

Package regulatory^{*}

Erik Nijenhuis
erik@xerdi.com

2026-09-10

This file is maintained by **Xerdi**.
Bug reports can be opened at
<https://github.com/Xerdi/regulatory>.

Abstract

The regulatory package is written for legal professionals in a broad sense. It provides the structures such a document is built from — articles, paragraphs, points and definitions — without taking any of $\LaTeX$ away.

Referring within the legal domain is a good deal harder than it looks, so this package refers the way $\LaTeX$ does: by labelling what is referred to and naming the label. The `\rref`, `\nref` and `\aref` families word the reference from there, in Dutch or in English.

Definitions are kept in one $\BibTeX$ file that serves every document citing the same terms, and they are cited with the `\gls` family of glossaries, which this package builds on rather than replaces. Articles, paragraphs, points and definitions of *another* document written with this package are referred to the same way, so two documents can name each other's provisions and share each other's terms — terms and conditions and a maintenance agreement, say. The document referred to can be embedded in the PDF file as an attachment, so that the reader has it in hand.

^{*}This document corresponds to regulatory 1.0.0, released on 2026-09-10.

Contents

1 Usage	4
2 Before you start	7
2.1 Engines and prerequisites	7
2.2 Definitions	7
2.3 HTML	7
2.4 Attachments	8
2.5 Languages	8
2.6 PDF conformance	8
2.7 Known limitations	8
2.8 What the tests establish	9
2.9 Public, extension and internal	10
3 Migration to 1.0	11
4 Structure	12
5 Definitions	14
6 References	16
6.1 Conjunction	17
7 Interdocument references	18
7.1 Which link opens an attachment	20
8 Signatures	22
9 External sources	23
10 Language support	28
10.1 English definitions	30
10.2 Dutch definitions	33
10.3 German definitions	37
10.4 French definitions	39
11 Markdown	41
11.1 Labels and references	41
11.2 An identifier of its own	42
11.3 Definitions in Markdown	42
12 HTML	44

13 Tests	46
13.1 The test documents	46
13.2 What the suite reads back	51
13.3 PDF conformance	53
13.4 The test files	55
14 Implementation	66
14.1 regulatory-struct	66
14.2 regulatory-defs	76
14.3 regulatory-ref	80
14.4 regulatory-attachments	92
14.5 regulatory-md	102
14.6 regulatory-sign	110
14.7 regulatory-sources	113
14.8 Markdown syntax extension	134
14.9 regulatory.4ht	136
Index	146

1 Usage

This bundle writes PDF, so the engine is `pdflatex` or `lualatex`. There is an HTML conversion as well, which runs on `make4ht` and needs a configuration of its own; see section 12.

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

Listing 1: `main.tex`

`regulatory` itself is a wrapper and nothing more: it loads the six modules below and hands every option to the first of them. Each module can be loaded on its own, and each asks for what it needs, so the order below is the dependency chain rather than a requirement.

- `regulatory-struct` the structures of section 4. This module is the base of the bundle: it holds the options of all of them and reads `regulatory.cfg`.
- `regulatory-defs` the definitions and definition lists of section 5.
- `regulatory-ref` the reference macro families of section 6 and the language support of section 10.
- `regulatory-attachments` the interdocument references and PDF attachments of section 7, which loads `regulatory-ref` and `regulatory-defs` in turn.
- `regulatory-sign` the signature fields of section 8. It emits nothing until one of its two commands is used.
- `regulatory-sources` the citations of legislation outside the document of section 9.

A seventh module, `regulatory-md`, comes on top of these as soon as `markdown` is loaded, whichever order the two are in (see section 11). The HTML conversion is not a module but a `tex4ht` configuration, `regulatory.4ht`, which `make4ht` finds by itself.

Every option is a key of the `/regulatory` family, administered by `regulatory-struct` for the whole bundle. The example above passes none, which means these defaults hold:

- `bib2gls true` definition lists are built with `bib2gls`. Pass `bib2gls=false` to have them written as `TEX` code instead.
- `md false` loading `markdown` is left to the document. With `md` the package loads it, which it has to do before `enumitem`.
- `alldefs false` a definition list holds the definitions the document uses. With `alldefs` it holds every declared definition, which is what Terms and Conditions want, where not every term necessarily appears in the text.

<code>legacy</code>	<code>false</code>	the conjunction of section 6. With <code>legacy</code> the older one is used, see <code>\setjuncto</code> .
<code>nameinlink</code>	<code>true</code>	the hyperlink of a reference is drawn around the label as a whole. With <code>nameinlink=false</code> it is drawn around the number only.
<code>defstyle</code>	<code>alttree</code>	the glossary style <code>\printdefs</code> falls back on when it is given none; see section 5.
<code>sourcestyle</code>	<code>full</code>	whether a citation of an external source is written out (<code>full</code>) or abbreviated (<code>short</code>); see section 9. It answers to <code>bronstijl</code> as well, and to <code>voluit</code> and <code>verkort</code> as values. A value that is none of the four is refused where it is given rather than emptying every lookup that reads it.
<code>attachmentlink</code>	<code>goto</code>	how an attachment of section 7 is opened from the page: with an embedded go-to action (<code>goto</code>) or with a file attachment annotation (<code>attachmentlink=annotation</code>). The two differ in what they mean, in which viewers act on them and in what the PDF standards ask of them; section 7.1 sets them side by side. Either way the file is also in the attachment pane of the viewer, which needs neither.

Coloured borders around hyperlinks are hidden with `\hypersetup{<hidelinks>}`, since this package loads `hyperref` itself. A file `regulatory.cfg` next to the document is read before the options given to the package are processed, so it can set other defaults for a whole directory of documents at once; the log says which of the two happened. It is read while the package loads, so it may require a package a house depends on as well.

That is worth knowing for one thing this release takes away. `xifthen` used to be a prerequisite and is not one any more, since nothing here uses a command of it that plain `ifthen` does not have. It pulled in `calc`, which this bundle never used either, so `calc` is no longer there for the asking. A document that writes `\widthof`, `\heightof` or one of the `calc` arithmetic forms was leaning on that and stops with an undefined command; it loads `\usepackage{<calc>}` itself from here on, or a whole directory of documents answers it once in `regulatory.cfg`.

A document that attaches other documents (see section 7) needs the PDF management of $\text{\LaTeX}$, which is activated with `\DocumentMetadata{<>}` in front of `\documentclass`. Without it, every attachment is typeset as plain text and the package says so once.

A source that is also converted to HTML declares it conditionally, and this is the one line to write. Under `tex4ht` the same declaration pulls in the list code of $\text{\LaTeX}$ and leaves the conversion nothing to turn a `paras` item into, so the items come out as running text; section 12 has the measurement. `tex4ht` defines `\HCode` before it reads the document, so that is what the line asks. Every example of this bundle is written this way, and a run that gets it wrong is told so rather than left to the result.

```
\ifdefined\HCode\else\IfDocumentMetadataTF{<>}{\DocumentMetadata{<>}}\fi
\documentclass[dutch]{article}
```

```
\usepackage{regulatory}
```

Listing 2: One source, both outputs

The `\IfDocumentMetadataTF` guard leaves a declaration that is already there alone, which is what lets the same source be built once more with a PDF standard declared in front of it; that is how the cases of section 13.3 are made.

The example of listing 1 can be generated to PDF as follows:

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

Listing 3: Commandline examples

A document with definition lists adds a step between the $\TeX$ runs:

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

Listing 4: Commandline with definitions

Under `latexmk`, `bib2gls` or `makeglossaries` can be run in a terminal of its own: `latexmk` sees the files change and typesets the document again.

2 Before you start

What this bundle needs, what it is measured on, and what it cannot do. Everything here is treated at length further on; this section is the short answer to whether regulatory fits a document before it is written around it.

2.1 Engines and prerequisites

The engine is `pdflatex` or `lualatex`: both produce PDF directly, and every example of this bundle is typeset with both on every run of the suite. Nothing here depends on either of the two, and no difference between them is known to this manual. `xelatex` and a DVI route are not tested and are not claimed.

The `md` route is the exception: `\markdownInput` reads its source through the Lua interpreter under `lualatex` and reaches for shell escape under `pdflatex`, so a Markdown document is more comfortable under `lualatex`.

All $\TeX$ dependencies of the bundle are available in $\TeX$ Live and $\text{MiK}\TeX$: `enumitem`, `fntcount`, `glossaries` and `glossaries-extra`, `hyperref`, `scrextend`, `titlesec` (only while the new heading interface is unavailable), `translations`, `xcolor`, `xstring` and the `zref` family. Two packages that earlier releases pulled in are gone; see section 3.

Outside $\TeX$ there is one prerequisite, and only on one route: the default definition route runs `bib2gls`, which needs Java. Section 5 has the three routes that do not.

2.2 Definitions

Four routes lead to a definition list, and the choice decides what has to run between the $\mathbb{E}\TeX$ runs. Section 5 has them in full; in one line each:

`bib2gls` the default. One $\text{B}\mathbb{E}\TeX$ file serves every document that cites the same terms, sorted and selected from, and it is what lets `\masterdocument` pull another document's definitions in. It needs Java.

`bib2gls=false` `makeglossaries` and `makeindex` instead, reading a file of `\newglossaryentry`. No Java, still a program between the runs, and of the four the least exercised.

`\newdefinition` definitions declared in the document itself. One $\mathbb{E}\TeX$ run and nothing else. No other document can cite them, and they are printed in the order they were declared in.

by hand the list written out with `\describe`, which decides what it lists and in what order. The entries still come from one of the three above.

2.3 HTML

A document converts with `make4ht`, which drives `tex4ht`. `tex4ht` carries no configuration for this package, so this one ships `regulatory.4ht` next to the package files: a document that finds `regulatory` finds it too, and without it a conversion succeeds and produces a document with no headings and no sections at all.

One condition: `\DocumentMetadata` must *not* be active in an HTML run, or the list markup of paras goes away. Listing 2 is the line to write, and section 12 has the measurements.

2.4 Attachments

Embedding another document in the PDF file needs the PDF management of $\text{\LaTeX}$, activated with `\DocumentMetadata` in front of `\documentclass`. Without it every attachment is typeset as plain text and the package says so once.

How the attachment is opened from the page is the `attachmentlink` option, and neither value is right for everybody: `goto` is the semantically correct embedded go-to action, which poppler does not implement, and `attachmentlink=annotation` is a file attachment annotation, which every measured viewer opens. Section 7.1 sets the two side by side. Either way the file is in the attachment pane of the viewer, which needs neither.

2.5 Languages

Dutch and English are fully supported. German and French ship the designations and the citation register of an act of the Union, both measured, and *not* the wording of an internal reference: a German document gets German words in the English arrangement. Any other language falls back on English entirely, and is told so once. Section 10 has the table and what each case looks like on the page.

2.6 PDF conformance

Which standard a document can carry is not a property of this package but of the document: of its tagging, of what it attaches, and of what it refers to. Section 13.3 is the measurement, run by veraPDF pinned by digest, and it is what the claims below are worth. In summary, for the documents of this bundle:

Profile	Measured	On what
PDF/A-1	out of reach	forbids embedded files outright
PDF/A-2b	pass	plain, and signed
PDF/A-2b	fail	with a non-PDF attachment
PDF/A-2a + UA-1	pass	tagged, with <code>attach-css=false</code>
PDF/A-3	out of reach	referring to provisions of another document
PDF/A-3a + UA-1	pass	tagged, attaching a PDF
PDF/A-4	pass	tagged, with <code>attach-css=false</code>
PDF/A-4f + UA-2	pass	tagged, attaching anything

Read that as: *this document in this configuration was measured to pass*, and never as “regulatory is PDF/A compatible”. The rows differ in one thing each, which is what makes them worth anything.

2.7 Known limitations

PDF/A-3 a document that refers to the *provisions* of an external document cannot reach it: the `/GoToR` link is written with a file specification that fails ISO 19005-3 rule 6.8-4,

and that specification is not written by this package. Attaching another document and citing its definitions costs nothing. See sections 7 and 13.3.

Languages German and French cannot word an internal reference or the name of another document; see section 10. Neither has a national citation register.

Sorting `\loadglsdefs` sorts under the Dutch collation `n1-NL` whatever language the document is in. A document in another language that wants its own collation prints its list with `\printunsrtglossary` after a `\GlsXtrLoadResources` of its own, or lays it out by hand with `\describe`.

HTML the conversion needs `\DocumentMetadata` to be absent, which is the same declaration the attachments need, so one source cannot be converted and carry attachments in the same run. The `almtree` definition style carries no list markup at all in HTML; `labeling` and `description` do. See section 12.

Markdown a Markdown source holds no `TEX`: a backslash is printed rather than obeyed, silently. Only the constructs of section 11 have a counterpart, and a bracketed span may not hold a second span or a bracketed link, which is what the identifier of section 11.2 is for.

`\aref` with more than one label, `nameinlink` has no effect and the hyperlink is drawn around the numbers alone.

`\loadsources` the bib reader takes a restricted, line-based syntax and is not a `BETEX` parser; a perfectly ordinary bib file may well be refused. Section 9 states the grammar.

Juriconnect this release records and guards the date a citation speaks as of, and does not print it and does not build a Juriconnect reference. See `\sourcedate` in section 9.

Legislation no legislation ships with this bundle. It holds the machinery for citing instruments and not the instruments: a citation title changes on its own clock and a package on CTAN does not.

2.8 What the tests establish

One thing is worth saying this early, because it is what the claims in this manual rest on. A document that is generated without an error is not thereby correct. Nearly everything this bundle can get wrong comes out as a finished document all the same: a citation worded in the wrong register, a definition that quietly did not arrive, a label that resolved to nothing, a heading that became a bold sentence.

So the suite does not check that a build succeeds. It reads the result back — references and the labels they resolve to, definitions and the order a program sorted them into, the wording of forty citations word for word, the sections and headings of the HTML conversion, the annotations and structure elements in the PDF file itself, and the warnings the package did and did not give. Two documents are there to *fail* and two more to *warn*, because a guarantee is worth what its refusal is worth. Section 13 is the whole of it.

2.9 Public, extension and internal

Three kinds of name occur in this manual, and they carry different promises for the 1.x releases.

Public API what an ordinary document writes: the environments `paras`, `provisions`, `definitions` and `externals`; `\article` and `\para`; the `\rref`, `\nref` and `\aref` families and the conjunction commands; `\loadglsdefs`, `\newdefinition`, `\printdefs` and `\describe`; `\refdocument`, `\masterdocument`, `\newartifact` and the `\document...` commands; `\signaturefield`; `\loadsources`, `\newsources`, `\srcref`, `\srcfield`, `\srctype`, `\ifsourceexists` and `\sourcedate`; `\autocitedefs` and the Markdown constructs; and the package options. These keep working through 1.x, and where one has to change, the old spelling stays as an alias.

Extension API what a language file, a citation register or an integration writes: `\ProvidesRegulatoryLanguage` and `\RegulatoryLoadLanguage`; `\rref@setup` and the format commands it is given, `\rref@reformat@...`, `\rref@label@...` and `\rref@group@braced`; `\newsourceregister`, `\newsourcedeterminer`, `\newsourcetype`, `\newsourcetypealias`, `\newsourcetypealias` and the `\regulatory@src@...` formats a register is built from; `\regulatorysetup` and `regulatory.cfg`; and `\regulatory@ifmodule`. Several of these carry an at sign, which is not an oversight: they are meant to be written in a `.def` file, which is read with the at sign made a letter. They are supported through 1.x as they stand.

Internal everything else, including every `\regulatory@...`, `\artifact@...` and `\rref@...` not named above, and every l3 name with a double underscore. These are documented in section 14 because the implementation is documented, and not because they may be used; they change without notice. Two artifact fields belong here as well and are described under `\refdocument`: `referred`, which the package sets itself, and `url`, which is reserved and does nothing.

3 Migration to 1.0

What a document written for an earlier release runs into. Everything below is a change that can be seen from outside the package; the rest of 1.0 is addition.

- `calc` **This is the one that stops a build.** `xifthen` is no longer a prerequisite, since nothing here used a command of it that plain `ifthen` does not have. It pulled in `calc`, which this bundle never used either, so `calc` is no longer there for the asking. A document that writes `\widthof`, `\heightof` or one of the `calc` arithmetic forms was leaning on that and now stops with an undefined command. The fix is `\usepackage{calc}` in the document, or `\RequirePackage{calc}` in a `regulatory.cfg` for a whole directory of them. This manual is one of the documents it caught, and asks for `calc` itself.
- Markdown the module used to be loaded with the `hybrid` option of `markdown`, which let every backslash through, and sources written that way wrote their labels and references as `TEX`. `markdown` has deprecated that option. Without it a backslash is *printed* rather than obeyed, so such a source loses its references without a single error being raised. Everything that needed it is written in Markdown now; see section 11, and read an older source through once.
- Languages an unsupported language no longer borrows the words of the last language file that happened to be loaded. Measured on the same Spanish document: an earlier release gave it Dutch headings and silently dropped the designations from its references, where 1.0 words everything in English and says once which language it has no definitions for. A document that was quietly relying on the first behaviour looks different and now says why.
- Structure one package became a bundle. Earlier releases shipped a single `regulatory.sty`; 1.0 ships eight, of which `regulatory` loads six, plus `regulatory.4ht` and the language files. A document that says `\usepackage{regulatory}` notices nothing, and one that installs by hand copies more files.
- Signatures `regulatory-sign` is where `xdp-sign` went. `\SignatureField` still answers, so a document written for that package keeps working when it moves over. A signature no longer costs the document its PDF/A-2b conformance.
- Source keys the types and fields of section 9 are named in English, and every Dutch name is an alias on top of it, so a `bib` file written either way is read either way. Two names changed, and they differ in what became of the old one. The option `branstijl` became `sourcestyle` and is aliased: `branstijl` still answers, and so do the values `voluit` and `verkort`, so a document that used it needs no change. The translation key subparagraph became `point` and is *not* aliased, since nothing outside this bundle had asked for it yet: a document that asked for that translation by name asks for `point` now.

4 Structure

This package provides familiar structures without taking anything of $\LaTeX$ away. The three levels are called *article*, *paragraph* and *point*, after the Dutch “artikel, lid en onderdeel” this package started out with.¹

`\article` `\article{<title>}` and `\para{<title>}` are the two headings. They are defined as separate macros and formatted with `titlesec`, or, whenever the new $\LaTeX$ heading interface is available, with a heading instance. Both take part in the table of contents, at the level of subsections and subsubsections.

`paras (env.)` The `paras` environment holds the two levels below an article, built with `enumitem`. A paragraph is labelled `\thearticle.\arabic*`, so it repeats the number of the article it belongs to, and a point is labelled `\abalphnum{\arabic*}`.

That `\abalphnum` comes from `fmtcount`² and is what lets an article run past 26 points: `\alph` stops there, whereas `\abalphnum` of, say, 125 is ‘du’.

`provisions (env.)` Not every document is divided into articles. The `provisions` environment is a flat enumeration, numbered within the `\section` it stands in rather than within an article, for a text that has one level and no headings of its own.

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

\article{...}
...

\para{...}
...

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

¹The English names are shortened where $\LaTeX$ already speaks for them: a heading of the second level is `\para` rather than `\paragraph`, and its list is `paras` rather than paragraphs. Which of “section” and “paragraph” names which level differs per jurisdiction, and neither reading is the one $\LaTeX$ means by it.

²The Dutch language definition is part of `fmtcount` itself now. Update the package when `fc-dutch.def` is missing.

`\end{provisions}`

See listings 5 and 6 in section 13.4 for more $\text{\LaTeX}$ examples.

5 Definitions

Definitions are cited with the `\gls{<label>}` family of `glossaries-extra`, which this module loads and configures.

`definitions (env.)` So that terms, abbreviations and definitions do not collide, this package adds two glossary types. Definitions of this document go under `definitions`; definitions of another document go under `externals` (see section 7).

Where the definitions come from is a question of its own, and the answer is not forced. Four routes are open, and how the entries arrive is independent of how the list is printed:

`bib2gls` the default. `\loadglsdefs{<src>}` reads a `BibTeX` file, which `bib2gls` sorts and selects from. *For:* one file serves every document that cites the same terms, the listing is sorted against the dictionary of the language, and only the definitions the document actually uses appear — or all of them with `alldefs`. It is also what lets `\masterdocument` pull another document's definitions in. *Against:* it needs Java and a `bib2gls` run between the two `TeX` runs.

`bib2gls=false` `\loadglsdefs{<src>}` then reads a file of `\newglossaryentry` instead. *For:* no Java. *Against:* the sorting is left to `makeindex`, so there is still a program to run in between, and of the four this is the route the package is exercised on least.

`\newdefinition` `\newdefinition{<label>}{<name>}{<description>}` declares a definition in the document itself. *For:* no second file and no second program at all — one `TeX` run is enough. *Against:* the definitions belong to this document, so no other document can cite them, and they are printed in the order they were declared rather than sorted.

by hand the list itself is written out, with `\describe{<label>}` where each description belongs. The entries still come from one of the three routes above. *For:* the article decides which terms it lists, in what order, and what stands between them. *Against:* the list is maintained by hand, so a new definition does not appear in it of its own accord.

Only the second route asks for a program between the `TeX` runs. The other two hand over entries that are already in the order they are to be printed in, which is why `\printdefs` reaches for a different printing command there — see section 14.

How the listing is laid out is a second choice, independent of the first. `\printdefs` takes the style as an optional argument and the `defstyle` option sets the default for the document. Three are on offer: `alttree`, the default that `glossaries` brings, which sets a name above its description; `labeling`, which aligns every description on the width handed to `\printdefs`; and `description`, which leaves the alignment to the class. Any other style `glossaries` knows can be named just as well. Writing the list out by hand, the fourth route above, uses the same two environments, so the layout matches whichever way the

list is produced.

That choice reaches further than the printed page. Measured on a two-entry list converted with `make4ht`: `alttree` carries no list markup at all, while `labeling` and `description` both give a `<dl>` with a `<dt>` and a `<dd>` per definition. A document that is published as HTML as well (section 12) has a reason to pick one of the two.

`\printdefs` `\printdefs{<width of text>}` prints the definition list. The `<width of text>` is the widest `[<style>]` name the list has to align on, which is what the `labeling` style uses; the other styles `{<width of text>}` ignore it.

`\describe` `\describe{<label>}` prints the description of one definition where it stands, which is the way to lay out a definitions article by hand rather than as a list. It adds the anchor point the hyperlinks of `\gls` need, which is what distinguishes it from printing the description with `\glsentrydesc`. A list built this way is aligned by whatever surrounds it, so `\glssetwidest` is only of interest when that surrounding is a `labeling` environment of one's own.

`\newdefinition` `\newdefinition` declares one definition in the document itself, under the same type and `{<label>}` category as the ones read from a file, so it lands in the same listing.
`{<name>}`

`{<description>}` `\loadglsdefs{<src>}` reads a `BiBTeX` file into the definitions type under the definitions category. The definitions the document uses are listed when `\printdefs` is called, or all of them with the `alldefs` option. The listing is sorted under the Dutch collation, `n1-NL`, whatever language the document is in — see section 2.7, which says what to do about it.
`\loadglsdefs`

`\loadextdefs` `\loadextdefs[<category>]{<src>}` reads the definitions of another document, under a category of their own so that definitions from different sources stay apart. `\masterdocument` calls it with the right category already, so a document rarely calls it itself. It belongs to regulatory-attachments and is described in section 7.

6 References

References to articles, paragraphs and points are built on `zref`, for which every structure of section 4 is configured. `zref` on its own offers less control over the wording than `cleveref` does.

`\rref` That is why this bundle words its own, starting with `\rref{<label>}`, which refers to
`\Rref` an article the way `\ref` refers to a `\section`. The family has four members:

`\rref` Starting with a lowercase letter and with a hyperlink.

Article	Paragraph	Point
2	(1)	(a)

`\rref*` Starting with a lowercase letter and without a hyperlink.

Article	Paragraph	Point
2	(1)	(a)

`\Rref` Starting with an uppercase letter and with a hyperlink.

Article	Paragraph	Point
2	(1)	(a)

`\Rref*` Starting with an uppercase letter and without a hyperlink.

Article	Paragraph	Point
2	(1)	(a)

The examples above already show where this differs from `\zref`: a reference is not the number the heading carries. The paragraph `ex1-lid:lorem` is numbered ‘2.1’ on the page and referred to as (1).

`\nref` The `\nref{<label>}` family adds the designation to the number, and has the same four
`\Nref` members as `\rref`. The table below shows `\Nref` alone.

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

Where `\nref` differs from `\zcref` is that the designation need not stand in front of its number: Dutch writes a paragraph as *<ordinal> <designation>* and a point as *<designation> <letter>*, and the language decides which.

`\rref` and `\nref` carry one level each. `\nref` gives a paragraph its designation but not the article it belongs to, and a reference to a paragraph that does not name its article names half of what it means. `\aref{<labels...>}` writes the whole reference: every level down
`\Aref` to the one named, in the order and the wording the language asks for. It takes more than one label and joins them, as a list, article 2(2), points (a), (c) and (d), of Example One³, or as a run, article 2(2), points (a) to (d), of Example One. One limitation: with more than one label, `nameinLink` has no effect, and the hyperlink is drawn around the

³Example One

numbers alone. The same labels read as the language of the document does:

```
\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}.
```

Zie artikel 2, tweede lid, onderdelen a, c en d, of Example One en artikel 2, tweede lid, onderdelen a tot d, of Example One.

What does not follow the language is the wording that names *another* document: that is the `ref` label of section 7, which is written once and stays as it is written.

6.1 Conjunction

```
\setmiddleconjunction Labels are joined through zref, and which words join them is set the way cleveref sets
  {\format} it:
\setlastconjunction \setmiddleconjunction{, }
\setrangeconjunction \setlastconjunction{ \GetTranslation{and} }
\setconjunction \setrangeconjunction{ \GetTranslation{to} }
  {\middle} \setconjunction{, }{ \GetTranslation{and} }{ \GetTranslation{to} }
  {\last}
  {\range}

\setjuncto Two more switch to the older notation and back. \setjuncto makes the last conjunc-
\unsetjuncto tion ‘ jo.\ ’ from that point on, and \unsetjuncto puts ‘ and ’ back. Both overwrite
whatever \setlastconjunction was given earlier, so a document that set its own word-
ing restores it with \setlastconjunction{\value} rather than with \unsetjuncto. The
package option legacy calls \setjuncto at once.
```

7 Interdocument references

A document written with this package refers to the articles, paragraphs, points and definitions of another one written with it.

Before building a document architecture on this. A document that refers to the provisions of an external document cannot reach **PDF/A-3**. Every such reference becomes a `/GoToR` action pointing at a file specification that carries `/AFRelationship/Unspecified`, has no `/EF` and is not listed in `/AF`, which fails ISO 19005-3 rule 6.8-4. The specification is written for the link, by the PDF management of $\LaTeX$ and not by this package, so the profile stays out of reach until that is settled upstream. Measured on the two example documents: `example1-nl` makes eleven such references and carries one such specification, and fails PDF/A-3b on that rule; `example2-nl` refers to no provisions of another document, *attaches* one and cites its definitions, has no `/GoToR` at all, and reaches PDF/A-3a. So attaching another document and citing its definitions is not what costs the profile — referring to its articles, paragraphs and points is. Every other profile in section 13.3 is unaffected either way.

`\refdocument` `\refdocument` in the preamble is what makes the articles, paragraphs and points of another document referable; `\aref` then reads its labels through `\zexternaldocument` of `{<name>}` `zref-xr`. Both documents may well use the same label, so every label of the other one is given a prefix, `ext-` unless another is named: `lid:lorem` is referred to as `ext-lid:lorem`. The prefix is for references and not for definitions, which `\gls` cites under their own label.

`\masterdocument` `\masterdocument` is the whole link rather than half of one. It does what `\refdocument` `{<prefix>}` does and three things more:

- `{<name>}`
 1. references with the `\aref` family;
 2. definitions of that document, cited with the `\gls` family;
 3. the document named in the sentence that cites it;
 4. a footnote at the first reference or definition, with the document itself embedded in the PDF file as an attachment.
- `{<opts...>}`

A document may have more than one master document, as this manual has:

```
\newcommand\definitionlabel[1]{~(see #1)}
\masterdocument[ex1-]{example1-en}{
  path=../test/,
  defs=example1,
  author=E. Nijenhuis,
  subject=Example One,
  description=The first example document,
  ref label=of Example One,
  def label=\definitionlabel
}

\masterdocument[ex2-]{example2-en}{
```

```

path=../test/,
defs=example2,
author=E. Nijenhuis,
subject=Example Two,
description=The second example document,
ref label=of Example Two,
def label=\definitionlabel
}

```

The third argument of `\refdocument` and `\masterdocument` holds the fields below. A field this package does not know is an error where it is given rather than a value stored under a name nothing reads back.

- `name` the first argument of the macro by default. It is the name the aux file of the other document is read under.
- `filename` the name with `.pdf` after it by default, for a document whose file is not named after it.
- `path` `./` by default, put in front of both the name and the filename, for documents kept in another directory.
- `ref label` the wording that names the other document in a sentence, as in “article 2 of the general terms”. It is empty by default, and `\documentlabel` then composes one from the subject: `\GetTranslation{of the} \artifactssubject {#1}`. Not every language can compose it; see section 10, where German and French say so out loud.
- `def label` what stands behind a definition that comes from the other document. The default is `-(\GetTranslation{see} #1)`. Its argument is the subject, with the footnote of the first occurrence attached to it.
- `footnote` `true` by default, so that the first reference or definition carries a footnote with the document attached to it. `false` leaves the footnote out.
- `footnote label` what that footnote says. Its argument is the attachment, with the subject as the text it is offered under; by default the macro prints that argument and nothing else.
- `defs` the definitions of the other document, loaded under it, so that a definition cited from it names where it comes from.
- `author` the author of the embedded file, which a PDF viewer shows beside the attachment.
- `subject` the name this document calls the other one by. It is what `ref label` and `def label` fall back on, and it is the description of the embedded file.
- `description` a longer description of the embedded file, shown by a viewer beside the attachment.

Two more fields are accepted and are not for a document to set; see section 2.9 for what that means for a release. `referred` records that an artifact has had its footnote, which `\documentfootnote` sets and a document that sets it by hand only uses to suppress a footnote it never saw. `url` is reserved: it is stored, it is readable with `\artifacurl`, and nothing in this bundle reads it. It was meant to name where the other document may be found, in the footnote beside the attachment, and that is not implemented. The name is kept free for exactly that.

<code>\documentlabel</code> <code>{\label}</code> <code>\documentfootnote</code> <code>[\link text]</code> <code>{\label}</code> <code>\documentattachment</code> <code>{\label}</code> <code>{\link text}</code> <code>\attachdocument</code> <code>[\description]</code> <code>{\filename}</code> <code>{\link text}</code>	A reference or a definition that comes from another document names that document, and the first one of either carries a footnote with the document embedded in it. Three commands do that between them: <code>\documentlabel</code>⁴ words the name, <code>\documentfootnote</code> places the footnote, and <code>\documentattachment</code> embeds the file and offers it under the text it is given. Any other file is embedded with <code>\attachdocument</code>, which needs no artifact of its own. These commands can also be called by hand. For example, <code>\documentfootnote{example2}</code>, which gives: ‘⁵’.
--	---

7.1 Which link opens an attachment

The `attachmentlink` option decides what `\documentattachment` puts on the page. The file itself is embedded either way and appears in the attachment pane of the viewer, which needs no link at all; what differs is what happens where the text stands.

`goto`, the default, makes that text a hyperlink carrying an embedded go-to action, which names a file *inside this document*. That is the thing there is to say, and it is the one action poppler does not implement. Measured in the shared library rather than read out of a manual: it names `GoTo`, `GoToR`, `Launch` and `Named` among its actions and `GoToE` nowhere, and `FileAttachment` among its annotations. poppler is the engine under most viewers on a free desktop, so there that link does nothing.

`attachmentlink=annotation` places a file attachment annotation over the same text instead, which Acrobat and poppler both act on. It shows nothing of its own — the text stays the only thing on the page — and it is the annotation that opens.

	goto	annotation
PDF construct	embedded go-to action	file attachment annotation
opens in Acrobat	yes	yes
opens in poppler	no	yes
in the attachment pane	yes	yes
structure element	Link	Annot
link text inside it	yes	no, beside it
measured PDF/A-4f	pass	pass
measured PDF/UA-2	pass	pass

The two viewer rows are what the two constructs are; the two structure rows are measured, and they are why the default did not change for this release. A file attachment is

⁴This one words the name for a reference and not for a definition, which has a `def label` of its own.

⁵Example Two

a markup annotation, and PDF/UA asks that one be enclosed in an Annot structure element. The annotation route puts it there, and the text it stands beside remains a sibling of it rather than its content, where the goto route makes that text the content of its Link element. Measured on the same document: goto gives a Link with two kids, the marked content of the text and the object reference; annotation gives an Annot with one. Joining the two on the annotation route is a job of its own and is not in this release.

Do not take a validator's word for that either way. The veraPDF this manual is measured with reported nothing when the Annot element was missing altogether and an older one did, so the suite reads the structure tree out of the PDF file itself rather than asking for a verdict; section 13.2 has that assertion.

What is left is a question about documents rather than about PDF. A reference a reader meets in the flow of a sentence is not the same offer as one that stands beside it, and goto is the route that makes the offer in the sentence. A document whose readers are on a free desktop and who are meant to open the attachment from the page says `attachmentlink=annotation` and gets the other trade.

8 Signatures

An agreement is signed, and a PDF can carry the field to sign it in. The regulatory-sign module draws such a field and puts a signature widget over it, so that a reader who opens the document in a viewer that can sign is offered the field where the line is.

It comes from xdp-sign of Xerdi's Documentation Project and is maintained here, so that a document that needs a signature does not depend on a package that is in no distribution.

`\signaturefield` Draws a signature field of $\langle width \rangle$ by $\langle height \rangle$. The $\langle name \rangle$ is what a viewer stores the signature under and what a signing tool asks for, so it is worth giving a telling one. The optional $\langle tooltip \rangle$ says what the field is for, which is also what a screen reader announces. $\{\langle width \rangle\}$ The field is placed through the kernel when `\DocumentMetadata` is declared, and $\{\langle height \rangle\}$ through the engine otherwise. A document that offers neither gets a warning and a drawn box without a field: a line to sign on paper, but nothing to sign in a viewer.

```
\article{Signatures}
\begin{paras}
  \item The client signs below.\
    \signaturefield[Signature of the client]{client}{6cm}{2cm}
\end{paras}
```

`\SignatureField` is the name this command had in xdp-sign and still answers to, so a document written for that package keeps working when it moves over.

A signature used to cost the document its PDF/A-2b conformance, and no longer does. Rule 6.3.3-1 asks every annotation that is not a `Popup`, a `Link` or of zero size to carry an appearance dictionary, and a widget is asked as well; veraPDF said of a field without one “An annotation does not contain an appearance dictionary”. Every field now points at one empty appearance, shared between all of them: the line to sign on is drawn on the page and not by the annotation, so an appearance that drew anything would draw it twice, and a signing tool replaces the normal appearance with its own the moment the field is signed. The case is in the table of section 13.3 and passes; the harness turns an unexpected pass into a failure, which is how this one announced itself.

9 External sources

A regulatory document cites instruments that are not this document: a statute, a regulation of the Union. `regulatory-sources` holds the ones a document cites and words the citations of them, so that how a citation reads is a property of the source and of the document rather than of the sentence it happens to stand in.

The module does three things. It *reads* sources, from a bib file (`\loadsources`) or from the document itself (`\newsources`), and holds them against the fields their type requires. It *stores* them, so that `\srcfield` and `\src type` give back what was declared. And it *words* a citation of one, whole or down to a provision, with `\src ref` — which is where the citation registers below come in.

Two things are deliberately elsewhere. The registers themselves live in the language definition files of section 10, since how a legal order words a citation is the same knowledge as what that language calls a paragraph. And no legislation ships with this bundle: a citation title changes on its own clock and a package on CTAN does not.

`\loadsources` Reads `<file>.bib`. The reader is this module's own, so that a document needs no second `{<file>}` program for a handful of records — and it is **not a B_BT_EX parser**. It reads a file line by line and knows four shapes, each of which is a whole line:

comment an empty line, or one whose first non-blank character is `%`. That character is this reader's, not B_BT_EX's, which has no comment character at all.

entry header `@<type>{<key>}`, and nothing after it on the line. The key holds no comma, no space and no brace.

field `<name> = {<value>}`, and nothing after it on the line. The value is between braces, and the comma at the end is optional.

closing brace `}` alone on a line, which ends the entry.

A line that is none of the four stops the run and names the file, the line number and the line itself. That is the point of the restriction: a reader that skips what it cannot read leaves a citation empty in a document that is otherwise fine, and section 13.1 has the two documents that hold it to that.

So these, all of which are ordinary B_BT_EX, are rejected here: a value in double quotes (`title = "x"`); a bare value (`year = 2024`); a whole entry on one line; a field spread over several lines; anything after the closing brace of a value on the same line; `@string` abbreviations and `#` concatenation; and `@comment` blocks. A value is stored as it stands and is never expanded, so a macro written in a file comes back as that macro.

```
@regulation{bw6,
  citetitle = {Burgerlijk Wetboek},
  abbreviation = {BW},
  book = {6},
  bwbid = {BWBRO005289},
  valid = {2024-01-01},
}
```

`\newsources` Declares one source in the document itself, which is the route that needs no file at all.
`{<key>}` It ends in the same store as a line read from a file, so nothing downstream can tell the
`{<type>}` two apart.
`{<fields>}`

`\newsourcetypealias` The types and fields are named in English, as the rest of this bundle is, and every one
`{<alias>}` of them answers to a Dutch name as well: `regeling` and `euhandel` for the types, and
`{<type>}` `citeertitel`, `afkorting`, `boek`, `geldig`, `soort`, `nummer`, `roepnaam`, `titel`, `pb` and `lidwoord` for
`\newsourcesfieldalias` the fields. A file may use either, and so may a document reading one back: the Dutch
`{<alias>}` name resolves to the English one before anything is stored. Another language is a line
`{<field>}` each.

`\newsourcetype` Declares a type, the fields an entry of it cannot do without, and the register its citations
`{<type>}` are worded in. Two are declared already: `regulation`, which needs a `citetitle`, and
`{<required fields>}` `euact`, which needs a `kind` and a `number`. A source that misses one of them stops the run,
`{<register>}` which is what lets everything downstream read a field without asking whether it is there.

`\srcfield` Read what was declared. A value is data and not $\TeX$: it comes back as it stands in the file.
`{<key>}`
`{<field>}` A citation of legislation is a citation of a *version* of it: the same article read a year apart
`\srctype` is not necessarily the same article, and the Juriconnect standard carries the date in the
`{<key>}` reference for that reason.

`\ifsourceexists` What this release does with that, exactly, so that nothing here is read as more than
`{<key>}` it is. It **records** as of when a document cites, with `\sourcedate` for the whole document
`{<true>}` or `date=<YYYY-MM-DD>` for one citation, which also answers to `datum`. It **warns** when a
`{<false>}` document says neither, and when a document speaks as of a later date than the `valid`
`\sourcedate` field of a source it cites. It does **not** print that date in a citation, and it does **not** build a
`{<YYYY-MM-DD>}` Juriconnect URI or a link to one. A `bwbid` or a `celex` field is stored and can be read back
with `\srcfield`; nothing in this bundle turns one into a reference. So: regulatory 1.0
has the date discipline a Juriconnect citation needs and not Juriconnect output.

A document that says neither is told once, as a warning and not an error: citing
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.

Every entry says when it was last held against the text it describes, in a `valid` field.
When the document speaks as of a later date than that, the entry cannot vouch for itself
and says so, once per source. An entry that leaves the field out is told so, because it
could then never be reported out of date at all: “not looked at yet” and “looked at and
current” would be the same silence, and a guard whose off switch is an absent field is
switched off by accident rather than by decision. The decision has a word of its own,
`valid = unverified`, which is silent. 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 citation title 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.

`\srcref` Cites a source, either as a whole or down to one of its provisions. The optional argument
`[<provision>]`
`{<key>}`

holds the provision — article, paragraph, point and subpoint, which answer to artikel, lid, onderdeel and onder as well — and stands before the key, as it does in \cite, so that a bracket in the sentence after a citation is not mistaken for it.

```
\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
```

The keys are the same in every line and the wording is not: what changes is the source, and with it the register it is cited in. The last two are the same provision of the same regulation, entered as avg with the Dutch register and as avgen with the English one — both stand in listing 12. The Dutch key names artikel, lid, onderdeel and onder are aliases of these and produce the same citation word for word, which section 13.2 reads back. Two things decide how that reads, and neither of them is the language of the document.

The first is the *register*, which belongs to the source: a citation is worded the way the legal order of the instrument words it. A Dutch contract that cites a regulation of the Union writes “lid 1, punt a)”, because that regulation writes it so itself — counted in the Dutch text of the General Data Protection Regulation: punt a) thirty-six times, onderdeel a) and sub a) not once. The register comes from the type of the entry and can be overridden per source with a register field.

The second is the *system*: written out in full, or shortened. The running text of a document is in full; a footnote is shortened, and so is a citation that stands between brackets in the running text. The *sourcestyle* option sets which of the two a document uses, with full or short, and the star asks for the other one. The full form is the one the Aanwijzingen voor de regelgeving prescribe for regulations themselves — a model rather than a rule here, since those instructions bind ministries and not contracts — and the shortened form is the one of the Leidraad voor juridische auteurs.

```
\newsourcedeterminer Says which article a kind of instrument takes in a register. It is not a property of the
    {\register} register: in French “de la directive” occurs sixteen times against “du règlement” seven,
    {\kind} so one default would be wrong more often than right, and “du directive” is not a matter
    {\article} of style but of grammar. The determiner field of an entry overrides it for the case no
    table can know.
```

`\newsourceregister` Declares a register: a set of keys for the full form and the same set for the shortened
`{<name>}` one. A register is not a language: the same document may cite in more than one, and it
`{<full>}` does so the moment it cites a Dutch statute next to a regulation of the Union. Five are
`{<short>}` shipped, each in the definition file of the language it belongs to and under the key it is
named by here:

`dutch` a Dutch national instrument, cited in Dutch. In full it is the form the Aanwijzingen voor de regelgeving lay down for regulations themselves; shortened it is the form of the Leidraad voor juridische auteurs. It is the register the type `regulation` takes when an entry names none. Declared in `regulatory-dutch.def`.

`dutch_eu` an act of the Union, cited in Dutch, which is a register of its own and not a variant of the first. It is the register the type `euact` takes by default. Declared in `regulatory-dutch.def`.

`german_eu` an act of the Union, cited in German. Declared in `regulatory-german.def`.

`french_eu` an act of the Union, cited in French. Declared in `regulatory-french.def`.

`english_eu` an act of the Union, cited in English, and the one of the five arranged the other way round. Declared in `regulatory-english.def`.

Those are the whole set, and the suite holds that list against the code: the registers a run declares are read back and compared with these five, so one added, renamed or lost fails a test rather than only this page. Two of the five are what a document gets without asking, since they are the defaults of the two shipped types; the other three are asked for with a register field on the entry. There is no national register for English, German or French — see the last paragraph of this section.

The three of the Union were each measured in the text of the same regulation in that language, and they differ where it matters: German writes no commas and calls the lettered point a *Buchstabe*, French writes the commas and the closing bracket but calls it a *point*. For those two the star does not shorten the wording but reaches for the abbreviation of the instrument, since that regulation never abbreviates a word of its own: `Abs.` and `par.` do not occur in it once, against `Artikel` five hundred and one.

A label never parts from its number: the regulation writes a non-breaking space between the two and a normal one between the parts. In French `paragraphe` is followed by a non-breaking space two hundred and eighty-six times and by an ordinary one five; in Dutch `artikel` two hundred and ninety-eight times against none at all.

That Dutch count is from the same regulation, and a regulation of the Union is written in the register of the Union: it says `lid 6` two hundred and eighty-five times and *zesde lid* not once. What carries over to the national register is `artikel 6`, since the label stands before its number in both. Where the national register puts the number first, *zesde lid*, there is no measurement of what binds it, and an ordinary space is what this bundle writes there.

A citation may name more than one provision. A comma makes a list and -- makes a run, and the register decides how either reads: it holds the words that join them and the plural of the designation. Both were measured in the same regulation, in each of

its languages — “artikelen 101 en 102”, “artikelen 12 tot en met 15”, “punten c) en e)”, “Buchstaben c und e”, “paragraphes 1, 2 et 4” — and the far end of a run is written apart from the near one, since the civil code is cited as “art. 6:162-164” with its book named once.

There is no national German or French register, and that is on purpose: German federal law writes § 5 Abs. 2 Nr. 3 BGB, which is another construction and not a translation of this one, and there is nobody here who could check the result. The registers themselves are not in the implementation but in the language definition file of their language, printed in section 10. That is one measurement in one place: 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*. It also spares whoever writes one a second set of names, since the register of Dutch is called *dutch* and so is the language.

10 Language support

This package began in Dutch. English came second, and the two word a reference so differently — “artikel 2, tweede lid, onderdeel b” against “Article 2(2), point (b)” — that wording one had to become a set of decisions rather than a set of words. That is what makes a third language possible, and it is also why a language file is more than a translation.

Four languages ship, and they are not equally far along. What a language file holds is three separate things, and German and French have two of the three:

	Designations	Internal references	Citation register
Dutch	full	full	dutch, dutch_eu
English	full	full	english_eu
German	full	not measured	german_eu
French	full	not measured	french_eu

Dutch and English are complete. Every designation, every reference format and both citation registers of Dutch are measured, and the wording of forty citations is read back by the suite (section 13.2).

German and French are covered in part, and the part that is missing shows. What this bundle has of them was measured in the German and French texts of the General Data Protection Regulation: the words a provision is called by, and how an act of the Union is cited in that language. What it does not have is the wording of an *internal* reference — the `\rref@setup` below — which was not measured, and which is left to whoever can check the result rather than guessed at. So:

- `\article` and `\para` carry German or French headings, and `\gls` and `\printdefs` are labelled in that language.
- `\srcref` words a citation of an act of the Union correctly, in the register of that language. A *national* German or French instrument has no register here at all; see below.
- `\rref`, `\nref` and `\aref` fall back on the English *arrangement* with German or French *words*. A German document therefore reads “Artikel 2(2), Buchstabe (b)”, which is German vocabulary in an English construction. It is visibly wrong rather than invisibly wrong, and nothing warns about it, since the words are in place.
- A reference to *another document* cannot be worded at all in either language, and that one does warn. German inflects the noun a connective governs — “der Verordnung” but “des Vertrags” — and French fuses the preposition with an article chosen by gender, so neither can compose the phrase from a subject. Both declare their connective empty, and `\documentlabel` then says so once per artifact and asks for a `ref label` (section 7). Giving one is the fix, and it is a line per document.
- There is no national German or French citation register, on purpose: German federal law writes § 5 Abs. 2 Nr. 3 BGB, which is another construction and not a translation of this one. A citation of such an instrument with the type *regulation*

is worded in the Dutch national register, since that is what that type defaults to, and that is wrong. Until such a register exists, say so with a register field rather than letting the default stand.

A language this bundle has no file for at all is a fifth case, and the mildest one: everything falls back on English, and the document is told so once at the start, with the language named. Nothing fails there — the references are simply worded in English, which is the only way it shows.

So *supported* means Dutch and English. German and French are usable for what was measured and are not the same claim, and this manual names which is which rather than counting four.

`\RegulatoryLoadLanguage` Every supported language has its own definition file `regulatory-⟨language⟩.def`, comparable to the `fc-⟨language⟩.def` files of `fmtcount`. At the end of the preamble, the definition file of every language loaded by `babel` or `polyglossia` is loaded, along with `regulatory-english.def`, which holds the defaults every other language builds upon. A language without a definition file falls back to English. Documents that select their language another way can load a definition file explicitly with `\RegulatoryLoadLanguage{⟨language⟩}`. Adding support for a new language comes down to writing such a file; the four shipped ones are printed in sections 10.1 to 10.4.

`\rref@setup` `\rref@setup` declares how `⟨lang⟩` words an internal reference, and configures its three `{⟨lang⟩}` levels at once. It is **extension API**: it is meant to be called, it is meant to be called from `{⟨article opts...⟩}` a language definition file, and it is supported as it stands for the whole of 1.x.
`{⟨para opts...⟩}` The at sign in the name is not a slip and it is not a plan to rename. A definition `{⟨point opts...⟩}` file is read with the at sign made a letter — `\RegulatoryLoadLanguage` sets its category code and restores it afterwards — which is the same convention the kernel’s own `.def` files and the `fc-⟨language⟩.def` files of `fmtcount` are written under. Inside such a file the name needs nothing. A document that declares a language in its own preamble instead wraps the call in `\makeatletter` and `\makeatother`, which is what an at sign asks of a preamble anywhere.

The other three arguments take the keys below, one set per level:

name the designation in lower case: ‘article’, ‘paragraph’, ‘point’. The default of each level is the translation of its own name,
`article \GetTranslation{article}`,
`paragraph \GetTranslation{paragraph}` and
`point \GetTranslation{point}`.

Name the same designation with an initial capital, and it defaults to the same translation with one.

names, Names the designation of a reference that names more than one structure: “articles 1 and 2”, “points (a), (b) and (d)”. It is a key of its own because it is not the language that decides whether the plural is used but the position of the designation: Dutch writes “onderdelen c en d” but keeps “het tweede en derde lid” singular, since there the designation follows the ordinals.

- `ref format` a macro of one argument, the number itself. Dutch writes a paragraph as an ordinal, for which `\ordinalstringnum` is used, and a point as a letter. A format that produces a word has to answer `\Nref` as well as `\nref`, which `\@ifrrefcap` is for: `\@ifrrefcap{↵\ordinalstringnum{...}}{\ordinalstringnum{...}}`.
- `label format` a macro of two arguments — the name and the result of `ref format` — which decides the order of the two. A Dutch article is `{#1 #2}`, a Dutch paragraph `{#2 #1}`, and an English paragraph `\@gobble{#1}#2`⁶. The value is a macro and not the text itself, so the first of those is passed as `\mylabelformat`, defined as: `\newcommand\mylabelformat[2]{#1 ↵#2}`.
- `group conjunction` what joins this level to the one above it. In “artikel.1, eerste lid” the article has a `group conjunction` of `,~.`
- `group format` a macro of one argument, which holds the levels above this one. A language that sets those apart — [Article 1(6),] point (a) — gives the point `\rref@group@braced`, which is what this bundle did for English until the European style guides settled the question the other way.

Every key can differ per level. When writing a new language, read `regulatory-english.def` first: it states every value in full, and it is the configuration a language without one of its own falls back on.

The definition files of the four languages this bundle ships follow. They are the source a translation starts from, so they are printed here rather than among the modules of section 14. Their comments, like the code, are in English.

Each of them states in its own words what it holds and what it does not, which is where the table at the head of this section comes from.

10.1 English definitions

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 section 10.2 for what a language that words its references differently needs on top of that.

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]
```

⁶`\@gobble` takes its argument and prints nothing, which is how the designation is dropped where the number stands alone.

```

\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 English citation register

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.

```

\newssourcedeterminer{english_eu}{treaty}{the}
\newssourcedeterminer{english_eu}{charter}{the}
}{}

```

10.2 Dutch definitions

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 Dutch citation registers

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
}

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

```

}}

10.3 German definitions

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 section 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 German citation register

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
}

\newsourcedeterminer{german_eu}{verordnung}{der}
\newsourcedeterminer{german_eu}{richtlinie}{der}
\newsourcedeterminer{german_eu}{empfehlung}{der}
\newsourcedeterminer{german_eu}{beschluss}{des}

```

```
\newsourcedeterminer{german_eu}{vertrag}{des}
}{}

```

10.4 French definitions

Part of a language file, for the same reason as section 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 French citation register

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
}

\newsource-determiner{french_eu}{règlement}{du}
\newsource-determiner{french_eu}{traité}{du}
\newsource-determiner{french_eu}{directive}{de~la}
\newsource-determiner{french_eu}{décision}{de~la}
\newsource-determiner{french_eu}{recommandation}{de~la}
}{}

```


11 Markdown

The regulatory-md module translates the constructs of a Markdown source into the structures of section 4. It is loaded as soon as markdown is, whatever the order of both packages, and the package option md loads markdown itself. That option is the safe way around, since markdown has to be loaded before enumitem to leave the lists of this package intact. A Markdown source is therefore limited to the constructs described here; chapters and the other components of a document class have no counterpart. Refer to listing 8 for a Markdown example, and check listing 7 to see how such a Markdown source can ultimately be used from within $\TeX$ (section 13.4).

A Markdown source contains no $\TeX$. The module used to be loaded with the hybrid option of markdown, which let every backslash through, and the examples of this bundle wrote their labels and references that way; markdown has deprecated it. Without it, a backslash is printed rather than obeyed, and a source that still holds one loses its references without a single error being raised, so a document converted from an older source is worth reading through once. Everything that used to need it is written in Markdown below.

The Markdown listings below are in Dutch in both manuals, and deliberately so: they are the constructs of example.md, which is a test fixture of this bundle and is read back by the suite (section 13.2). Nothing in this section depends on the language of the text.

11.1 Labels and references

An attribute writes a label and a link reads one. A heading and a bracketed span both take an identifier between braces, which becomes the label of the article or of the paragraph it is written in:

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

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

A reference is a link whose target starts with #, and the three shapes a Markdown-link can take are the three references of section 6:

<#label> names the structure it points at, as \Aref does, and takes several labels at once: <#lid:a, lid:b>.

[](#label) the number alone, as \rref does.

[tekst](#label) keeps the text and links it.

A label of another document is referred to in the same way, with the prefix that \refdocument gave it: <#ex2-lid:lorem>.

A bracketed span may hold a reference of the first shape, but not a second span or a link written with brackets; markdown then leaves the whole construct as it stands. For a paragraph that both carries a label and cites a definition there is the identifier of section 11.2.

11.2 An identifier of its own

A bracketed span may not hold another span or a link written with brackets, which is a limit of Markdown itself and an awkward one for a provision that both carries a label and cites a definition. This bundle therefore adds one rule to the grammar of the reader, which lets the identifier stand on its own, before the text it labels:

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

Only `{#` opens such an identifier, so a brace anywhere else in the text is left alone. The rule lives in `regulatory-syntax.lua`, which `markdown` reads itself, and which is listed with the implementation in section 14: it is Lua rather than `TEX`, since a renderer can only decide how a construct that the reader already knows is typeset, and this one has to be recognised first.

11.3 Definitions in Markdown

A definition list declares definitions. The term carries the label as a bracketed span and the body of the item becomes the description, which is the same thing `\newdefinition` does (section 5):

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

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

The other way 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, since nothing in the prose can perturb it, and the one to reach for when the definitions are maintained apart from the text:

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

Neither of the two prints anything where it stands. Where the list appears is decided by `\printdefs`, or from the source by a division, which takes the two arguments of that command as attributes and prints the list after its own content:

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

```
De volgende begrippen worden in deze overeenkomst gebruikt.
```

```
:::
```

A division with nothing in it is not recognised by `markdown`, so a line of text belongs in it. When definitions are declared and the list is never printed, the module says so at the end of the run.

A definition is cited the way anything else is referred to, with a link: `<#onderhands>` prints the name of the entry and `[een onderhandse akte](#onderhands)` keeps the wording of the sentence, both hyperlinked to the definition list.

`\autocitedefs` Marks every occurrence of the name of a definition in the text that follows, without the text having to say so. It is opt-in, and worth weighing: the matching is literal, so it is 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 what is converted after the call, so the definitions belong in a source of their own, converted first.

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

12 HTML

This package is written for PDF, as section 1 says, but a regulatory document that ends up on a website is a regular enough wish. That conversion is the work of `tex4ht`, driven by `make4ht`, which configures a package through a file `<package>.4ht`. `tex4ht` carries none for this package, so this one ships its own as `tex/regulatory.4ht`, next to the package files. A document that can find `regulatory` finds the configuration along with it. One thing a document has to do itself: *not* declare `\DocumentMetadata` in an HTML run. It activates the PDF management that section 1 asks for, which pulls in the $\LaTeX$ list code and leaves `tex4ht` nothing to turn a `paras` item into: measured on listing 5, twelve list elements with the declaration absent and none with it present. `tex4ht` defines `\HCode` before it reads the document, which is what the examples test on — see listing 2.

Both of these are said out loud rather than left to the result. A run that converts with PDF management active is told that the list markup goes away, and a run in which the configuration of this package was not found is told that an article and a paragraph become running text without a heading. Neither stops the conversion, and neither shows in an output one does not already know how to read: a heading that is no heading looks like a sentence in bold.

Whether the conversion runs at all without one turns on a single line of the preamble. The headings of this package go through `titlesec` for as long as `\ParseLaTeXeHeading` is unavailable, and `tex4ht` replaces the very sectioning machinery `titlesec` attaches to, which leaves it with nothing to hook onto. Measured with `make4ht`:

with `\DocumentMetadata` the new heading interface is available, `titlesec` is not loaded at all, and the conversion finishes without a single error.

without `titlesec` is loaded and `make4ht` stops on `Package titlesec Error: No format for this command`.

section 1 already asks for `\DocumentMetadata` for the sake of the attachments, so a document that attaches anything is on the good side of that line by the time it reaches `make4ht`. A document that attaches nothing has no reason to carry the declaration.

A conversion that finishes is not the same as a conversion that means something, and what falls away here does not announce itself. listing 5 declares `\DocumentMetadata` and converts without an error either way, which says nothing about what comes out:

	without <code>regulatory.4ht</code>	with <code>regulatory.4ht</code>
headings <code><h<n>></code>	0	5
<code><section></code>	0	5
anchors <code>id='article.<n>'</code>	4	4

The anchors are there either way, since `hyperref` writes them. It is the structure that falls away: without the configuration file an article is a paragraph with a bold run in it, and for a document whose structure *is* its meaning that is the difference between a text a reader can navigate and one that only looks right.

The file is therefore needed in both cases, for two different reasons: without `\DocumentMetadata` it keeps the conversion from failing at all, and with it, it is the only thing that produces the structure. Since it travels with the package, both are settled by loading regulatory at all.

13 Tests

The tests of this package live in the test directory and answer two different questions. `make test` typesets every example with every engine this package supports, which is `lualatex` and `pdflatex`, and needs nothing beyond a $\TeX$ installation. `make conformance` typesets the same documents once per PDF standard and has veraPDF judge the result, which needs Docker; its outcome is kept in the repository as section 13.3, since the validator is not available everywhere this package is built. That one is left out of `make test` for the same reason, and joins it with `make test WITH_CONFORMANCE=1`.

Every suite builds in a directory of its own under `test/out`: `out/lualatex` and `out/pdflatex` for the engines, `out/html` for the conversion of section 12 and `out/conformance` for the standards. They therefore do not overwrite each other, and what a suite produced stays where it is afterwards, which is what makes the HTML of a run something one can open and look at. Sharing nothing is also what lets them run at the same time, which `make test` does; each of them can be asked for on its own as `make suite-lualatex`, `make suite-html` and so on. The sources are copied into such a directory rather than found through `TEXINPUTS`: `\markdownInput` opens its file with Lua, which does not consult `kpathsea`, so that one has to be there for real.

That a document comes out is the smaller half of what a run establishes. Nearly everything this package gets wrong comes out as a document all the same — a citation worded in the wrong register, a definition that quietly did not arrive, a label that resolved to nothing. Every suite therefore ends in a set of assertions that read the run back, described in section 13.2, and two of the documents are there to fail.

13.1 The test documents

Ten documents stand in that directory. Two of them exist in both languages and one of them is converted rather than typeset, since what it is for is only read under `tex4ht`, so a suite typesets eleven files per engine and converts four. Every one of them is a driver that sets the language, declares the artifacts and loads the definitions, and that leaves the body to a file of its own, so that both language versions share one text:

`example1-nl.tex` and `example1-en.tex`, both of which read `example1.tex` and load the definitions of `example1.bib`.

`example2-nl.tex` and `example2-en.tex`, which read `example2.tex` and `example2.bib` the same way. They add a definition of their own with `\newdefinition` and print the list with the `description` style, so that both routes of section 5 that need no second program are covered.

`md-example.tex` which reads `example.md` with `\markdownInput` instead, and loads the definitions of `example1.bib`. It exists in Dutch only.

`sign-example.tex` which places the signature fields of section 8 and has no definitions at all. It is written uncompressed, so that the annotation over a field can be read back out of the PDF file without a PDF tool. It frames each field itself: the module draws a line to write on, and the frame is what shows on paper how far the area reaches that a viewer offers.

`index-example.tex` which takes the second definition route of section 5: `bib2gls=false`, so `\loadglsdefs` reads `index-defs.tex` of `\newglossaryentry` commands and `makeindex` does the sorting. It is the only document that needs a program between the runs other than `bib2gls`, and until it was written that route had no coverage at all.

`attach-example.tex` which is the only document that asks for `attachmentlink=annotation` (section 7.1), so without it that value would ship untested. It is tagged on purpose and written uncompressed, so that the annotation and its place in the structure tree can both be read out of the PDF file without a PDF tool.

`nolang-example.tex` which names no language at all and is converted rather than typeset. Every other document here loads `babel`, and a tracked language is what makes `tracklang` load a `glossaries-⟨language⟩.ldf` and with it translator. Anything in `regulatory.4ht` that reaches for a command of that package therefore works in every other document and is undefined in this one, which is how a call to `\providetranslation` survived in that file.

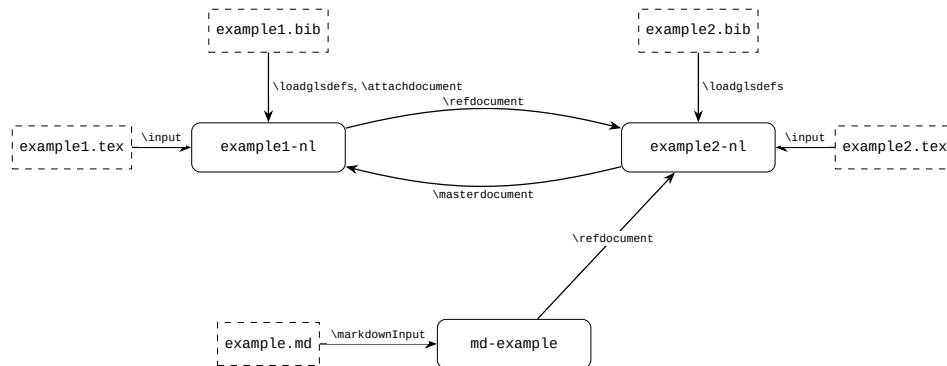
`date-example.tex` which cites without saying as of when, and then cites a source whose data was checked before the date it gives. It is the second of the two documents that are meant to warn.

`source-example.tex` which loads `sources.bib` and cites what is in it in every register of section 9. It declares two sources in the document itself as well, so that both routes into the reader are covered.

`de-example.tex` the one document that is meant to *warn*. It refers to two other documents in German, one of which says how it is to be named and one of which does not, and German cannot word that from a subject on its own. It is therefore kept out of the assertion that allows the package no warning at all, and read by an assertion of its own.

Two more are typeset that have to fail: `badsources.tex` loads a bib file with a required field missing, and `badsyntax.tex` one with a line the reader does not accept. They are not examples of anything. A reader that skips what it cannot read leaves a citation empty in a document that is otherwise fine, so the guarantee it gives is only worth what its refusal is worth, and that is what these two measure.

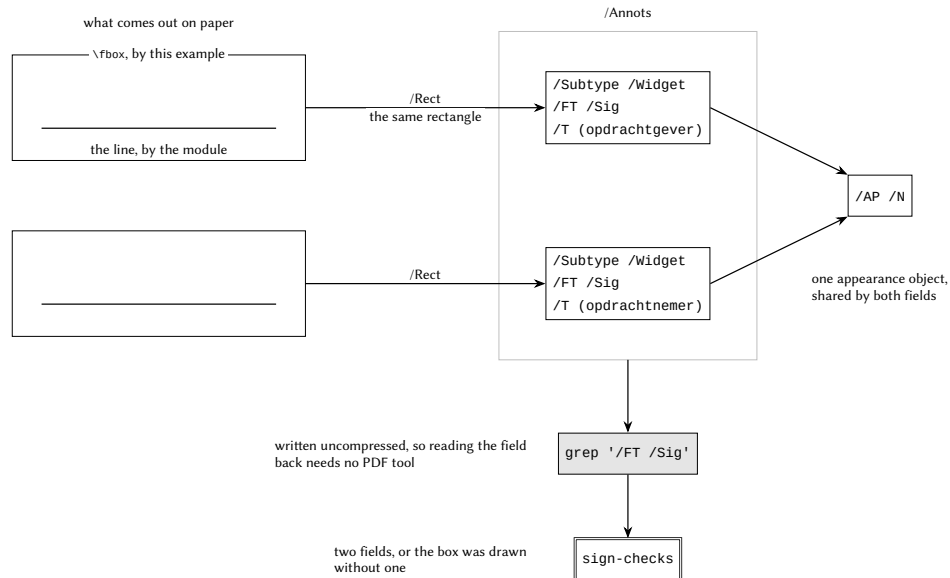
What makes the first three a test rather than three separate documents is the way they point at each other, which is what section 7 is about. `example1-nl` names `example2-nl` with `\refdocument`, so it can refer to its articles without attaching it. `example2-nl` names `example1-nl` with `\masterdocument` instead, which additionally loads its definitions and attaches the document itself. Both together therefore cover a reference in both directions, and the pair of them is declared once more by this manual, which is how the listing of section 13.4 can refer to their articles and embed their PDF files. `example1-nl` finally attaches a file that is not a PDF at all with `\attachdocument`, which is the case the PDF standards disagree about most:



A solid box is a document that becomes a PDF file, a dashed one is a file it reads. An arrow between two documents is a declaration in the preamble of the document it starts from. `md-example` loads `example1.bib` as well, which is left out of the picture for the sake of legibility.

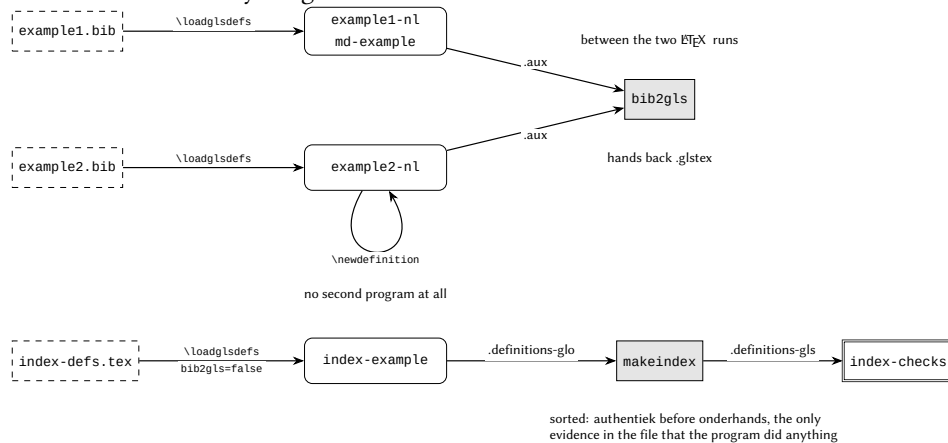
The other six are drawn as well, each on the one thing it is there for. They keep the shapes of the picture above and add three: a grey box is a program that runs between two $\text{\LaTeX}$ runs, a sharp-cornered box something written out literally — keys in a document, an object in the PDF file — and a doubled box the assertion of section 13.2 that reads the result back.

`sign-example` places two of the fields of section 8. What it draws and what it carries come apart without anything looking wrong: the frame and the line are set by $\text{\TeX}$ and are on the page whatever happens, while the annotation that makes the area a field is invisible on paper and shows only when somebody tries to sign. Both fields point at the same appearance object, which is what an empty appearance is for — the line is drawn by the page, so an appearance that drew one would draw it twice.

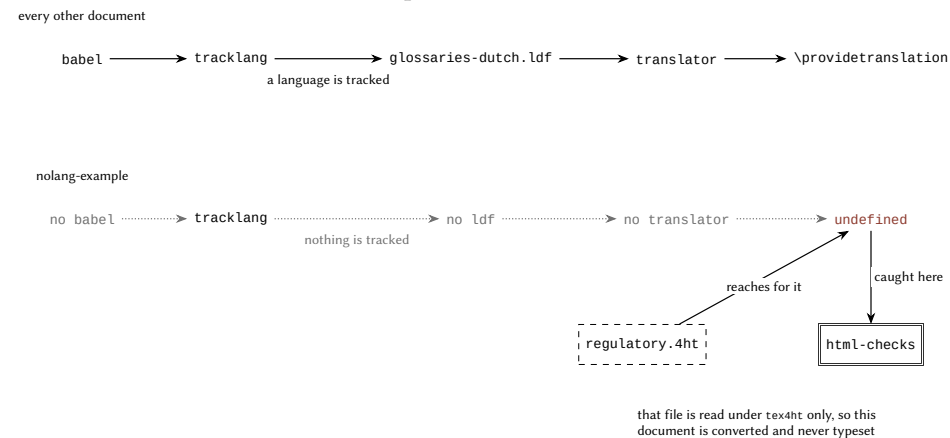


The routes of section 5 differ in what has to run between the two $\text{\LaTeX}$ runs, and that is

what separates the documents below. `index-example` is the only one on the second route and the only one that asks for a program other than `bib2gls`. Its assertion does not read the list but the order: the document declares *Onderhandse* first and *Authentieke* second, so a file that comes back the other way round is the only evidence anywhere in the run that `makeindex` did anything at all.

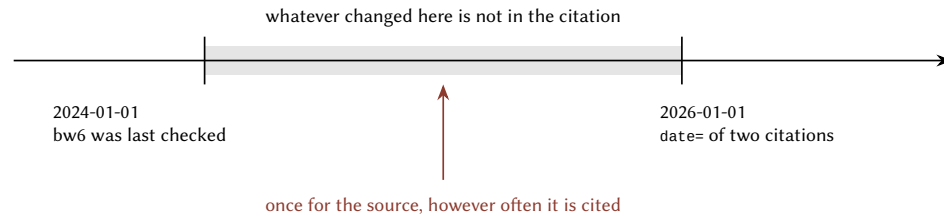


A tracked language is what makes `tracklang` load a `glossaries-⟨language⟩.ldf`, and that file is what brings `translator` along. Every other document here loads `babel`, so a command of that package works in all of them and is undefined in the one that names no language at all, which is how a call to `\providetranslation` survived in `regulatory.4ht`. The headings this document does get are English: the language files this bundle ships are loaded whatever the document speaks.



`date-example` is meant to warn four times over, and its assertion counts one of each and nothing beyond them. One is about a source that was last checked before the date this document speaks as of, and it is said once however often that source is cited; one is about a document that nowhere says as of when it cites; and two are about entries that cannot be held against a date at all. The fourth entry says as much on purpose and stays silent, which is the difference the picture is drawn for: a guard whose off switch is an absent

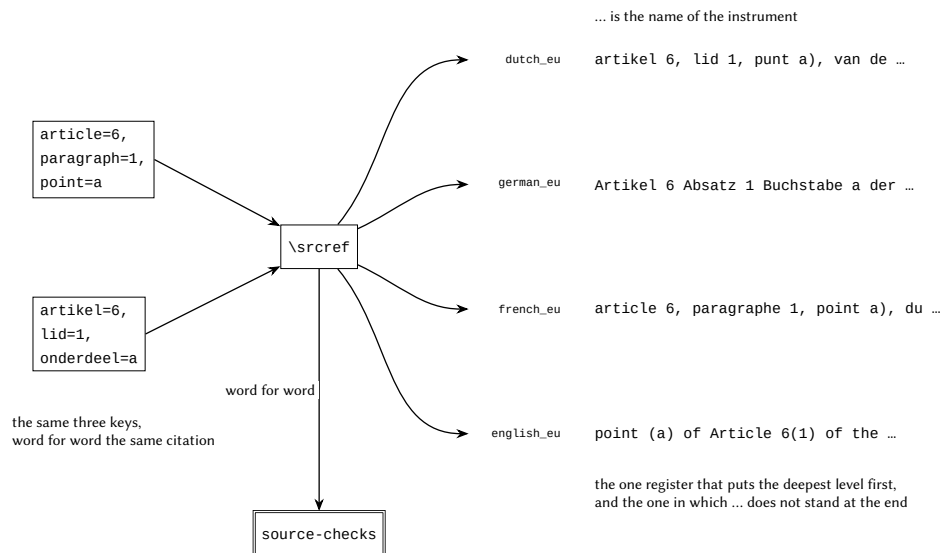
field is switched off by accident rather than by decision.



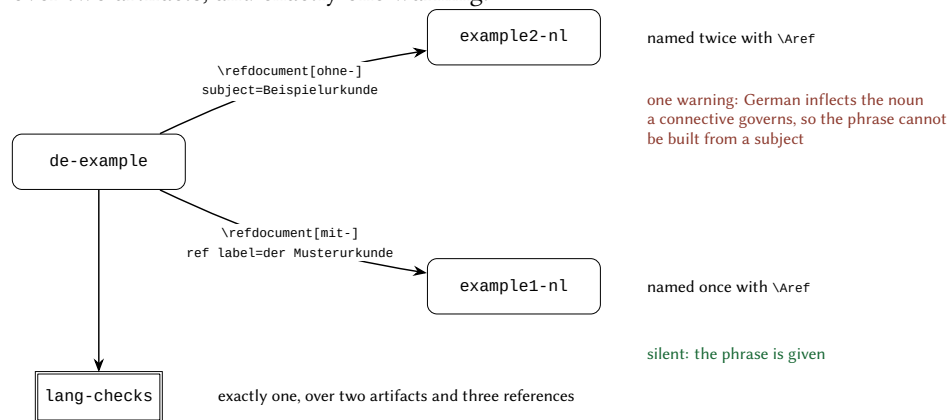
and what the document writes itself

\srcref	no date=, no \sourcedate	it says nowhere as of when it cites
zonder	no valid field	could never be reported out of date
onbekeken	valid = unverified	a decision, and decisions are silent
scheef	valid = 1 januari 2024	neither YYYY-MM-DD nor unverified
date-checks	four warnings, one of each, and nothing beyond them	

The register belongs to the source and not to the sentence it stands in, so the same three keys come out as four different citations. The English one is the only register that puts the deepest level first, and the only one in which the name of the instrument does not follow the provision. The Dutch key names are aliases of the English ones and therefore have to produce the same citation word for word; where they do not, the alias is not what it says it is. source-example is read back that way for forty citations, since one in the wrong register fails nowhere — it simply stands there wrong.



German inflects the noun a connective governs — *des Vertrags, des Beschlusses* — so no rule builds the phrase from a subject in the nominative. `de-example` names two documents, one of which carries a wording of its own and one of which does not, and refers to the one without it twice. That is what makes a single number enough: three references over two artifacts, and exactly one warning.



13.2 What the suite reads back

Every engine suite ends in ten sets of assertions, each of which prints what it counted and stops the build when it is wrong. They are written the other way round from a test that checks a result: each of them exists because something once failed without failing, and its job is to make that same silence audible next time. Every one of them has been broken on purpose to see it bite — an assertion that has never failed has not established anything yet.

`lang-checks` requires exactly one warning over the two documents `de-example` names,

and refers to one of them twice. That single number carries three claims at once: the one without a wording of its own says so, the one with a wording does not, and the one that does say so says it once however often it is named. A warning that fires per reference would drown its own log.

- `ref-checks` counts the references that resolved to nothing and the labels that were defined twice, over every document rather than only the one being worked on: a label with a space in it led nowhere while every document then in the suite used labels without one.
- `log-checks` allows the package no warning at all. What this bundle says about a document it has no reason to complain about is a defect, and a warning nobody reads is a warning that was not written.
- `attach-checks` reads the annotation of `attachmentlink=annotation` out of the PDF file: that it is placed and registered on the page, that it carries the file and an appearance, that no go-to link survived beside it, and that it sits in an `Annot` structure element as the reference beside it sits in a `Link` one. That last pair is asserted here rather than left to veraPDF because the validator this suite is pinned to does not report the missing element and an older one did: the object is the evidence and the verdict is not.
- `md-checks` reads the intermediate file the Markdown conversion writes and requires that no raw $\TeX$ survived in it, and that the syntax extension of section 11.2 was found. A missing extension leaves an identifier standing as text, which reads as a typing error and not as a broken build.
- `sign-checks` counts the signature annotations in the PDF file itself. The box under a field is drawn by $\TeX$ and shows up whatever happens; what makes it a field is invisible on paper and only shows when somebody tries to sign.
- `source-checks` reads back what the reader of bib files read, field by field, and the wording of forty citations word for word. A citation in the wrong register fails nowhere; it simply stands there wrong. The same target then typesets the two documents that have to fail and stops if either of them succeeded.
- `index-checks` reads back that the indexed route was taken, that both definitions were written to the file `makeindex` reads, and that they came back in the other order than they were declared in — which is the only evidence in the result that the program between the runs did anything. The glossary files are removed before the run: measured, with the `makeglossaries` step taken out on purpose the checks still passed, because the sorted file of the previous build was still there.
- `date-checks` reads back the two things a document can get wrong about the date a citation speaks as of: saying nothing at all, and outrunning the data of the source it cites. The second is counted rather than only found, since it

is meant to be said once per source and the example cites the same one twice.

`html-checks` counts what has to survive the conversion of section 12: sections, headings, paragraph items, definitions of both routes and the two list styles. Those disappear without an error, since a lost configuration produces plain text and not a failure. It also reads back that the document which names no language still names none, by the words it uses: that document is only worth converting while it is language-less, and its structure would survive a language being added to it.

13.3 PDF conformance

A regulatory document tends to be archived, so it is worth knowing which PDF standards it can carry. The documents above are typeset once more for every case below, with nothing changed but a `\DocumentMetadata` declared in front of the document on the commandline, and veraPDF decides. The expected column is what this package claims: `pass` has to validate, `xfail` is known to fail on a defect described below, and `fail` is a case that *has* to fail — without one, the harness cannot show that it measures anything at all.

Case	Document	Standard	expected	Result
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

Measured with veraPDF 1.30.2, with the validator pinned by digest⁷.

⁷`verapdf/cli@sha256:d5ee329657cf9bc4b2400392dd54c7d0a0ce9980ff6fa2da5590eebeec007cdb`

One claim a document written with this package cannot make. **PDF/A-1** forbids embedded files outright, so any document that attaches something is out of reach.

PDF/A-2a and **PDF/A-4** stood here as a second, and they do not belong there. Tagging hangs a pair of HTML files on the catalog as associated files, and both standards require every embedded file to be a PDF/A itself — but those two files are the whole obstacle, and `tagpdf` has a key that leaves them out: `\tagpdfsetup{attach-css=false}`. The table carries both halves as two pairs of cases that differ in nothing else, on a document that attaches nothing of its own: without the key PDF/A-2a fails on rule 6.8-5 and PDF/A-4 on rule 6.9-3, with it both pass, and **PDF/UA-1** passes either way. What is given up is what those two files are for — styling hints for lists and for the alignment of MathML — and not the structure a reader is read by.

The difference between the profiles is measured the same way: the same document with the same attachment appears twice in the table, failing PDF/A-2b on rule 6.8-5 and passing PDF/A-4f. Anything that is not a PDF therefore decides the profile, and it is worth knowing which one before a document is issued rather than after.

One case is `xfail`. It fails ISO 19005-3 rule 6.8-4 on a file specification for the external document: it carries `/AFRelationship /Unspecified`, has no `/EF` and is not listed in `/AF`. It is written for the hyperlink to that document, by the PDF management of $\LaTeX$ and not by this package, so PDF/A-3 stays out of reach for a document that refers to the provisions of another one until that is settled upstream. Section 7 says so where a reader meets the feature.

The second document reached PDF/A-3a when the citation of a definition that lives elsewhere stopped being a hyperlink to that document. That link went to an anchor which is not in this file, so it led nowhere in any case; withdrawing it took the file specification with it. Measured on the two documents as they stand: `example1-nl` carries eleven `/GoToR` actions and the one specification without an `/EF`; `example2-nl` carries no `/GoToR` at all, and its one specification for the document it attaches has an `/EF` and `/AFRelationship /Source`.

13.4 The test files

The sources of the examples the suite builds, listed here and attached to this manual as PDF files. They are one set of fixtures and both manuals print the same one, so the ones with a text of their own are in Dutch either way; `example1-en.tex` and `example2-en.tex` are the English drivers over the same bodies.

```
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}.
```

Listing 5: `example1.tex`

The result of this example is `example1-en.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

```

Listing 6: example2.tex

The result of this example is example2-en.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}

```

Listing 7: md-example.tex

The result of this example 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.
```

Listing 8: example.md

The result of this example 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 }

```

Listing 9: example1.bib

The result of this example is example1-en.pdf.

```

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

```

Listing 10: example2.bib

The result of this example is example2-en.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}

```

Listing 11: sign-example.tex

The result of this example 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 \newsourc{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 \newsourc{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 \newsourc{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 \newsourc{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 \newsourc{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 \newsourc{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}

```

Listing 12: source-example.tex

The result of this example 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 }

```

Listing 13: sources.bib

The result of this example is source-example.pdf.

14 Implementation

The seven modules of the bundle are documented below, in the order they are loaded: the six that regulatory loads, and regulatory-md, which joins them as soon as markdown is loaded. The Markdown syntax extension and the tex4ht configuration follow, which are not modules but are generated from sources of their own all the same. Every module is written as a .dtx file, from which both this chapter and the module itself are generated. The commentary, like the code, is in English.

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 Package options

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 section 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
\ifregulatory@bibtogls switches 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 ,
branstijl .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 `\srcrf` 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 Prerequisites

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 Colors

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 Headings

`\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 Lists

`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 Reference properties

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 Language files

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 What the run cannot fix

`\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 Postponed 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 Prerequisites

`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 The definitions glossary

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

14.2.3 List styles

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 <code>alttree</code> style glossaries brings: one built on the <code>labeling</code> environment of <code>scrxextend</code> , which aligns every description on the widest name given to <code>\printdefs</code> , and one on the <code>description</code> environment of the class, which leaves the alignment to the class. Both put the anchor on the name, so a <code>\gls</code> 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 *<width of text>*. 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*\glstylefont[1]{\textmd{##1}}%
  \setboolean{regulatory@defslisted}{true}%
  \stepcounter{regulatory@defsprinted}%
  \protected@edef\regulatory@thiswidest{#2}%
  \expandafter\glstylefont\expandafter{\regulatory@thiswidest}%
  \edef\regulatory@thisstyle{%
    \ifcsname @glstyle@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{%
    \begin{group}%
    \setglossarystyle{\regulatory@thisstyle}%
    \ifthenelse{\boolean{regulatory@indexeddefs}}{%
      \printglossary[type=definitions,title={},nonumberlist=true]%
    }{%
      \printunsrtglossary[type=definitions,title={},nonumberlist=true]%
    }%
    \end{group}%
  }%
}
```

`\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 Prerequisites

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 Switches

`\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 Reference formats

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@refformat@noop[1]{#1}
\newcommand\rref@refformat@parenthesis[1]{(#1)}
\newcommand\rref@refformat@ordinal[1]{\ifrrrefcap{\Ordinalstringnum{#1}}{\ordinalstringnum{#1}}}
\newcommand\rref@refformat@alpha[1]{\@ifrrrefcap{\ABAlphnum{#1}}{\abalphnum{#1}}}
\newcommand\rref@refformat@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 Language configuration

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@reformat@parenthesis,
        label format=\rref@label@refonly,
        group conjunction={,},
        group format=\rref@reformat@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@reformat@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@reformat@` 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 Reference macros

`\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 The current reference

`\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 Complete references

\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 Enumerations

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 Prerequisites

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 Artifacts

`\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 Embedding files

\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_pdffile_mimetypes_prop \l__regulatory_ext_str
  {
    \pdfdict_put:nne {\l_pdffile}{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_pdffile/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_pdffile/Filespec}{Desc}{\l__regulatory_desc_str}
  }
  \pdffile_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:Nc \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 Mentioning the source

`\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}}{true}}{\documentfootnote{\@aref@filelabel}}{}%
    }%
  }%
}

```

```

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

```

14.4.5 Linking documents

`\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}}{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 `\zexternaldocument` 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 Prerequisites

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 Conversion options

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 section 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 Structures

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 section 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 References

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

```

14.5.5 Definitions

`\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 Definitions in the 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 Citing a definition

`\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 Prerequisites

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 The annotation

\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 Signature fields

`\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 Prerequisites

```

\RequirePackage{regulatory-struct}

```

14.7.2 Sources of a source

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 }
      }
    }
  }
  {
    \msg_warning:nnx { regulatory-sources } { no-valid }
    { \l__regulatory_src_key_str }
  }
}

\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:nnnnn { 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 Where a register lives

Not here. A register is written in the language definition file of its language, `regulatory-⟨language⟩.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. section 10 has the loading rule.

14.7.5 Naming more than one

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 Citing a source

`\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 }{%
  \begingroup
    \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}%
    \fi
  }
}

```

```

\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 syntax extension

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 section 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 (section 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{>}%
\b:regarticle\GetTranslation{Article}\ \thearticle.\c:regarticle
#2%
\d: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
  \a:regarticle\HCode{>}%
  \b:regarticle\c:regarticle
  #2%
  \d: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}%
  \a:regpara\reg@anchorid{\@currentHref}\HCode{>}%
  \b:regpara\thearticle.\thepara.\c:regpara
  #2%
  \d:regpara
  \reg@inparatrue
  \par\ShowPar
}

\newcommand\reg@para@star[2][]{%
  \reg@closepara
  \reg@blockmode

```

```

\ a:regpara\HCode{>}%
\ b:regpara\c:regpara
#2%
\ d: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 nowhere 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.

Index

Every command and environment of this bundle, with the page it is described on in *italics* and the page it is defined on underlined. Which of the three kinds of name a command is — public, extension or internal — is settled in section 2.9.

Symbols		I	<code>\newsourcetypealias</code> . . . 24
<code>\@ifnameinlink</code> 81	<code>\ifregulatory@alldefs</code> . 66	<code>\Nref</code> 16, 86	
<code>\@ifrrefcap</code> 81	<code>\ifregulatory@bibtogls</code> . 66	<code>\nref</code> 16, 86	
	<code>\ifregulatory@legacy</code> . . 66		
A	<code>\ifregulatory@markdown</code> . 66	O	
<code>alldefs (option)</code> 4	<code>\ifregulatory</code>	options:	
<code>\Aref</code> 16, 89	<code>@nameinlink</code> 66	<code>alldefs</code> 4	
<code>\aref</code> 16, 89	<code>\ifsourceexists</code> . . . 24, 121	<code>attachmentlink</code> 5	
<code>\aref@postlink@setup</code> 88, 100		<code>bib2gls</code> 4	
<code>\aref@setcurrent</code> 88	L	<code>defstyle</code> 5	
<code>\article</code> 12, 70	<code>legacy (option)</code> 5	<code>legacy</code> 5	
<code>\attachdocument</code> 20, 99	<code>\loadextdefs</code> 15, 101	<code>md</code> 4	
<code>attachmentlink (option)</code> . . 5	<code>\loadglsdefs</code> 15, 79	<code>nameinlink</code> 5	
<code>\autocitedefs</code> 43, 109	<code>\loadsources</code> 23, 119	<code>sourcestyle</code> 5	
B	M	P	
<code>\begincompaction</code> 90	<code>\markdownRendererDl</code>	<code>\para</code> 12, 70	
<code>bib2gls (option)</code> 4	<code>DefinitionBegin</code> . . . 107	<code>paras (env.)</code> 12, 71	
<code>\bubblesort</code> 91	<code>\markdownRendererDlItem</code> 104	<code>\printdefs</code> 15, 78	
	<code>\markdownRendererFenced</code>	<code>\ProvidesRegulatory</code>	
D	<code>DivAttributeContext</code>	<code>Language</code> 73	
<code>definitions (env.)</code> 14	<code>Begin</code> 107	<code>provisions (env.)</code> . . . 12, 71	
<code>defstyle (option)</code> 5	<code>\markdownRenderer</code>		
<code>\describe</code> 15, 78	<code>HeadingOne</code> 103	R	
<code>\documentattachment</code> . 20, 98	<code>\markdownRendererLink</code> 104	<code>\refdocument</code> 18, 101	
<code>\documentfootnote</code> . 20, 100	<code>\markdownRendererOl</code>	<code>\regulatory-description</code> 77	
<code>\documentlabel</code> 20, 99	<code>Begin</code> 103	<code>\regulatory-labeling</code> . . 77	
	<code>\markdownRendererOlItem</code>	<code>\regulatory@attach</code>	
E	<code>WithNumber</code> 103	<code>@annot</code> 96	
environments:	<code>\markdownRenderer</code>	<code>\regulatory</code>	
<code>definitions</code> 14	<code>RegulatoryLabel</code> . . . 105	<code>@attachmentlink</code> 97	
<code>externals</code> 14	<code>\markdownRendererTilde</code> 105	<code>\regulatory</code>	
<code>paras</code> 12, 71	<code>\masterdocument</code> . . . 18, 102	<code>@checkenvironment</code> . . 75	
<code>provisions</code> 12, 71	<code>md (option)</code> 4	<code>\regulatory@defstyle</code> . . 68	
<code>externals (env.)</code> 14		<code>\regulatory@embed</code> 95	
	N	<code>\regulatory@embedonce</code> . 98	
F	<code>nameinlink (option)</code> 5	<code>\regulatory@extdefs</code>	
<code>\findendlist</code> 90	<code>\newartifact</code> 92	<code>@nohyper</code> 101	
	<code>\newdefinition</code> 15, 79	<code>\regulatory@ifembed</code> . . . 98	
G	<code>\newsourcetype</code> 24, 116	<code>\regulatory@ifmodule</code> . . 69	
<code>\GetTranslation</code> 69	<code>\newsourcedeterminer</code> . . 25	<code>\regulatory@load</code>	
	<code>\newsourcetypealias</code> . . 24	<code>@languages</code> 74	
H	<code>\newsourceregister</code> 26	<code>\regulatory@md@dlitem</code> 106	
<code>\hashchar</code> 80	<code>\newsourcetype</code> 24, 114		

<code>\regulatory@md</code>	<code>\Rref</code>	16, <u>86</u>	<code>\setlastconjunction</code> .	17, <u>81</u>
<code>@dlitemend</code>	<code>\rref</code>	16, <u>86</u>	<code>\setmiddleconjunction</code>	
<code>\regulatory@md@link</code> ..	<code>\rref@get</code>	<u>85</u>		17, <u>81</u>
<code>\regulatory@md@nodefs</code> <u>110</u>	<code>\rref@init@lang@article</code> <u>83</u>		<code>\setrangeconjunction</code> 17, <u>81</u>	
<code>\regulatory@md@yaml</code> ..	<code>\rref@init@lang@para</code> ..	<u>83</u>	<code>\SignatureField</code>	<u>113</u>
<code>\regulatory@sign@annot</code> <u>110</u>	<code>\rref@init@lang@sub</code> . . .	<u>83</u>	<code>\signaturefield</code>	22, <u>113</u>
<code>\regulatory@sign</code>	<code>\rref@labellist</code>	<u>89</u>	<code>\sourcedate</code>	24, <u>128</u>
<code>@appearance</code>	<code>\rref@lang</code>	<u>84</u>	<code>sourcestyle (option)</code>	5
<code>\regulatory@sign@place</code> <u>111</u>	<code>\rref@number</code>	<u>88</u>	<code>\srcfield</code>	24, <u>120</u>
<code>\regulatory@sourcestyle</code> <u>68</u>	<code>\rref@set</code>	<u>85</u>	<code>\srcref</code>	24, <u>127</u>
<code>\regulatory@src</code>	<code>\rref@setcurrent</code>	<u>87</u>	<code>\srcregister</code>	<u>120</u>
<code>@messages</code>	<code>\rref@setup</code>	29, <u>84</u>	<code>\srctype</code>	24, <u>120</u>
<code>\RegulatoryLoadLanguage</code>	S			
.	<code>\setconjunction</code>	17, <u>81</u>	U	
<code>\regulatorysetup</code>	<code>\setjuncto</code>	17, <u>82</u>	<code>\unsetjuncto</code>	17, <u>82</u>