn
\prop_new:N \g__regulatory_src_required_prop
\prop_new:N \g__regulatory_src_type_prop
\prop_new:N \g__regulatory_src_register_prop
\prop_new:N \g__regulatory_src_typealias_prop
\prop_new:N \g__regulatory_src_fieldalias_prop
\seq_new:N \g__regulatory_src_keys_seq

\ior_new:N \g__regulatory_src_ior
\str_new:N \l__regulatory_src_key_str
\str_new:N \l__regulatory_src_type_str
\str_new:N \l__regulatory_src_file_str
\str_new:N \l__regulatory_src_query_str
\int_new:N \l__regulatory_src_line_int
\bool_new:N \l__regulatory_src_open_bool
\seq_new:N \l__regulatory_src_parts_seq
\tl_new:N \l__regulatory_src_required_tl
\clist_new:N \l__regulatory_src_required_clist
\tl_new:N \l__regulatory_src_one_tl
\tl_new:N \l__regulatory_src_two_tl
```

`\newsourcetype` Declares an entry type, the fields an entry of it cannot do without, and the register its citations are worded in. The required fields are what the wording needs in every register: a Dutch regulation is named by its citeertitel, an act of the Union by what kind of act it is and by its number.

The register belongs to the type and not to the document, because it is the legal order of the instrument that decides how a citation of it reads. A Dutch contract that cites the GDPR words that citation the way the regulation itself does. An entry may override it with a register field.

```

\NewDocumentCommand \newsourcetype { m m m }
{
  \prop_gput:Nnn \g__regulatory_src_required_prop { #1 } { #2 }
  \prop_gput:Nnn \g__regulatory_src_register_prop { #1 } { #3 }
}

\NewDocumentCommand \newsourcetypealias { m m }
{ \prop_gput:Nnn \g__regulatory_src_typealias_prop { #1 } { #2 } }

\NewDocumentCommand \newsourcetypealias { m m }
{ \prop_gput:Nnn \g__regulatory_src_fieldalias_prop { #1 } { #2 } }

\cs_new:Npn \__regulatory_src_alias:Nn #1#2
{
  \tl_if_blank:eTF { \prop_item:Nn #1 { #2 } }
  { #2 } { \prop_item:Nn #1 { #2 } }
}
\cs_generate_variant:Nn \__regulatory_src_alias:Nn { Ne }

\cs_new:Npn \regulatory@src@type@of #1
{ \__regulatory_src_alias:Ne \g__regulatory_src_typealias_prop { #1 } }
\cs_new:Npn \regulatory@src@field@of #1
{ \__regulatory_src_alias:Ne \g__regulatory_src_fieldalias_prop { #1 } }
\ExplSyntaxOff

\newsourcetype{regulation}{citetitle}{dutch}
\newsourcetype{euact}{kind,number}{dutch-eu}

```

The names above are English, as the rest of this bundle is, and every one of them answers to a Dutch name as well. A bib file is written by the person who keeps the records, and that person writes `citetitle` because that is what the field is called in the law that defines it. Neither name is a translation of the other in the reader: the Dutch one resolves to the English one before anything is stored, so a file may use either and a document may read back either.

```

\newsourcetypealias{regeling}{regulation}
\newsourcetypealias{euhandeling}{euact}

\newsourcetypealias{citeertitel}{citetitle}
\newsourcetypealias{afkorting}{abbreviation}
\newsourcetypealias{boek}{book}
\newsourcetypealias{geldig}{valid}
\newsourcetypealias{soort}{kind}
\newsourcetypealias{nummer}{number}
\newsourcetypealias{roepnaam}{shortname}

```

```

\newsourcedalias{titel}{title}
\newsourcedalias{pb}{oj}
\newsourcedalias{lidwoord}{determiner}

```

`\newsourced` Declares one source in the document itself, which is the route that needs no file at all. It ends in the same store as a line read from a bib file, so nothing downstream can tell the two apart.

```

\ExplSyntaxOn
\NewDocumentCommand \newsourced { m m m }
{
  \__regulatory_src_begin:nn { #2 } { #1 }
  \keyval_parse:nnn
    { \__regulatory_src_novaluen { \__regulatory_src_field:nn } { #3 }
    \__regulatory_src_end:
  }

  \cs_new_protected:Npn \__regulatory_src_novaluen #1
  {
    \msg_error:nnn { regulatory-sources } { no-value } { #1 }
  }
}

```

Opening an entry

`\__regulatory_src_begin:nn` Opens an entry of *<type>* under *<key>*. An undeclared type and a key that is already taken are both errors: the first is a record nothing can word, the second is a record that would quietly replace one the document already relies on.

```

\cs_new_protected:Npn \__regulatory_src_begin:nn #1#2
{
  \str_set:Nx \__regulatory_src_type_str { \regulatory@src@type@of { #1 } }
  \str_set:Nn \__regulatory_src_key_str { #2 }
  \prop_if_in:NVF \g__regulatory_src_required_prop \__regulatory_src_type_str
  {
    \msg_error:nnxx { regulatory-sources } { unknown-type }
    { \__regulatory_src_type_str } { \__regulatory_src_key_str }
  }
  \seq_if_in:NVT \g__regulatory_src_keys_seq \__regulatory_src_key_str
  {
    \msg_error:nnx { regulatory-sources } { duplicate-key }
    { \__regulatory_src_key_str }
  }
  \prop_new:c { g__regulatory_src_e_ \__regulatory_src_key_str _prop }
  \bool_set_true:N \__regulatory_src_open_bool
}

```

Storing a field

`\__regulatory_src_field:nn` Stores one field of the entry that is open. The name of the field is lowered and expanded into the key it is stored under, while the value is not

expanded at all: a value is data, and a document that writes a macro in a bib file gets that macro back rather than what it produced at reading time.

```
\cs_new_protected:Npn \__regulatory_src_field:nn #1#2
{
  \bool_if:NTF \l__regulatory_src_open_bool
  {
    \use:x
    {
      \prop_gput:cnn
      { g__regulatory_src_e_ \l__regulatory_src_key_str _prop }
      { \regulatory@src@field@of { \str_lowercase:n { #1 } } }
      { \exp_not:n { #2 } }
    }
  }
  { \msg_error:nnn { regulatory-sources } { stray-field } { #1 } }
}
```

Closing an entry

`\__regulatory_src_end:` Closes the entry and holds it against the fields its type requires. This is the moment the guarantee is made: from here on a source may be read without asking whether it is complete, which is what lets the commands that word a citation be expandable and silent.

```
\cs_new_protected:Npn \__regulatory_src_end:
{
  \prop_get:NVN \g__regulatory_src_required_prop \l__regulatory_src_type_str
  \l__regulatory_src_required_tl
  \quark_if_no_value:NF \l__regulatory_src_required_tl
  {
    \clist_set:NV \l__regulatory_src_required_clist
    \l__regulatory_src_required_tl
    \clist_map_inline:Nn \l__regulatory_src_required_clist
    {
      \prop_if_in:cnF
      { g__regulatory_src_e_ \l__regulatory_src_key_str _prop } { ##1 }
      {
        \msg_error:nnxxx { regulatory-sources } { missing-field }
        { ##1 } { \l__regulatory_src_key_str }
        { \l__regulatory_src_type_str }
      }
    }
  }
  \__regulatory_src_checkvalid:
  \prop_gput:NVV \g__regulatory_src_type_prop
  \l__regulatory_src_key_str \l__regulatory_src_type_str
  \seq_gput_right:NV \g__regulatory_src_keys_seq \l__regulatory_src_key_str
  \bool_set_false:N \l__regulatory_src_open_bool
}
```


__regulatory_src_checkvalid: Every entry says when it was last held against the text it describes. The date it gives is what \sourcedate is compared with, so an entry without one is never reported stale: “not looked at yet” and “looked at and current” would come out as the same silence. A guard whose off switch is an absent field is switched off by accident rather than by decision, so the absence is said out loud and the decision is given a word of its own — valid = unverified, which is silent, because the author has then said it. Anything else that is not a date is said as well, since a date in another shape compares wrongly rather than not at all.

Asked for by the sibling session that keeps the sources this reader is written for, on the argument this whole bundle is built on.

There is deliberately no fourth state for “checked and known to have changed”. Such an entry already warns: it keeps its last good date, the document speaks later, and the source is named. A word in place of that date would fire the same warning while throwing the date away, and the date is the one thing that says how far behind the entry is and whether the citation in front of the reader predates the change. A note that an entry is known to be wrong is for a person reading the file, not a state this guard reads.

```
\cs_new_protected:Npn \__regulatory_src_checkvalid:
{
  \prop_get:cnTF { g__regulatory_src_e_ \l__regulatory_src_key_str _prop }
  { valid } \l__regulatory_src_valid_tl
  {
    \str_if_eq:VnF \l__regulatory_src_valid_tl { unverified }
    {
      \regex_match:NVF \c__regulatory_src_date_regex \l__regulatory_src_valid_tl
      {
        \msg_warning:nnxx { regulatory-sources } { valid-not-iso }
        { \l__regulatory_src_key_str } { \l__regulatory_src_valid_tl }
      }
    }
  }
}

\cs_generate_variant:Nn \regex_match:NnF { NVF }

\msg_new:nnn { regulatory-sources } { no-valid }
{
  The-source~`#1'~does-not-say-when-it-was-last-held-against-the-text-it~
  describes,~\
  so-it-can-never-be-reported-out-of-date.~Give-it-valid~~YYYY-MM-DD,~or~
  valid~~unverified-to-say-so-on-purpose.~Reported
}

\msg_new:nnn { regulatory-sources } { valid-not-iso }
{
  The-source~`#1'~gives-valid~~`#2',~which-is-neither-YYYY-MM-DD-nor~
```

```

unverified.~\\
Dates-are-compared-as-they-are-written,~so-this-one-compares-wrongly-rather-
than-not-at-all.~Reported
}

```

`\loadsources` Reads $\langle file \rangle$.bib. The file is looked up the way $\TeX$ looks up a file it is given, so it is found next to the document or anywhere on the input path; a file that is not there is an error and not an empty list of sources.

```

\NewDocumentCommand \loadsources { m }
{
  \file_get_full_name:nNTF { #1 .bib } \__regulatory_src_file_str
  { \__regulatory_src_read:V \__regulatory_src_file_str }
  { \msg_error:nnn { regulatory-sources } { file-not-found } { #1 .bib } }
}

\cs_new_protected:Npn \__regulatory_src_read:n #1
{
  \int_zero:N \__regulatory_src_line_int
  \bool_set_false:N \__regulatory_src_open_bool
  \ior_open:Nn \g__regulatory_src_ior { #1 }
  \ior_str_map_inline:Nn \g__regulatory_src_ior
  { \__regulatory_src_line:nn { #1 } { ##1 } }
  \ior_close:N \g__regulatory_src_ior
  \bool_if:NT \__regulatory_src_open_bool
  {
    \msg_error:nxxx { regulatory-sources } { unclosed }
    { \__regulatory_src_key_str } { #1 }
  }
}

\cs_generate_variant:Nn \__regulatory_src_read:n { V }

```

Reading one line

`\__regulatory_src_line:nn` One line of the file, tried against the four shapes the subset knows. The order is the order they occur in: a comment or a blank line first, since a file holds more of those than anything else.

```

\regex_const:Nn \c__regulatory_src_blank_regex { \A\s* (\%.*)? \Z }
\regex_const:Nn \c__regulatory_src_open_regex
{ \A\s* \@ (\w+) \s* \{ \s* ([^\s\{ \}]+) \s* ,? \s* \Z }
\regex_const:Nn \c__regulatory_src_field_regex
{ \A\s* (\w+) \s* = \s* \{ (.*?) \} \s* ,? \s* \Z }
\regex_const:Nn \c__regulatory_src_close_regex { \A\s* \} \s* ,? \s* \Z }

\cs_new_protected:Npn \__regulatory_src_pair:N #1
{
  \tl_set:Nx \l__regulatory_src_one_tl
  { \seq_item:Nn \l__regulatory_src_parts_seq { 2 } }
  \tl_set:Nx \l__regulatory_src_two_tl
  { \seq_item:Nn \l__regulatory_src_parts_seq { 3 } }
}

```

```

\exp_args:NVV #1 \l__regulatory_src_one_tl \l__regulatory_src_two_tl
}

\cs_new_protected:Npn \__regulatory_src_line:nn #1#2
{
  \int_incr:N \l__regulatory_src_line_int
  \regex_match:NnTF \c__regulatory_src_blank_regex { #2 }
  { }
  {
    \regex_extract_once:NnNTF \c__regulatory_src_open_regex { #2 }
    \l__regulatory_src_parts_seq
    { \__regulatory_src_pair:N \__regulatory_src_begin:nn }
    {
      \regex_extract_once:NnNTF \c__regulatory_src_field_regex { #2 }
      \l__regulatory_src_parts_seq
      { \__regulatory_src_pair:N \__regulatory_src_field:nn }
      {
        \regex_match:NnTF \c__regulatory_src_close_regex { #2 }
        { \__regulatory_src_end: }
        {
          \msg_error:nnxxx { regulatory-sources } { unreadable }
          { \int_use:N \l__regulatory_src_line_int } { #1 } { #2 }
        }
      }
    }
  }
}

```

`\srcfield` Reads one field of a source. Both are expandable and both give nothing for a field that is not there, which is safe precisely because a source cannot be declared without the fields its type requires: what is absent here is optional by construction.

```

\cs_new:Npn \__regulatory_src_read:nn #1#2
{ \prop_item:cn { g__regulatory_src_e_ #1 _prop } { #2 } }
\cs_generate_variant:Nn \__regulatory_src_read:nn { ee }

\cs_new:Npn \srcfield #1#2
{ \__regulatory_src_read:ee { #1 } { \regulatory@src@field@of { #2 } } }
\cs_new:Npn \srctype #1
{
  \prop_item:Nn \g__regulatory_src_type_prop { #1 }
}

```

`\srcregister` The register a source is worded in: the one its type declares, unless the entry says otherwise.

```

\cs_new:Npn \srcregister #1
{
  \tl_if_blank:eTF { \srcfield { #1 } { register } }
  {
    \prop_item:Ne \g__regulatory_src_register_prop

```

```

        { \prop_item:Nn \g__regulatory_src_type_prop { #1 } }
    }
    { \srcfield { #1 } { register } }
}

```

`\ifsourceexists` Whether $\langle key \rangle$ was declared at all. The key is made into a string before it is looked for: the keys of a file were read as text, and a sequence compares what it holds character by character, category code and all — so a key typed in the document would never equal the same key read from a file.

```

\NewDocumentCommand \ifsourceexists { m m m }
{
  \str_set:Nn \l__regulatory_src_query_str { #1 }
  \seq_if_in:NVTF \g__regulatory_src_keys_seq \l__regulatory_src_query_str
    { #2 } { #3 }
}

```

14.7.3 Registers

A citation is worded in the register of the instrument it points at, and every register has two systems: one written out in full and one shortened. Which of the two applies is a property of the place the citation stands in — the Leidraad has it in full in the running text and shortened in a footnote, and shortened as well when it stands between brackets in the running text — so the call decides and the document sets the default.

Seven keys are enough to word a citation. Five of them are macros, and each of them is handed the macro to put its result in rather than expanding to it: writing an ordinal in words is work that `fmtcount` does with a command that cannot be expanded, and a citation is assembled before it is typeset so that it can be read back.

`article` the word before the number of an article.

`number` the number itself, which is where the colon notation of the civil code lives.

`paragraph` the paragraph, which Dutch calls a lid.

`point` the lettered point, which Dutch calls an *onderdeel*.

`subpoint` the level below that, which the standards call a *subonderdeel*.

`separator` what stands between the parts: a comma in full, a space when shortened.

`determiner` the article a name takes when neither the entry nor the kind of the instrument says, which is a genitive in the registers of the Union: “der Verordnung”, “du règlement”.

`connective` what stands between the last part and the instrument: “connective” when written out in full, nothing when shortened.

`assemble` how the parts are arranged: from the outside in, which is what a register that leaves this out gets, or the other way round, which is what English does.

attachnum the shape the number of a paragraph takes when it hangs on its article rather than standing on its own. Brackets in English, and around each of them where there is more than one: “Article 6(1), (2) and (3)”, measured, against “paragraphs 1 and 2” where the same paragraphs stand by themselves.

attach how that number is bound to the article, which in English is simply against it. Only a register arranged the other way round needs either of these two.

source the instrument itself, named from the fields of its entry.

```
\ExplSyntaxOff
\newcommand\regulatory@src@ordinal[1]{\ordinalstringnum{#1}}

\ExplSyntaxOn
\prop_new:N \g__regulatory_src_register_store_prop
\cs_generate_variant:Nn \prop_item:Nn { Ne }
```

Which registers were declared, in the order they were, so that the set of them can be read back. The store above is keyed by register, system and key at once, so it answers what a register says and not which registers there are, and the manual names them one by one. This is what lets the suite hold that list against the code rather than against the last time somebody looked.

```
\seq_new:N \g__regulatory_src_registers_seq

\cs_new_protected:Npn \__regulatory_src_register_put:nnnn #1#2#3#4
{ \prop_gput:Nnn \g__regulatory_src_register_store_prop { #1 / #2 / #3 } { #4 } }

\cs_new_protected:Npn \__regulatory_src_register_bare:nnn #1#2#3
{ \msg_error:nnnn { regulatory-sources } { register-key-without-value } { #1 } { #2 } { #3 } }

\msg_new:nnn { regulatory-sources } { register-key-without-value }
{
  The-key-`#3'-of-register-`#1'-(#2)-was-given-no-value.-\\
  Every-key-of-a-register-carries-one:-a-word,-a-number-format-or-an-arrangement.
}

\NewDocumentCommand \newsourceregister { m m m }
{
  \seq_if_in:NnF \g__regulatory_src_registers_seq { #1 }
  { \seq_gput_right:Nn \g__regulatory_src_registers_seq { #1 } }
  \keyval_parse:nnn
  { \__regulatory_src_register_bare:nnn { #1 } { full } }
  { \__regulatory_src_register_put:nnnn { #1 } { full } }
  { #2 }
  \keyval_parse:nnn
  { \__regulatory_src_register_bare:nnn { #1 } { short } }
  { \__regulatory_src_register_put:nnnn { #1 } { short } }
  { #3 }
}
```

```

\cs_new:Npn \regsrc@get #1#2#3
{ \prop_item:Ne \g__regulatory_src_register_store_prop { #1 / #2 / #3 } }

\cs_new_protected:Npn \regsrc@set #1#2#3#4
{
  \prop_get:NnNF \g__regulatory_src_register_store_prop { #1 / #2 / #3 } #4
  { \cs_set_eq:NN #4 \relax }
}
\cs_generate_variant:Nn \prop_get:NnNF { NeNF }
\ExplSyntaxOff

```

The register of the Dutch legislator, which is also the one a Dutch author writes in. In full it is the form the Aanwijzingen voor de regelgeving prescribe for regulations themselves — a model rather than a rule for a document that is not a regulation, since those instructions bind ministries and not contracts. Shortened it is the form of the Leidraad: no commas, the designation before the number, and the colon notation for the civil code.

```

% Where the designation stands: before its numbers in nearly every register, after
% them where a paragraph is written as an ordinal.
\newcommand\regulatory@src@before[3]{\def#1{#3-#2}}
\newcommand\regulatory@src@after[3]{\def#1{#2\space#3}}

% One number, in the shape its level takes.
\newcommand\regulatory@src@num@plain[2]{\def#1{#2}}
\newcommand\regulatory@src@num@paren[2]{\def#1{#2}}
\newcommand\regulatory@src@num@degree[2]{\def#1{#2\textdegree}}
\newcommand\regulatory@src@num@parens[2]{\def#1{(#2)}}

% A paragraph written into the article it belongs to, which is the English form.
\newcommand\regulatory@src@attach@plain[3]{\def#1{#2#3}}
\newcommand\regulatory@src@num@ordinal[2]{%
  \storeordinalstringnum{regsrc}{#2}%
  \expandafter\let\expandafter\regulatory@src@ord\csname @fcs@regsrc\endcsname
  \let#1\regulatory@src@ord
}
\newcommand\regulatory@src@num@unmeasured[2]{%
  \PackageWarning{regulatory-sources}{%
    This register has no measured form for the level below a point,\MessageBreak
    so the number is written on its own. Reported}%
  \def#1{#2}%
}

\newcommand\regulatory@src@paragraph@absatz[2]{\def#1{Absatz-#2}}
\newcommand\regulatory@src@paragraph@paragraphe[2]{\def#1{paragraphe-#2}}
\newcommand\regulatory@src@point@buchstabe[2]{\def#1{Buchstabe-#2}}
\newcommand\regulatory@src@point@point[2]{\def#1{point-#2}}

```

Which article a name takes is not a property of the register but of the instrument, and the entry already says which instrument it is. Counted in the same regulation: “de la directive” sixteen times against “du règlement” seven, so a single default for French is wrong more often than it is right, and “du directive” is not a matter of style but of grammar. German is the same story one step further: “der” carries Verordnung and Richtlinie but not Beschluss or Vertrag, which take “des”.

So the determiner is looked up by the kind of the instrument, and the determiner field of an entry overrides that for the case no table can know.

The pieces the registers above are built from. A number carries the book of the civil code in the shortened form and leaves it to the name of the instrument in the full one, which is what “artikel 96 ... van Boek 6 van het Burgerlijk Wetboek” against “art. 6:96 BW” comes down to.

```
\newcommand\regulatory@src@number@plain[3]{\def#1{#3}}
\newcommand\regulatory@src@number@colon[3]{%
  \ifthenelse{\equal{#2}{}}{\def#1{#3}}{\def#1{#2:#3}}%
}
```

The ordinal keeps an ordinary space between the number and the word it belongs to. The regulation binds a label to its number with a space that does not break, but it writes its numbers after the label, `lid 6`, and never as an ordinal: *zesde lid* does not occur in it once. What binds *zesde* to *lid* in the national register is therefore not measured, and this bundle does not decide it.

```
\newcommand\regulatory@src@source@full@store[2]{%
  \protected@edef#1{\regulatory@src@source@full{#2}}%
}
\newcommand\regulatory@src@source@short@store[2]{%
  \protected@edef#1{\regulatory@src@source@short{#2}}%
}

\ExplSyntaxOn
\prop_new:N \g__regulatory_src_determiner_prop
\tl_new:N \l__regulatory_src_det_tl
\tl_new:N \l__regulatory_src_kind_tl
\cs_generate_variant:Nn \str_lowercase:n { V }

\NewDocumentCommand \newsourcedeterminer { m m m }
{
  \tl_set:Nx \l__regulatory_src_kind_tl { #1 / \str_lowercase:n { #2 } }
  \prop_gput:Nvn \g__regulatory_src_determiner_prop \l__regulatory_src_kind_tl { #3 }
}
```

Worked out before the citation is built rather than while it is: the entry decides first, then the kind of the instrument, then the default of the register. Doing it here rather than in an expandable macro keeps the lookup a plain lookup.

```
\cs_new_protected:Npn \regulatory@src@setdet #1
{
  \tl_set:Nx \l__regulatory_src_det_tl { \srcfield { #1 } { determiner } }
```

```

\tl_if_blank:VT \l__regulatory_src_det_tl
{
  \tl_set:Nx \l__regulatory_src_kind_tl { \srcfield { #1 } { kind } }
  \tl_set:Nx \l__regulatory_src_kind_tl
  {
    \regulatory@src@reg /
    \str_lowercase:V \l__regulatory_src_kind_tl
  }
  \tl_set:Nx \l__regulatory_src_det_tl
  {
    \exp_args:NNV \prop_item:Nn
    \g__regulatory_src_determiner_prop \l__regulatory_src_kind_tl
  }
}
\tl_if_blank:VT \l__regulatory_src_det_tl
{ \tl_set:Nx \l__regulatory_src_det_tl { \regulatory@src@defdet } }
\tl_set_eq:NN \regulatory@src@thisdet \l__regulatory_src_det_tl
}

\cs_new:Npn \regulatory@src@determiner #1 { \regulatory@src@thisdet }
\ExplSyntaxOff
\ExplSyntaxOn

\cs_new:Npn \regulatory@src@naam #1
{
  \tl_if_blank:eTF { \srcfield { #1 } { shortname } }
  { \srcfield { #1 } { citetitle } } { \srcfield { #1 } { shortname } }
}
\cs_new:Npn \regulatory@src@source@full #1
{
  \tl_if_blank:eF { \srcfield { #1 } { book } }
  { Boek~ \srcfield { #1 } { book } \c_space_tl van~ }
  \tl_if_blank:eF { \regulatory@src@determiner { #1 } }
  { \regulatory@src@determiner { #1 } ~ }
  \regulatory@src@naam { #1 }
}
\cs_new:Npn \regulatory@src@source@short #1
{
  \tl_if_blank:eTF { \srcfield { #1 } { abbreviation } }
  { \regulatory@src@naam { #1 } } { \srcfield { #1 } { abbreviation } }
}
\ExplSyntaxOff
\ExplSyntaxOn

```

14.7.4 Waar een register staat

Not here. A register is written in the language definition file of its language, `regulatory- $\langle language \rangle$ .def`, next to the words that language calls a provision by. Those two are one measurement: what says that German writes “Artikel 6 Absatz 1 Buchstabe a” in a citation is the same

reading of the same text that says it calls a paragraph an *Absatz*. Keeping them apart would mean one measurement landing in two files, where it can drift.

It also spares whoever writes one a second set of names. The language is dutch to babel and the register is dutch here, and dutch_eu is that same language writing in the legal order of the Union.

Those files are read for every language this bundle ships, whatever language the document is in, because a register belongs to the source and not to the document: a Dutch contract cites a German regulation in German. paragraaf 10 has the loading rule.

14.7.5 Meer dan één noemen

A citation may name a list of provisions or a run of them, which the source writes as “artikelen 101 en 102” and “artikelen 12 tot en met 15”. Both are written by the author rather than worked out here: `point={c,d}` is a list and `point={a--c}` is a run, and the register decides how either reads.

The far end of a range does not always read like the near one: the civil code is cited as “art. 6:162-164”, where the book is named once and the second number stands on its own. A run therefore formats its two ends separately, and every other level hands the same format twice.

Three things are counted apart because the register words them apart: the numbers, which each go through the format of their level; the words that join them; and the designation, which takes its plural as soon as there is more than one. Measured in the same regulation, with the non-breaking space where the source puts it: “artikelen 101 en 102”, “artikelen 12 tot en met 15”, “punten c) en e)”, “Buchstaben c und e”, “Absätze 1, 2 und 3”, “paragraphes 1 et 4”.

```
\ExplSyntaxOn
\seq_new:N \__regulatory_src_items_seq
\tl_new:N \__regulatory_src_joined_tl
\int_new:N \__regulatory_src_count_int

\cs_new_protected:Npn \regulatory@src@join #1#2#3#4#5#6
{
  \tl_clear:N \__regulatory_src_joined_tl
  \tl_if_in:nnTF { #2 } { -- }
    { \__regulatory_src_range:nnnn { #2 } { #3 } { #6 } { #5 } }
    { \__regulatory_src_list:nnn { #2 } { #3 } { #4 } }
  \tl_set_eq:NN #1 \__regulatory_src_joined_tl
}

\cs_new_protected:Npn \__regulatory_src_range:nnnn #1#2#3#4
{
  \seq_set_split:Nnn \__regulatory_src_items_seq { -- } { #1 }
  \int_set:Nn \__regulatory_src_count_int { 2 }
  \seq_pop_left:NN \__regulatory_src_items_seq \l_tmpa_tl
  \exp_args:NNV #2 \__regulatory_src_piece_tl \l_tmpa_tl
  \tl_put_right:NV \__regulatory_src_joined_tl \__regulatory_src_piece_tl
  \tl_put_right:Nn \__regulatory_src_joined_tl { #4 }
  \seq_pop_left:NN \__regulatory_src_items_seq \l_tmpa_tl
}
```

```

\exp_args:NNV #3 \l__regulatory_src_piece_tl \l_tmpa_tl
\tl_put_right:NV \l__regulatory_src_joined_tl \l__regulatory_src_piece_tl
}

\cs_new_protected:Npn \__regulatory_src_list:nnn #1#2#3
{
  \seq_set_split:Nnn \l__regulatory_src_items_seq { , } { #1 }
  \seq_set_map_x:NNn \l__regulatory_src_items_seq \l__regulatory_src_items_seq
    { \tl_trim_spaces:n { ##1 } }
  \int_set:Nn \l__regulatory_src_count_int
    { \seq_count:N \l__regulatory_src_items_seq }
  \int_step_inline:nnn { 1 } { \l__regulatory_src_count_int }
  {
    \int_compare:nNnT { ##1 } > { 1 }
    {
      \int_compare:nNnTF { ##1 } = { \l__regulatory_src_count_int }
        { \tl_put_right:Nn \l__regulatory_src_joined_tl { #3 } }
        { \tl_put_right:Nn \l__regulatory_src_joined_tl { ,~ } }
    }
    \tl_set:Nx \l_tmpa_tl { \seq_item:Nn \l__regulatory_src_items_seq { ##1 } }
    \exp_args:NNV #2 \l__regulatory_src_piece_tl \l_tmpa_tl
    \tl_put_right:NV \l__regulatory_src_joined_tl \l__regulatory_src_piece_tl
  }
}

\tl_new:N \l__regulatory_src_piece_tl
\cs_new:Npn \regulatory@src@items { \int_use:N \l__regulatory_src_count_int }
\ExplSyntaxOff

```

14.7.6 Een bron aanhalen

`\srcref` Cites $\langle key \rangle$, either as a whole or down to one of its provisions. The optional argument holds the provision — article, paragraph, point and subpoint — and stands before the key, as it does in `\cite`, so that a bracket in the sentence after the citation is not mistaken for it.

The star asks for the system that is not the default of the document, which is what a citation between brackets or in a footnote wants. It may also be named outright with system. The default of the document is the `sourcestyle` option, and it is `full`, since that is what the running text of a document is.

What it comes to is built up in `\regulatory@src@last` before it is typeset, so that the wording of a citation can be read back rather than only looked at. That is what the test suite asserts on: a citation in the wrong register fails nowhere, it simply stands there wrong.

```

\ExplSyntaxOff
\newcommand\regulatory@src@system{\regulatory@sourcestyle}

```

The Dutch names are meta keys, and the value of a meta key is braced on the way

through: a meta key sets its target by running the key parser again, so `onderdeel={c,d}` would arrive as two keys, of which the second is a key named `d` that does not exist. Measured: without the braces the list and range citations lost their second item and $\LaTeX$ said only that a key was unknown, naming a letter.

```
\ExplSyntaxOn
\keys_define:nn { regulatory / srcref }
{
  article .code:n = \def \regulatory@src@article { #1 } ,
  paragraph .code:n = \def \regulatory@src@paragraph { #1 } ,
  point .code:n = \def \regulatory@src@point { #1 } ,
  subpoint .code:n = \def \regulatory@src@subpoint { #1 } ,
  system .code:n = \def \regulatory@src@thissystem { #1 } ,
  date .code:n = \tl_set:Nn \l__regulatory_src_thisdate_tl { #1 } ,
  datum .meta:n = { date = {#1} } ,
  artikel .meta:n = { article = {#1} } ,
  lid .meta:n = { paragraph = {#1} } ,
  onderdeel .meta:n = { point = {#1} } ,
  onder .meta:n = { subpoint = {#1} } ,
}
\cs_new_protected:Npn \regulatory@src@setkeys #1
{ \keys_set:nn { regulatory / srcref } { #1 } }
\ExplSyntaxOff
```

`\sourcedate` The date a citation speaks as of. A reference to legislation is a reference to a *version* of it: the same article read a year apart is not necessarily the same article, and the Juriconnect standard says so by carrying the date in the reference itself. This bundle does not print the date and does not build the link yet, but it refuses to let a document leave it unsaid, because the alternative is a citation that silently means “whenever this was last typeset”. It is a warning and not an error on purpose: a document that cites without a date is defective, but it is not broken, and a package that cannot be added to an existing document without stopping it is a package that does not get added. The warning is said once, and names the command that answers it.

An entry may carry a `valid` field, which is the date the data of that entry was last checked. When the document speaks as of a later date than that, the entry cannot vouch for itself and says so, once per source. That is what makes a shared file of sources safe to use at all, wherever it is kept: the identifiers in it never age, the dates do, and a document that outruns the data it was handed is told rather than quietly given whatever the last update to that file happened to contain. This bundle ships no legislation itself and is not the place for it — a `citeertitel` changes on its own clock and a package on CTAN does not — but it is the place for the machinery that says when the two have come apart.

```
\ExplSyntaxOn
\tl_new:N \g__regulatory_src_date_tl
\tl_new:N \l__regulatory_src_thisdate_tl
\seq_new:N \g__regulatory_src_dated_seq
\bool_new:N \g__regulatory_src_nodate_bool
```

```

\regex_const:Nn \c__regulatory_src_date_regex { \A \d{4}-\d{2}-\d{2} \Z }

\msg_new:nnn { regulatory-sources } { date-not-iso }
{
  The-date~`#1'-is-not-of-the-form-YYYY-MM-DD.-\\
  Dates-are-compared-as-they-are-written,~so-another-form-would-compare-wrongly~
  rather-than-not-at-all.
}

\NewDocumentCommand \sourcedate { m }
{
  \regex_match:NnTF \c__regulatory_src_date_regex { #1 }
  { \tl_gset:Nn \g__regulatory_src_date_tl { #1 } }
  { \msg_error:nnn { regulatory-sources } { date-not-iso } { #1 } }
}

\cs_new_protected:Npn \__regulatory_src_checkdate:n #1
{
  \tl_if_empty:NT \l__regulatory_src_thisdate_tl
  { \tl_set_eq:NN \l__regulatory_src_thisdate_tl \g__regulatory_src_date_tl }
  \tl_if_empty:NNTF \l__regulatory_src_thisdate_tl
  {
    \bool_if:NF \g__regulatory_src_nodate_bool
    {
      \bool_gset_true:N \g__regulatory_src_nodate_bool
      \msg_warning:nn { regulatory-sources } { no-date }
    }
  }
  { \__regulatory_src_checkstale:n { #1 } }
}

\cs_new_protected:Npn \__regulatory_src_checkstale:n #1
{
  \tl_set:Nx \l__regulatory_src_valid_tl { \srcfield { #1 } { valid } }
  \tl_if_empty:NF \l__regulatory_src_valid_tl
  {
    \str_compare:eNeT \l__regulatory_src_thisdate_tl > \l__regulatory_src_valid_tl
    {
      \seq_if_in:NnF \g__regulatory_src_dated_seq { #1 }
      {
        \seq_gput_right:Nn \g__regulatory_src_dated_seq { #1 }
        \msg_warning:nnxxx { regulatory-sources } { stale-source }
        { #1 } { \l__regulatory_src_valid_tl } { \l__regulatory_src_thisdate_tl }
      }
    }
  }
}

\cs_generate_variant:Nn \str_compare:nNnT { eNeT }

\msg_new:nnn { regulatory-sources } { no-date }

```

```

{
  This-document-cites-legislation-without-saying-as-of-when.-\\
  Put~\iow_char:N\\sourcedate\{YYYY-MM-DD\}~in~the~preamble,~or~give~one~
  citation~its~own~with~date=YYYY-MM-DD.~Reported
}

\msg_new:nnn { regulatory-sources } { stale-source }
{
  The-source~`#1'~was~last~checked~on~#2,~and~this~document~speaks~as~of~#3.~\\
  Whatever~changed~in~between~is~not~in~this~citation.~Reported
}
\cs_new_protected:Npn \regulatory@src@checkdate #1
{ \__regulatory_src_checkdate:n { #1 } }
\cs_new_protected:Npn \regulatory@src@cleardate
{ \tl_clear:N \l__regulatory_src_thisdate_tl }
\ExplSyntaxOff

\newcommand\regulatory@src@thislabel{}
\newcommand\regulatory@src@label[2]{%
  \ifnum\regulatory@src@items>1
    \edef\regulatory@src@thislabel{%
      \regsrc@get{\regulatory@src@reg}{\regulatory@src@thissystem}{#2}}%
  \else
    \edef\regulatory@src@thislabel{%
      \regsrc@get{\regulatory@src@reg}{\regulatory@src@thissystem}{#1}}%
  \fi
}

\newcommand\regulatory@src@last{}
\newcommand\regulatory@src@part[1]{%
  \ifx\regulatory@src@parts\empty\else
    \protected@xdef\regulatory@src@last{\regulatory@src@last\regulatory@src@sep}%
  \fi
  \protected@xdef\regulatory@src@last{\regulatory@src@last#1}%
  \let\regulatory@src@parts\relax
}

```

Every level puts what it comes to in a macro of its own, worked out to the last token there and then, rather than straight into the citation. Worked out, because the designation of a level lives in one macro that every level writes to in turn, and a part that kept the macro rather than its text would read the word of whatever level came after it. Its own macro, because where the parts stand with respect to one another is a decision of the register and not of this module. Nearly every register names them from the outside in — the article, then the paragraph, then the point — and joins them with its separator. English does the reverse, and measured in the English text of the same regulation it is not close: “point (a) of Article 6(1)” twenty-eight times against nothing at all for either of the two arrangements the other registers use. So the deepest level comes first, each level is joined to the next with “of”, and the paragraph is not named but written into the article between brackets.

Two arrangements are therefore offered and the assemble key of a register picks one. A register that says nothing is arranged from the outside in, which is what all four of the others do.

```

\newcommand\regulatory@src@p@article{}
\newcommand\regulatory@src@p@paragraph{}
\newcommand\regulatory@src@p@paragraphnum{}
\newcommand\regulatory@src@p@point{}
\newcommand\regulatory@src@p@subpoint{}
\newcommand\regulatory@src@p@source{}

\newcommand\regulatory@src@add[1]{%
  \ifx#1\@empty\else\expandafter\regulatory@src@part\expandafter{#1}\fi
}

\newcommand\regulatory@src@assemble@forward{%
  \regulatory@src@add\regulatory@src@p@article
  \regulatory@src@add\regulatory@src@p@paragraph
  \regulatory@src@add\regulatory@src@p@point
  \regulatory@src@add\regulatory@src@p@subpoint
  \ifx\regulatory@src@parts\@empty\else
    \protected@edef\regulatory@src@p@source{%
      \regulatory@src@connective\regulatory@src@p@source}%
  \fi
  \regulatory@src@add\regulatory@src@p@source
}

```

The other way round, with one thing more to do than turning the order about: a paragraph that hangs on an article is written (1) and takes no word, while one that stands on its own keeps its word, “point (a) of paragraph 1”. Both occur in the same regulation, so the register hands over the shape of the attached one in attach and the level keeps its own wording for the other case.

```

\newcommand\regulatory@src@assemble@backward{%
  \ifx\regulatory@src@p@article\@empty\else
    \ifx\regulatory@src@p@paragraphnum\@empty\else
      \ifx\regulatory@src@attach\relax\else
        \protected@edef\@@attachnow{%
          \noexpand\regulatory@src@attach
          \noexpand\regulatory@src@p@article
          {\regulatory@src@p@article}{\regulatory@src@p@paragraphnum}}%
        \@@attachnow
        \let\regulatory@src@p@paragraph\@empty
      \fi
    \fi
  \fi
  \regulatory@src@add\regulatory@src@p@subpoint
  \regulatory@src@add\regulatory@src@p@point
  \regulatory@src@add\regulatory@src@p@paragraph
  \regulatory@src@add\regulatory@src@p@article
}

```

```

\regulatory@src@add\regulatory@src@p@source
}

\NewDocumentCommand \srcref { s O{ } m }{%
\beginingroup
\def\regulatory@src@article{%
\def\regulatory@src@paragraph{%
\def\regulatory@src@point{%
\def\regulatory@src@subpoint{%
\IfBooleanTF{#1}{%
\ifthenelse{\equal{\regulatory@src@style}{full}}{%
\def\regulatory@src@thissystem{short}%
}%
\def\regulatory@src@thissystem{full}%
}%
}%
\edef\regulatory@src@thissystem{\regulatory@src@style}%
}%
\regulatory@src@cleardate
\regulatory@src@setkeys{#2}%
\regulatory@src@checkdate{#3}%
\edef\regulatory@src@reg{\srcregister{#3}}%
\regsrc@set{\regulatory@src@reg}{\regulatory@src@thissystem}{separator}%
{\regulatory@src@sep}%
\edef\regulatory@src@defdet{%
\regsrc@get{\regulatory@src@reg}{\regulatory@src@thissystem}{determiner}}%
\regulatory@src@setdet{#3}%
\let\regulatory@src@parts\@empty
\gdef\regulatory@src@last{%
\let\regulatory@src@p@article\@empty
\let\regulatory@src@p@paragraph\@empty
\let\regulatory@src@p@paragraphnum\@empty
\let\regulatory@src@p@point\@empty
\let\regulatory@src@p@subpoint\@empty
\regsrc@set{\regulatory@src@reg}{\regulatory@src@thissystem}{and}%
{\regulatory@src@and}%
\regsrc@set{\regulatory@src@reg}{\regulatory@src@thissystem}{to}%
{\regulatory@src@to}%
\regsrc@set{\regulatory@src@reg}{\regulatory@src@thissystem}{connective}%
{\regulatory@src@connective}%
\regsrc@set{\regulatory@src@reg}{\regulatory@src@thissystem}{attach}%
{\regulatory@src@attach}%
\ifx\regulatory@src@article\@empty\else
\regsrc@set{\regulatory@src@reg}{\regulatory@src@thissystem}{articlenum}{\@fmt}%
\def\@numfmt##1##2{\@fmt##1{\srcfield{#3}{book}}{##2}}%
\def\@numrest##1##2{\@fmt##1}{##2}%
\expandafter\regulatory@src@join\expandafter\regulatory@src@piece
\expandafter{\regulatory@src@article}{\@numfmt}%
{\regulatory@src@and}{\regulatory@src@to}{\@numrest}%
\regulatory@src@label{articleone}{articlemany}%

```

```

\protected@edef\regulatory@src@p@article{%
\regulatory@src@thislabel~\regulatory@src@piece}%
\fi
\ifx\regulatory@src@paragraph\@empty\else
\regsrc@set{\regulatory@src@reg}{\regulatory@src@thissystem}{paragraphnum}{\@@fmt}%
\expandafter\regulatory@src@join\expandafter\regulatory@src@piece
\expandafter{\regulatory@src@paragraph}{\@@fmt}%
{\regulatory@src@and}{\regulatory@src@to}{\@@fmt}%
\regulatory@src@label{paragraphone}{paragraphmany}%
\regsrc@set{\regulatory@src@reg}{\regulatory@src@thissystem}{paragraph}{\@@compose}%
\expandafter\@@compose\expandafter\regulatory@src@p@paragraph
\expandafter{\regulatory@src@piece}{\regulatory@src@thislabel}%
\protected@edef\regulatory@src@p@paragraph{\regulatory@src@p@paragraph}%
\regsrc@set{\regulatory@src@reg}{\regulatory@src@thissystem}{attachnum}{\@@afmt}%
\ifx\@@afmt\relax
\let\regulatory@src@p@paragraphnum\regulatory@src@piece
\else
\expandafter\regulatory@src@join\expandafter\regulatory@src@p@paragraphnum
\expandafter{\regulatory@src@paragraph}{\@@afmt}%
{\regulatory@src@and}{\regulatory@src@to}{\@@afmt}%
\fi
\fi
\ifx\regulatory@src@point\@empty\else
\regsrc@set{\regulatory@src@reg}{\regulatory@src@thissystem}{pointnum}{\@@fmt}%
\expandafter\regulatory@src@join\expandafter\regulatory@src@piece
\expandafter{\regulatory@src@point}{\@@fmt}%
{\regulatory@src@and}{\regulatory@src@to}{\@@fmt}%
\regulatory@src@label{pointone}{pointmany}%
\regsrc@set{\regulatory@src@reg}{\regulatory@src@thissystem}{point}{\@@compose}%
\expandafter\@@compose\expandafter\regulatory@src@p@point
\expandafter{\regulatory@src@piece}{\regulatory@src@thislabel}%
\protected@edef\regulatory@src@p@point{\regulatory@src@p@point}%
\fi
\fi
\ifx\regulatory@src@subpoint\@empty\else
\regsrc@set{\regulatory@src@reg}{\regulatory@src@thissystem}{subpointnum}{\@@fmt}%
\expandafter\regulatory@src@join\expandafter\regulatory@src@piece
\expandafter{\regulatory@src@subpoint}{\@@fmt}%
{\regulatory@src@and}{\regulatory@src@to}{\@@fmt}%
\regulatory@src@label{subpointone}{subpointmany}%
\regsrc@set{\regulatory@src@reg}{\regulatory@src@thissystem}{subpoint}{\@@compose}%
\expandafter\@@compose\expandafter\regulatory@src@p@subpoint
\expandafter{\regulatory@src@piece}{\regulatory@src@thislabel}%
\protected@edef\regulatory@src@p@subpoint{\regulatory@src@p@subpoint}%
\fi
\fi
\regsrc@set{\regulatory@src@reg}{\regulatory@src@thissystem}{source}{\@@fmt}%
\@@fmt\regulatory@src@p@source{#3}%
\regsrc@set{\regulatory@src@reg}{\regulatory@src@thissystem}{assemble}%
{\regulatory@src@assembler}%
\ifx\regulatory@src@assembler\relax
\let\regulatory@src@assembler\regulatory@src@assemble@forward

```



```

\fi
\regulatory@src@assembler
\endgroup
\regulatory@src@last
}
\ExplSyntaxOn

```

\regulatory@src@messages What the module says when it stops. Every one of these names the record it was reading, since a bib file that is wrong is wrong in one entry and the rest of it is fine.

```

\msg_new:nnn { regulatory-sources } { unknown-type }
{
  Source~'#2'~is~of~type~'#1',~which~no~one~declared.~Declare~it~with~
  \iow_char:N\newsourcetype,~or~use~one~of:~
  \prop_map_function:NN \g__regulatory_src_required_prop
  \__regulatory_src_type_name:nn
}
\cs_new:Npn \__regulatory_src_type_name:nn #1#2 { ~#1 }

\msg_new:nnn { regulatory-sources } { missing-field }
{
  Source~'#2'~of~type~'#3'~has~no~'#1'.~A~source~of~that~type~cannot~be~
  written~out~without~it.
}
\msg_new:nnn { regulatory-sources } { duplicate-key }
{ Source~'#1'~was~already~declared.~The~second~one~would~replace~it. }
\msg_new:nnn { regulatory-sources } { stray-field }
{ Field~'#1'~stands~outside~any~entry. }
\msg_new:nnn { regulatory-sources } { no-value }
{ Field~'#1'~was~given~no~value. }
\msg_new:nnn { regulatory-sources } { file-not-found }
{ There~is~no~file~'#1'~to~read~sources~from. }
\msg_new:nnn { regulatory-sources } { unclosed }
{ Source~'#1'~is~still~open~at~the~end~of~'#2'.~A~closing~brace~is~missing. }
\msg_new:nnn { regulatory-sources } { unreadable }
{
  Line~#1~of~'#2'~is~none~of~the~four~shapes~this~reader~knows:\\
  ~~#3\\
  An~entry~opens~with~an~at~sign,~the~type~and~the~key;~a~field~reads~
  name~=={value};~an~entry~closes~with~a~brace~of~its~own.~Each~of~the~three~
  stands~on~a~line~of~its~own,~and~a~value~is~in~braces~and~not~in~quotes:~
  this~is~a~restricted~syntax~and~not~a~BibTeX~parser.
}
\ExplSyntaxOff

```

14.8 Markdown syntaxuitbreiding

markdown lets a document add rules to the grammar of its reader. This file adds one, and it exists because of a limit that has no way around it in Markdown itself: an identifier can be attached to a heading and to a bracketed span, and a bracketed span may not

contain another span or a link written with brackets. A paragraph that carries a label and cites a definition therefore cannot be written as one span, and the label has to be pushed onto the opening words of the sentence.

With this extension the identifier stands on its own, before the text it labels:

```
1 1. {#lid: lorem} Een [persoonsgegevens](#pg) wordt verwerkt, zie <#lid:eerder>.
```

What is added is syntax, not typesetting: a renderer decides how a construct that the reader already recognises is set, and no renderer can make the reader recognise something new. The extension is read by markdown itself, in Lua, before any of that.

The interface is a table with three fields. Both versions are checked, and `grammar_version` for equality rather than for a lower bound, so an update of markdown that changes its grammar makes this file an error instead of a surprise.

```
1 local regulatory_syntax = {
2   api_version = 2,
3   grammar_version = 4,
```

`finalize_grammar` is handed the reader, whose grammar can be added to. The pattern is an identifier between braces, spelled the way Pandoc spells one, and what it produces is a call of a renderer that this bundle defines in `regulatory-md`.

The characters allowed in an identifier are the ones a label of this bundle is made of: letters, digits, and the punctuation that separates a type from a name, `art: lorem` and `ex2-lid: lorem`.

```
5 finalize_grammar = function(reader)
6   local labelchar = lpeg.R("az", "AZ", "09") + lpeg.S(":-_")
7   local label = lpeg.P("{#") * lpeg.C(labelchar^1) * lpeg.P("}")
8   / function(identifier)
9     return {"\\markdownRendererRegulatoryLabel{", identifier, "}"}
10  end
```

The rule is spliced in front of the one that reads links and emphasis, so that a brace opening an identifier is seen before anything else can claim it. A brace is not a character the reader stops at on its own, which is what the second call says: without it the brace would be swallowed by the run of ordinary text around it and the pattern would never be tried.

Only `{#` starts the pattern, so a brace anywhere else in the text is left alone.

```
12 reader.insert_pattern("Inline before LinkAndEmph", label, "RegulatoryLabel")
13 reader.add_special_character("{")
14 end
15 }
16
17 return regulatory_syntax
```

14.9 regulatory.4ht

This is the tex4ht configuration of paragraaf 12, which is a source of its own rather than a module of the bundle: it is not a package, it is read by tex4ht and by nothing else, and it only exists at all because an HTML run replaces the machinery the printed headings are built on. It identifies itself with `\ProvidesFile` all the same, so that a document can tell whether it was found: tex4ht ships no configuration for this package, and a conversion without one fails silently rather than loudly (paragraaf 12).

14.9.1 What this fixes

This file is what emits the structure of a regulatory document in HTML. It replaces `\article` and `\para` outright, so it does that in either of the two heading branches the package can take (see below), and without it there is no structure at all: an article becomes a paragraph with a bold run in it, and the output carries no `<h1>..<h6>` and no `<section>`. Measured on example1-nl, four articles:

	<code><section></code>	<code><h<n>></code>	make4ht
without this file	0	0	exit 0
with it	5	5	exit 0

Note both exit codes. The conversion *succeeds* without this file and produces a document with no headings; what falls away does not announce itself.

A document that does not declare `\DocumentMetadata` hits a hard failure on top of that, because regulatory then builds `\article` with titlesec's `\titleclass` and tex4ht replaces the sectioning machinery titlesec hooks into, leaving `\titleformat` nothing to attach to. Measured on the same document with neither this file nor `\DocumentMetadata`: make4ht exits 1 on 'No format for this command', 'Missing number' and 'Illegal unit of measure', and four faults arrive at once, none of which announces itself:

1. `titlesec` errors per heading and leaks its spacing argument into the text: '0ptEerste artikel'. Measured: five occurrences of '0pt' in the output.
2. It also consumes the first letter of the following text as an argument, so 'Een genummerde paragraaf' comes out as '0ptE' plus 'en genummerde paragraaf'. The text is not misformatted, it is broken apart.
3. Article numbers vanish. In a legal instrument the number is not decoration; it is the address by which a clause is cited.
4. `\aref` still emits `<a href="#article.1">`, but nothing emits an element with that id. Measured: three such links, zero matching ids.

That branch is the narrower case, not the reason this file exists. A document that attaches anything declares `\DocumentMetadata` already and never reaches `titlesec` – and still needs this file for everything above.

The `paras` environment needs nothing here: it is an ordinary `enumitem` list and `tex4ht` already renders it with correct numbering.

14.9.2 Two traps

This file must not use `\makeatletter` or `\makeatother`. The `@` sign is already a letter when the file runs (measured: `\catcode`\@` is 11), so `\makeatletter` is redundant and the matching `\makeatother` is destructive: it turns `@` back into an ordinary character for the rest of the package, so every `\rref@...` internal silently becomes garbage. The symptom is a build that hangs and then dies inside `color.4ht` with `Missing \begin{document}` – an error naming a file that has nothing to do with the cause.

(An earlier version of this comment explained the catcode by saying `tex4ht` inputs a .4ht ‘the moment it sees `\ProvidesPackage`’. That reason is wrong; only the conclusion held. See the next paragraph for what was measured.)

The redefinitions are made directly, not deferred. Measured 2026-09-05 against the split package (`regulatory` + `-struct` + `-ref` + `-attachments`), with a stub .4ht per package reporting its own state: `tex4ht` defers each file past the whole `usepackage` chain, not merely past the package that names it – `regulatory-struct.4ht` already sees `\attachdocument`, which `regulatory-attachments` defines afterwards. So `\article` and `\para` exist by the time we get here and we are overwriting them. `\AtEndOfPackage` looks like the careful choice and is not: it never fires, because by then no package is being loaded, and the redefinitions then silently never happen at all.

Ordering caveat if this file is ever split along the package lines: `tex4ht`’s own `titlesec.4ht` runs between the clusters: `regulatory.4ht` and `regulatory-struct.4ht` before it, `regulatory-ref.4ht` and `regulatory-attachments.4ht` after. An `\article/\para` fix must land in the first two. Keeping everything in one file keeps it on the safe side of that line.

14.9.3 Which heading branch this patches

`regulatory-struct` builds `\article` through `titlesec`’s `\titleclass` only when `\ParseLaTeXeHeading` is absent. A bare `\DocumentMetadata` – no testphase needed – loads `latex-lab-testphase-sec-template.sty` and defines it, so such a document takes the `\DeclareInstanceheading` branch and never reaches `titlesec`. Measured under both `pdflatex` and `lualatex`, 2026-09-05.

That does not make this file optional for those documents. It replaces `\article` outright, so it is what emits the markup in either branch. Measured on one document with two articles, `\DocumentMetadata` on line 1, differing only in whether this file was present:

	$\langle h \rangle$	regulatory-*	make4ht
without this file	0	0	exit 0
with it	2	6	exit 0

The id='article.N' anchors are hyperref's and appear either way. So without this file the conversion *succeeds* and produces a document with no headings at all – the same shape as the titlesec finding on the PDF side, where a clean build was no evidence either. What falls away does not announce itself.

```
\immediate\write-1{regulatory.4ht 2026-09-05}
```

```
\NewConfigure{regarticle}{4}
\NewConfigure{regpara}{4}
\NewConfigure{regextlink}{2}
\Configure{regextlink}{false{}}
```

The anchor id must be the one \aref links to. hyperref's \refstepcounter sets \@currentHref to 'article.N' – exactly the '#article.1' the broken output was already pointing at – so emitting that as the element id is what turns those dead links into live ones.

```
\newcommand\reg@anchorid[1]{\HCode{ id="#1"}}
```

Close any open paragraph before block-level markup, or the <section> is emitted inside a <p>. \IgnorePar stops tex4ht opening a fresh one immediately.

```
\newcommand\reg@blockmode{\ifvmode\IgnorePar\fi\EndP}
```

A <section> opened by \article has to be closed by the next \article or at the end of the document. These are sectioning commands, not environments – there is no \endarticle to hook – so an explicit flag tracks it.

```
\newif\ifreg@inarticle \reg@inarticlefalse
\newif\ifreg@inpara \reg@inparafalse
```

```
\newcommand\reg@closepara{%
  \ifreg@inpara \reg@blockmode\HCode{</section>}\reg@inparafalse \fi
}
\newcommand\reg@closearticle{%
  \reg@closepara
  \ifreg@inarticle \reg@blockmode\HCode{</section>}\reg@inarticlefalse \fi
}
```

```
\def\article{\@ifstar\reg@article@star\reg@article@plain}%
%
% The sectioning signature is \article*[toc-entry]{title}, whichever of the two
% branches of regulatory-struct built it -- titlesec, or the heading instance of
% latex-lab that a \DocumentMetadata brings with it -- so both forms are accepted
% here and no existing document needs changing.
\newcommand\reg@article@plain[2][{}]{%
  \reg@closearticle
  \reg@blockmode
  \refstepcounter{article}%
  % \@currentlabel is what \label captures. Keep it the bare number so \aref
  % and \ref print the same thing the heading does.
  \def\@currentlabel{\thearticle}%
}
```

```

\def\regarticle\reg@anchorid{\@currentHref}\HCode{>}%
\def\regarticle\GetTranslation{Article}\ \thearticle.\c:regarticle
#2%
\def\regarticle
\reg@inarticletrue
\par\ShowPar
}

```

Starred: an unnumbered heading, so the counter is deliberately not stepped

```

% and no anchor is emitted -- there is no number for anything to cite.
\newcommand\reg@article@star[2][]{%
\reg@closearticle
\reg@blockmode
\def\regarticle\HCode{>}%
\def\regarticle\c:regarticle
#2%
\def\regarticle
\reg@inarticletrue
\par\ShowPar
}

```

14.9.4 The \para heading

\para is used, despite appearances. Grepping a legal source for \para finds mostly \param, and grepping for ^\para finds nothing because the calls are indented. It is what a document uses for sub-headings within an article, and leaving it out leaves those broken in a document whose articles all look correct.

Nested inside the article's <section> rather than beside it: the sub-heading belongs to the article above it, and a flat structure would tell anything reading the outline otherwise.

```

\def\para{\@ifstar\reg@para@star\reg@para@plain}

\newcommand\reg@para@plain[2][]{%
\reg@closepara
\reg@blockmode
\refstepcounter{para}%
\def\@currentlabel{\thearticle.\thepara}%
\def\regpara\reg@anchorid{\@currentHref}\HCode{>}%
\def\regpara\thearticle.\thepara.\c:regpara
#2%
\def\regpara
\reg@inparatrue
\par\ShowPar
}

\newcommand\reg@para@star[2][]{%
\reg@closepara
\reg@blockmode
\def\regpara\HCode{>}%

```

```

\begin{regpara}\c:regpara
#2%
\end{regpara}
\reg@inparatrue
\par\ShowPar
}

```

14.9.5 Cross-document references

A reference to another document must not become a hyperlink in HTML.

In the PDF a link to another document is right whenever the reader holds both: it opens a document they were sent. On the web that no longer follows. A document cited with `\refdocument` or `\masterdocument` is commonly one issued to a counterparty rather than published, and a link to it would point at a file that is not there. A dead link in a legal text is worse than no link, since it suggests the reader is being denied something.

This is a choice, not a limitation. A publisher who does put every cited document on the web wants the opposite, and gets it by redefining `\rref@linkpr` after this file is read.

The reference itself stays: the sentence still says which article of which agreement it means, which is what a reader needs in order to look it up in the copy they were sent. Only the anchor goes.

regulatory already draws the distinction we need. `\rref@current@url` is set by `\rref@setlink` and is empty for a reference inside this document, non-empty for one resolved through `zref-xr` from another document's `.aux`. One test, at the single point where the package turns a reference into a link.

Note this is a decision about the *output format*, which is why it belongs here and not in the source. Writing `\aref*` in the `.tex` would drop the link from the PDF as well, where it is wanted. A reference whose anchor this format cannot produce must not look like a link either. In HTML a `paras` item carries no anchor under the name `\aref` points at (see the limitation above), so linking it would promise a jump that does not happen. The wording is what carries the meaning – “het in eerste lid genoemde bedrag” tells the reader exactly where to look – and a link that silently does nothing subtracts from that rather than adding to it.

The counter is the discriminator, and it separates the two cases cleanly: `para` is the sectioning command, which this file *does* anchor, while `parasi` and `parasii` are the enumerate levels of the `paras` list, which it cannot. Testing the counter rather than the type keeps a linked sub-heading linked.

The two levels are recognised by their name beginning with `paras` rather than by being named one by one. `regulatory-struct` declares the list with two levels today; naming the counters here would tie this file to that number, and a third level would stop being recognised without anything saying so, which is the failure this whole file is written against.

The five letters matter: `para` is four, so the heading, which does have an anchor, falls outside the test and keeps its link. Shortening the comparison to the first four characters would silently take the headings with it.

14.9.6 Links to another document

A reference to a provision of another document is not linked by default. The other document is a PDF as far as this run knows: whether it is published as HTML as well is a property of how a family is published, not of the document, and a link to a file that is not there is worse than no link at all.

That is a default, not a rule, and a family that *is* published as HTML wants those links. Hence `\Configure{regextlink}{true}{.html}` in the `cfg` of such a family, which turns them into an `\href`. The second argument is what the URL of the other document is missing: that URL is the name the artifact was declared under, without an extension, since it names a document rather than a file. A link to `example2-nl` reaches nothing in a browser, and a link to `example2-nl.html` does.

It belongs in the `cfg` that `make4ht -c` reads, not in the preamble of the document: the configuration of a package is applied when that package is read, which is after the preamble has been seen, so a `\Configure` there is overwritten by the default below. Measured both ways: without it a reference to an article of another document comes out as `Artikel 1`, `van Voorbeeld Twee`, and with it as `<a href='example2-nl.html#article.1'>`.

The switch stands per conversion, not per document referred to. A family in which some of the referred documents are published and others are not has no good setting: off loses the links that would work, on writes the ones that do not. No such family has come up, so no shape has been invented for it; if one does, the setting to make is a property of the artifact rather than of the run.

14.9.7 Labels that never reached zref

`tex4ht` replaces `\label` with one of its own (`latex.4ht`, at `\def\label`), which keeps the original under `\:label` and calls it only when no sectioning unit is open; inside one it calls `\l:bel` instead, and that one writes the anchor and the `\newlabel` by itself. It does not run the `label` hook of the kernel, which is where `zref-clever` writes the `zref` label that this bundle refers by.

The consequence was measured on the examples: an aux file written by `pdflatex` holds eleven `zref` labels, the same document converted by `make4ht` holds none. Nothing failed. Within one document it goes unnoticed, since `tex4ht` resolves its own `\newlabel`; between two documents it does not, because `\externaldocument` reads `zref` labels and there were none to read, so every reference to another document came out as two question marks.

Only the branch that skips the hook is patched, so a label outside a sectioning unit is not recorded twice. The name is reached through `\csname`, since it holds a colon and this file cannot count on that being a letter.

```
\expandafter\let\expandafter\reg@ht@lbel\csname l:bel\endcsname
\expandafter\def\csname l:bel\endcsname#1{%
  \reg@ht@lbel{#1}%
  \zref@wrapper@babel\zref@label{#1}%
}

\newif\ifreg@anchorless
\ExplSyntaxOn
```



```

\str_new:N \l__regulatory_html_counter_str
\cs_new_protected:Npn \reg@checkanchorless
{
  \reg@anchorlessfalse
  \str_set:Nx \l__regulatory_html_counter_str { \rref@current@counter }
  \str_if_eq:eeT { \str_range:Nnn \l__regulatory_html_counter_str { 1 } { 5 } }
    { paras } { \reg@anchorlesstrue }
}
\ExplSyntaxOff

\def\reg@extlinkyes{true}

\def\rref@linkpr#1{%
  \begingroup
  \reg@checkanchorless
  \edef\reg@refurl{\rref@current@url}%
  \edef\reg@extlink{\a:regextlink}%
  \ifreg@anchorless
    % No anchor by that name in this output, so no link.
    \endgroup#1%
  \else
    \ifx\reg@refurl\empty
      \endgroup\href{\rref@current@fullurl}{#1}%
    \else
      \ifx\reg@extlink\reg@extlinkyes
        % A family published as HTML: the other document is a file next to this one,
        % under the name it has here plus whatever suffix that publication gives it.
        \endgroup\href{\rref@current@url\b:regextlink\hashchar\rref@current@anchor}{#1}%
      \else
        % Another document, not published as HTML, so no link.
        \endgroup#1%
      \fi
    \fi
  \fi
}

```

14.9.8 A known limitation: references to an item of paras

References to an article work; references to a paragraph do not. A legal text cites both – ‘het in eerste lid genoemde bedrag’ is as common as ‘zie artikel 3’ – so this is worth knowing about rather than discovering.

The cause is a name clash, not a missing anchor. For a list item tex4ht emits its own generic anchor (x1-31), while hyperref’s \refstepcounter has meanwhile recorded \@currentHref as parasi.1.1. regulatory stores that recorded value in zref, and \aref links to it – so the link points at a name nothing carries. Verified on a two-item probe:

ids in the output	x1-2r1, article.1, x1-31
link target	parasi.1.1

What was tried and does *not* work: hooking \refstepcounter to emit an extra anchor

with `\@currentHref`, both directly and deferred to `\AtBeginDocument`. Neither errors and neither produces an anchor – `\refstepcounter` runs inside the construction of the item’s label, and output written there is discarded. A working fix has to hook the item itself, not the counter.

Left unfixed deliberately rather than half-fixed: an approach that silently does nothing is worse in this file than an absence that is written down. How a package should attach a hyperref anchor to an `enumitem` item is a question for the `tex4ht` maintainer.

14.9.9 Attachments

`\attachdocument` and `\documentattachment` do two things: embed a file inside the PDF, and link to that embedded copy. Neither half survives this output format, and both fail as an error rather than as an absence.

`\embedfile` counts what it has embedded in `\g__pdfmanagement_EmbeddedFiles_int`, which exists only while `pdfmanagement` is active – it is not, here, because there is no `\DocumentMetadata` in an HTML run. The link is built from `\pdfannot_link_begin:nnw`, an `l3pdfannot` command with no meaning outside a PDF backend. A single footnote carrying an attachment therefore produced six ‘Undefined control sequence’ errors and `make4ht` stopped.

An HTML page has no attachments and cannot acquire any, so there is nothing to translate the construct into. Both halves become no-ops and the link text is typeset as text – the same principle the cross-document references follow above: a link this format cannot honour must not look like a link.

Where that text argument happens to be a public URL the reader still sees where to go. Do not turn it into an `\href` on the strength of such a case: `\documentattachment` passes a document name in the same position, and a blanket hyperlink would point that at nothing.

The replacement is made at the two public commands, not at `\embedfile` and `\regulatory@attachmentlink` underneath them. Neutralising those two looks tighter and does not hold: `\embedfile` is set up again by its own package at begin-document, so a `\let` placed in a hook here is overwritten and the errors come back – measured, four of them left. Redefining the public commands removes the only paths that reach `\embedfile` at all (`\regulatory@embedonce` is called from `\documentattachment` and now-here else), so nothing downstream can put it back.

Parameters are taken outside the hook and the hook only `\lets`. Writing `\renewcommand...#2` inside a hook body does not survive the round of parameter reduction the token list passes through: `#2` arrives as a literal digit, so the footnote reads its own parameter number instead of the text.

```
\newcommand\reg@attachdocument@text[3][]{#3}
\newcommand\reg@documentattachment@text[2]{#2}
\AtBeginDocument{%
  \let\attachdocument\reg@attachdocument@text
  \let\documentattachment\reg@documentattachment@text
}
```

14.9.10 The definitions list

The article that defines a contract's terms came out as one flat paragraph: every term and its description separated by nothing but whitespace. For the one article whose job is to distinguish a term from its meaning, that is the worst place in the document to lose the distinction.

It comes from KOMA-Script's `labeling`, which the legal documents use to list the definitions explicitly:

```
\begin{labeling}{Overeenkomst van opdracht}
  \item[Algemene Voorwaarden] \describe{terms}
  \item[Consument] \describe{consumer}
```

`tex4ht` configures `itemize`, `enumerate` and `description`, but nothing ships a configuration for `labeling` – so it falls through to the generic handling and the item structure is lost.

Worth recording because it is where the search naturally goes first: the glossary style is not involved. `regulatory` prints with `style=almtree`, and changing that has no effect at all here – verified by redefining the style and by forcing `style=altlist` onto the print command. Neither changed anything, because these documents do not print a glossary at this point; they enumerate the terms by hand.

A list of terms with their meanings is a description list, so `labeling` is configured as one. Modelled on `tex4ht`'s own description configuration in `docbook.4ht`: five arguments – open list, close list, start item, after label – with `\end:itm` carrying the close tag of the previous item, because a `<dd>` can only be closed once the next `<dt>` arrives or the list ends.

```
\ConfigureList{labeling}%
  {\EndP\HCode{<dl class="regulatory-definitions">}}%
  \PushMacro\end:itm
  \global\let\end:itm=\empty
  {\PopMacro\end:itm \global\let\end:itm \end:itm
  \EndP\HCode{</dd></dl>}\ShowPar}
  {\end:itm \global\def\end:itm{\EndP\HCode{</dd>}}}%
  \HCode{<dt class="regulatory-term">}}
  {\EndP\HCode{</dt>}\HCode{<dd class="regulatory-definition">}\par\ShowPar}

\AtEndDocument{\reg@closearticle}
```

14.9.11 Default markup

`<section>` plus a real heading, not a `<div>` with a styled first line. The number sits in its own `<span>` so a stylesheet can set it apart, while staying inside the `<h2>` – a screen reader should announce "Artikel 1. Wettelijke kaders" as one heading, because that is what it is.

`h2` rather than `h1`: the document title is the `h1` and an article sits under it, which keeps the heading outline valid. Emitting headings at all is the whole point; emitting them at the wrong level would trade one accessibility problem for another.

```

\Configure{regarticle}
{ \HCode{<section class="regulatory-article">}}
{ \HCode{<h2 class="regulatory-article-heading"><span class="regulatory-article-number">}}
{ \HCode{</span> }}
{ \HCode{</h2>}}

\Configure{regpara}
{ \HCode{<section class="regulatory-para">}}
{ \HCode{<h3 class="regulatory-para-heading"><span class="regulatory-para-number">}}
{ \HCode{</span> }}
{ \HCode{</h3>}}

```

14.9.12 Translations

This file used to declare “Article” for English and Dutch itself, with `\providetranslation`, deferred to `\AtBeginDocument` because `regulatory-struct` loads translations well after this file is read. Two things were wrong with it. `\providetranslation` is not of translations at all but of translator, which those two packages renamed around each other in 2016; measured, translations alone leaves it undefined. So deferring did not avoid an undefined control sequence, it moved it to `\begindocument`, where the braced arguments are typeset instead. And every document of this suite happens to load glossaries, which pulls translator in, so the conversion came out clean here and nowhere else — the green was a property of what the author loads beside this bundle.

The declarations are gone rather than corrected: `regulatory-struct` declares the fallback and each language file declares its own, and since every shipped language file is loaded whatever the document speaks, Dutch is there to be found. Reported by the sibling sessions of Xerdi’s Documentation Project, who read it in the HTML of the manual.

Register

Ieder commando en iedere omgeving van deze bundel, met de bladzijde waarop hij beschreven wordt *cursief* en de bladzijde waarop hij gedefinieerd wordt onderstreept. Welke van de drie soorten namen een commando is — publiek, uitbreiding of intern — staat in paragraaf 2.9.

Symbols		I		\newsrcetype 25, <u>117</u>	
\@ifnameinlink	<u>84</u>	\ifregulatory@alldefs . .	<u>69</u>	\newsrcetypealias	<u>25</u>
\@ifrrefcap	<u>84</u>	\ifregulatory@bibtogls . .	<u>69</u>	\Nref	<u>17, 89</u>
A		\ifregulatory@legacy . .	<u>69</u>	\nref	<u>17, 89</u>
alldefs (option)	<u>5</u>	\ifregulatory@markdown . .	<u>69</u>	O	
\Aref	<u>17, 92</u>	\ifregulatory		options:	
\aref	<u>17, 92</u>	@nameinlink	<u>69</u>	alldefs	<u>5</u>
\aref@postlink@setup <u>91, 103</u>		\ifsourceexists	<u>25, 124</u>	attachmentlink	<u>5</u>
\aref@setcurrent	<u>91</u>	L		bib2gls	<u>4</u>
\article	<u>13, 73</u>	legacy (option)	<u>5</u>	defstyle	<u>5</u>
\attachdocument	<u>21, 102</u>	\loadextdefs	<u>16, 104</u>	legacy	<u>5</u>
attachmentlink (option) . .	<u>5</u>	\loadglsdefs	<u>16, 82</u>	md	<u>4</u>
\autocitedefs	<u>45, 112</u>	\loadsources	<u>24, 122</u>	nameinlink	<u>5</u>
B		M		sourcestyle	<u>5</u>
\begincompaction	<u>93</u>	\markdownRendererDl		P	
bib2gls (option)	<u>4</u>	DefinitionBegin	<u>110</u>	\para	<u>13, 73</u>
\bubblesort	<u>94</u>	\markdownRendererDlItem	<u>107</u>	paras (env.)	<u>13, 74</u>
D		\markdownRendererFenced		\printdefs	<u>16, 81</u>
definitions (env.)	<u>15</u>	DivAttributeContext		\ProvidesRegulatory	
defstyle (option)	<u>5</u>	Begin	<u>110</u>	Language	<u>76</u>
\describe	<u>16, 81</u>	\markdownRenderer		provisions (env.)	<u>13, 74</u>
\documentattachment	<u>21, 101</u>	HeadingOne	<u>106</u>	R	
\documentfootnote	<u>21, 103</u>	\markdownRendererLink	<u>107</u>	\refdocument	<u>19, 104</u>
\documentlabel	<u>21, 102</u>	\markdownRendererOl		\regulatory-description	<u>80</u>
E		Begin	<u>106</u>	\regulatory-labeling . .	<u>80</u>
environments:		\markdownRendererOlItem		\regulatory@attach	
definitions	<u>15</u>	WithNumber	<u>106</u>	@annot	<u>99</u>
externals	<u>15</u>	\markdownRenderer		\regulatory	
paras	<u>13, 74</u>	RegulatoryLabel	<u>108</u>	@attachmentlink	<u>100</u>
provisions	<u>13, 74</u>	\markdownRendererTilde	<u>108</u>	\regulatory	
externals (env.)	<u>15</u>	\masterdocument	<u>19, 105</u>	@checkenvironment . . .	<u>78</u>
F		md (option)	<u>4</u>	\regulatory@defstyle . .	<u>71</u>
\findendlist	<u>93</u>	N		\regulatory@embed	<u>98</u>
G		nameinlink (option)	<u>5</u>	\regulatory@embedonce	<u>101</u>
\GetTranslation	<u>72</u>	\newartifact	<u>95</u>	\regulatory@extdefs	
H		\newdefinition	<u>16, 82</u>	@nohyper	<u>104</u>
\hashchar	<u>83</u>	\newsources	<u>25, 119</u>	\regulatory@ifembed . . .	<u>101</u>
		\newsourcedeterminer	<u>27</u>	\regulatory@ifmodule . . .	<u>72</u>
		\newsourcfieldalias	<u>25</u>	\regulatory@load	
		\newsourceregister	<u>27</u>	@languages	<u>77</u>

<code>\regulatory@md@dlitem</code>	109	<code>\regulatorysetup</code>	71	<code>\setjuncto</code>	18, 85
<code>\regulatory@md</code>		<code>\Rref</code>	17, 89	<code>\setlastconjunction</code>	18, 84
<code>@dlitemend</code>	110	<code>\rref</code>	17, 89	<code>\setmiddleconjunction</code>	
<code>\regulatory@md@link</code>	108	<code>\rref@get</code>	88		18, 84
<code>\regulatory@md@nodefs</code>	113	<code>\rref@init@lang@article</code>	86	<code>\setrangeconjunction</code>	18, 84
<code>\regulatory@md@yaml</code>	111	<code>\rref@init@lang@para</code>	86	<code>\SignatureField</code>	116
<code>\regulatory@sign@annot</code>	113	<code>\rref@init@lang@sub</code>	86	<code>\signaturefield</code>	23, 116
<code>\regulatory@sign</code>		<code>\rref@labellist</code>	92	<code>\sourcedate</code>	25, 131
<code>@appearance</code>	115	<code>\rref@lang</code>	87	<code>sourcestyle (option)</code>	5
<code>\regulatory@sign@place</code>	114	<code>\rref@number</code>	91	<code>\srcfield</code>	25, 123
<code>\regulatory@sourcestyle</code>	71	<code>\rref@set</code>	88	<code>\srcref</code>	26, 130
<code>\regulatory@src</code>		<code>\rref@setcurrent</code>	90	<code>\srcregister</code>	123
<code>@messages</code>	137	<code>\rref@setup</code>	30, 87	<code>\srctype</code>	25, 123
<code>\RegulatoryLoadLanguage</code>		S		U	
.....	30, 76	<code>\setconjunction</code>	18, 84	<code>\unsetjuncto</code>	18, 85