LaTeX Package contract*

Markus Kohm

Version v0.91 of 2024-02-07

The contract package is intended for cautelar jurisprudence. It is intended to provide flexible help for lawyers and notaries in drafting contracts, statutes and legal commentaries. It has been developed in cooperation with Dr Alexander Willand, and is still in the process of development.

Package contract is a KOMA-Script spin-off. It has been released from 2011 till 2023 as scrjura. With KOMA-Script 3.42 scrjura is removed from KOMA-Script and released as new package contract.

Contents

---

[*] The repository of this package can be found at `https://github.com/komascript/latex-contract` where you also should report issues.

List of Tables

List of Figures

User Manual

If you want to write a contract, the articles of association for a company or an association, a law, or a legal commentary, the package contract will provide typographical support.

Although contract is intended to provide general help for legal documents, the contract is the central element of the package. Particular attention is paid to clauses, titles, and numbered provisions — if there are several of them in a clause —, numbered sentences, entries in the table of contents, and cross references according to German standards.

The package has been developed in cooperation with Dr Alexander Willand of Karlsruhe. Many of its features go back to constructive inquiries from Prof Heiner Richter of the Hochschule Stralsund University of Applied Sciences.

Some of you may search for the German user manual formally available in [Koh23a]. The author's acute overload has resulted in this no longer being freely available. However, the corresponding chapter in [Koh20a] and [Koh20b] is still accessible and applicable with appropriate adjustments to the package name.

1 Package Loading and Option Setting

You can load the package as common using:

```
\usepackage[⟨options⟩]{contract}   .
```

In case of package contract the ⟨*options*⟩ are ⟨*key*⟩=⟨*value*⟩ options.

Note: In case of using package contract with package hyperref you should load hyperref always after contract.

`\contractSetup`

To change options after loading the package, you should use:

```
\contractSetup{⟨options⟩}
```

with ⟨*options*⟩ is a comma separated list of ⟨*key*⟩=⟨*value*⟩ options as explained before.

Note: Currently the options are still implemented using the internal KOMA-Script package scrkbase. Therefore you could also setup ⟨*options*⟩ using the KOMA-Script commands `\KOMAoptions` or `\KOMAoption`, which are described in [Koh23b]. Moreover, because of the implementation, currently `\contractSetup` could also be used to setup other KOMA-Script options not related to package contract. However, you should avoid this, because it will fail, when the implementation will be changed.

2 Changing the Fonts of Elements

Currently package contract still uses the font selection features of the internal KOMA-Script package scrkbase. So the commands `\setkomafont`, `\addtokomafont`, and `\usekomafont`, which are described in [Koh23b], can be used to change the fonts of the following elements:

`Clause`:

Alias for ⟨*environment*⟩`.Clause` within any contract environment, e. g., `contract.Clause` within environment `contract`. If no corresponding element is defined, `contract.Clause` is used.

`contract.Clause`: (Default: `\sffamily\bfseries\large`)

The heading of a `\Clause` within the environment `contract`.

`⟨environment⟩.Clause`: (Default: *none*)
The heading of a `\Clause` within environment ⟨*environment*⟩, which has been defined using `\DeclareNewJuraEnvironment`, if the font has been setup using property `ClauseFont` or the element has been defined explicitly.

`parnumber`: (Default: *empty*)
The paragraph number within a contract environment.

`sentencenumber`: (Default: *empty*)
The sentence number printed by `\Sentence`.

Note: There are plans to no longer use package scrkbase, but maybe package scrextend instead, if scrjura is used with a non-KOMA-Script class. So the commands `\setkomafont`, `\addtokomafont` and `\usekomafont` would still be available in future.

3 Clauses in the Table of Contents

The headings of clauses can also be added automatically to the table of contents, if desired. For this the package uses command `\DeclareTOCStyleEntry` of KOMA-Script package tocbasic to define an entry level named `cpar`. Usage of package tocbasic also means, that you should not use another package to configure the Table of Contents, e.g., tocloft, tocstyle etc.

`juratotoc` *(opt.)* Clauses are shown in the table of contents only if their level number is less than or equal to the `tocdepth` counter. By default, the level number is `\maxdimen`, which is also used if the option is switched off using `juratotoc=false`. Because the `tocdepth` counter usually has a one-digit value, clause entries are therefore not normally displayed in the table of contents.

If you switch on the option using the `juratotoc=true`, the level number 2 is used so that clauses are shown in the table of contents on the same level as subsections. For the default setting of `tocdepth`, clauses are then shown in all KOMA-Script classes or standard classes.

You can also use `juratotoc=⟨integer⟩` to use ⟨*integer*⟩ instead of `\maxdimen` or 2 as level number.

Internally usage of this option results in a call of

```
\DeclareTOCStyleEntry[level=⟨integer⟩]{default}{cpar}
```

respectively

```
\DeclareTOCStyleEntry[level=2]{default}{cpar}
```

respectively

```
\DeclareTOCStyleEntry[level=\maxdimen]{default}{cpar}   .
```

`juratocindent` *(opt.)*
`juratocnumberwidth` *(opt.)*
These options determine the indentation and spacing for clause entries in the table of contents. Any valid ⟨*length*⟩ can be assigned. The defaults are the same as for subsection entries in scrartcl.

Internally, usage of these options results in calls to:

```
\DeclareTOCStyleEntry[indent=⟨length⟩]{default}{cpar}
```

respective

```
\DeclareTOCStyleEntry[numwidth=⟨length⟩]{default}{cpar}   .
```

4 Environment for Contracts

The essential mechanisms of package contract are available only inside contract environments, either the predefined `contract` or any other environment defined with command `\DeclareNewJuraEnvironment`.

`contract` Currently, this is the one and only predefined environment for contract. Using it activates automatic numbering of paragraphs and the `\Clause` and `\SubClause` commands, which will be documented below, are given concrete form.

The `contract` environment must not be nested within itself. Within a document, however, you can use the environment several times. The clauses within these environments are treated as if they were within a single environment. As a result, ending the environment really only temporarily interrupts it, and the old environment is continued by the beginning of a new environment. However, you cannot end the environment within a clause. If you want instead print several contracts, you would need to define several contract environments using command `\DeclareNewJuraEnvironment`.

`contract` *(opt.)* The whole document becomes a contract if you use this option while loading the package with:

```
\usepackage[contract]{contract}
```

or as a global option with `\documentclass`. The document then behaves exactly as if it would contain one `contract` environment.

4.1 Clauses

Clauses[1] in a legal sense are defined in package contract only within contracts, that is inside the `contract` environment or other environments declared with `\DeclareNewJuraEnvironment`.

`\Clause`
`\SubClause`
Each clause starts either with `\Clause[`⟨*property list*⟩`]` or `\SubClause[`⟨*property list*⟩`]`. The optional argument ⟨*property list*⟩ is a comma separated list of ⟨*key*⟩`=`⟨*value*⟩.

These are the most important commands inside of a contract. Without using any additional ⟨*key*⟩, `\Clause` creates the heading of a clause, which consists of the sign “§”, followed by its number. In contrast, `\SubClause` creates the heading of a clause with the last number used by `\Clause` and adds a lower-case letter. `\SubClause` is mainly intended for cases where an act or a contract is amended and not only are clauses changed or deleted but new clauses are inserted between existing ones without completely changing the numbering.

Both commands accept a comma-separated list of ⟨*key*⟩`=`⟨*value*⟩ properties and also some ⟨*key*⟩s without value. An overview of the available properties is shown in Table 1. The most important of them will be discussed in more detail.

By default, a skip of two lines is inserted before the heading and a skip of one line afterwards. You can change the size of these skips with the `preskip` and `postskip` properties. The new values apply not only to the current clause but from the current clause until the end of the current contract environment. You can also make the appropriate settings in advance with

```
\setkeys{contract}{preskip=⟨skip⟩,postskip=⟨skip⟩}
```

---

[1] In English, a “clause” in a legal document is a section, paragraph, or phrase that relates to a particular point. Although it is common in English to also use the terms “article” or “section” for what we here call a “clause”, we use the latter term throughout to avoid confusion with the article class and the `\section` and `\paragraph` sectioning divisions of most document classes.

Table 1: Available properties for the optional argument of `\Clause` and `\SubClause`

`dummy`	The heading will not be printed but is counted in the automatic numbering.
`head=`⟨*running head*⟩	If running heads are active, this ⟨*running head*⟩ is used instead of the clause ⟨*title*⟩.
`nohead`	The running head stays unchanged.
`notocentry`	Does not make an entry into the table of contents.
`number=`⟨*number*⟩	Uses ⟨*number*⟩ for the output of the clause number.
`preskip=`⟨*skip*⟩	Changes the vertical ⟨*skip*⟩ before the clause heading.
`postskip=`⟨*skip*⟩	Changes the vertical ⟨*skip*⟩ after the clause heading.
`title=`⟨*title*⟩	The clause ⟨*title*⟩ will be printed in addition to the clause number. This is also used as the default for the ⟨*running head*⟩ and the ⟨*entry*⟩ in the table of contents.
`tocentry=`⟨*entry*⟩	Regardless of the clause ⟨*title*⟩, an ⟨*entry*⟩ into the table of contents will be made, if such entries are activated (see option `juratotoc`).

regardless of the specific clause and outside of a contract environment. You can also set these options inside the preamble after loading `contract`, but you cannot set them while loading the package or by using `\contractSetup`.

By default, clause headings use the font style `\sffamily\bfseries\large`. See section 2 for information about how to change the font for element `contract.Clause`.

With the `title`, `head`, and `tocentry` property, you can title a clause in addition to the number. You should enclose the ⟨*value*⟩ of each property inside curly brackets. Otherwise, for example, commas which are meant to be part of the ⟨*value*⟩ will be confused with the delimiters between different properties of the ⟨*property list*⟩. Empty values for `head` and `tocentry` cause empty entries. If you want to avoid an entry, use the `nohead` resp. `notocentry` property.

Instead of consecutive numbers, you can also set a clause number manually with the `number` property. However, this does not affect the numbers of the subsequent clauses. Empty numbers are not possible. Fragile commands inside `number` have to be protected with `\protect`. You should use only numbers and letters as a `number`.

With the `dummy` property, you can suppress the output of the whole heading of a clause. The automatic numbering, however, will still count this clause. In this way, you can skip an automatically numbered clause with

```
\Clause[dummy]
```

in case the clause corresponding clause has been deleted in a later version of a contract.

Note that the `dummy` property only accepts the ⟨*value*⟩s `true` and `false`. All other ⟨*value*⟩s are usually ignored, but can lead to an error message in the worst case scenario.

`\Clauseformat` As already mentioned, clauses and subclauses are normally numbered. The number is formatted with the help of the `\Clauseformat` command, which expects the ⟨*number*⟩ as the only argument. The default is the following:

```
\newcommand*{\Clauseformat}[1]{\S #1}
```

Table 2: Available values for the `clausemark` option to activate running heads

`both`	Clauses generate left and right marks for running heads, if the document provides automatic running heads.
`false`, `off`, `no`	Clauses do not generate marks for running heads and therefore do not change running heads.
`forceboth`	Clauses use `\markboth` to generate left and right marks for running heads even if the document does not provide automatic running heads for the current page style.
`forceright`	Clauses use `\markright` to generate right marks for running heads even if the document does not provide automatic running heads for the current page style.
`right`	Clauses generate right marks for running heads, if the document provides automatic running heads.

This produces the section mark, `\S` (§), followed by a non-breaking space and the number. If you redefine this command, be sure it remains expandable.

`juratitlepagebreak` *(opt.)*

Usually, page breaks are prohibited within heading of all kinds. However, some lawyers require page breaks within clause headings. You can allow such a break by using option `juratitlepagebreak` or `juratitlepagebreak=`⟨*boolean*⟩. Boolean values of `true` or `false` can be used to toggle the option on or off. Using the option without a value is the same as using `true`. The option can be used as an optional argument of `\documentclass`, `\usepackage` when loading package contract, or as an argument of `\contractSetup`.

`clausemark` *(opt.)*

Since clauses are a subordinate structure with independent numbering, they do not produce running heads by default. You can, however, create running heads with various settings using:

`clausemark=`⟨*value*⟩

as an optional argument of `\documentclass`, `\usepackage` when loading package contract, or as an argument of `\contractSetup`. You can find the available ⟨*value*⟩s and their meanings in Table 2.

4.2 Paragraphs

Within clauses, contract usually numbers paragraphs automatically. With this, the paragraphs provide a powerful structuring element, similar to `\paragraph` or `\subparagraph` in normal documents. For this reason, contracts usually use a vertical skip between paragraphs. The contract package does not provide its own mechanism for this. Instead, you should uses the `parskip` option of the KOMA-Script classes. If you do not use a KOMA-Script class, see the documentation of the used class or package parskip [Mit21].

`parnumber` *(opt.)*

The option can be used as an optional argument of `\documentclass`, `\usepackage` when loading package contract, or as an argument of `\contractSetup`.

The default numbering of paragraphs is `parnumber=auto` and `parnumber=true`. Sometimes you may need to disable the automatic numbering. You can do this with `parnumber=false`. In this case, only the sentence numbering is reset.

To implement this option, it has been necessary to hook into the paragraph-building mechanism of LaTeX. In some rare cases, this can have a negative effect. If so, you can undo the change with `parnumber=manual`. On the other hand, LaTeX itself sometimes undoes the change. In those cases you can activate it again with `parnumber=auto`.

Clauses that consist of a single paragraph do not automatically receive a paragraph number. For this to work, there must not be two clauses with an identical number in a document. However should you ever need such numbering, you should switch to another contract environment (see `\DeclareNewJuraEnvironment`). Note that the number of paragraphs in a clause is not available before the end of the clause. Therefore you need a least two LaTeX runs before the automatic paragraph numbering is correct.

`par` (*cnt.*)
`\thepar`
`\parformat`
`\parformatseparation`

For numbering the paragraphs inside a clause we use the `par` counter. The output of `\thepar` will display an Arabic number, because the default is `\arabic{par}`. `\parformat` provides the format, which is `\thepar` in rounded brackets. When numbering a paragraph manually, you should also use `\parformat`. It makes sense to call `\parformat` with a subsequent `\parformatseparation`, or at least a `\nobreakspace` or tilde.

With automatic numbering, `\parformat` is followed by `\parformatseparation`, which currently consists of `\nobreakspace`, the non-breakable space.

The paragraph number is usually printed using the currently active font. See section 2 for information how to change the font of the `parnumber` element.

Note: contract currently assumes internally that `\thepar` is an Arabic number. Therefore you should definitely not redefine it!

`\withoutparnumber`

If the paragraph number is not printed, contract executes the `\withoutparnumber` command at the beginning of the new paragraph. The initial definition of this command is empty. This means it is a kind of dummy command that does nothing. It has been implemented because of a user request. Most users can ignore this command.

`\ellipsispar`
`\parellipsis`

Sometimes — particularly in comparative commentaries — it is desirable to omit paragraphs but to mark the omission. Those omitted paragraphs should be taken into account by the paragraph counter. The package contract provides the command `\ellipsispar` to do this.

By default, `\ellipsispar` omits precisely one paragraph. Using the optional argument of

```
\ellipsispar[⟨number⟩]
```

you can omit multiple paragraphs. In any case, the output shows just one unnumbered paragraph, which consists only of the ellipsis defined by `\parellipsis`. The automatic numbering of paragraphs takes the ⟨*number*⟩ of omitted paragraphs into account.

Example: Suppose you are writing a comment on the German[2] penal code, but only on paragraph 3 of § 2. Nevertheless, you'd like to indicate the omission indirectly. You can do this with:

```
\documentclass[parskip=half]{scrartcl}
\usepackage{contract}
\begin{document}
\begin{contract}
  \Clause{title={Temporal application},number=2}
```

---

[2] Please remember, this translation does not refer to an existing law but is only an example of how you might realise such a commentary with contract.

```
    \ellipsispar[2]

    If the law that applies at the time the criminal act is
    committed is changed before the verdict, then the most
    lenient law shall be applicable.

    \ellipsispar[3]
  \end{contract}
  \end{document}
```

To see the result, just give it a try.

The ellipsis is by default `\textellipsis`, if such a command is defined. If not, `\dots` is used. You can redefine `\parellipsis` at any time with `\renewcommand`.

4.3 Sentences

Paragraphs in contracts consist of one or more sentences, some of which may also be numbered. However, as automatic numbering is cumbersome and error-prone, it has not yet been implemented in contract. Semi-automatic numbering, however, is supported.

`sentence` (*cnt.*)
`\thesentence`
`\sentencenumberformat`
`\Sentence`

Manual numbering of sentences is done with the `\Sentence` command. It adds one to the `sentence` counter. By default, `\sentencenumberformat` prints `\thesentence` as an Arabic number in superscript.

The sentence number is usually printed using the currently active font. See section 2 for information how to change the font of the `sentencenumber` element.

Using babel offers an easy way to define a shorthand for `\Sentence`:

```
\useshorthands{'}
\defineshorthand{'S}{\Sentence\ignorespaces}
```

With this definition, any space after `'S` will be ignored. You can even use the dot as an abbreviation for a dot and a new sentence number:

```
\defineshorthand{'.}{. \Sentence\ignorespaces}
```

For details regarding `\useshorthands` and `\defineshorthand`, please consult the manual of the babel package (see [BB24]). You can find an example of their application in section 8, page 14.

5 Cross References

The conventional mechanism to set cross references using `\label`, `\ref`, and `\pageref` does not suffice for clauses, paragraphs, and sentences. Therefore contract provides additional commands.

`\ref`
`\refL`
`\refS`
`\refN`

The commands

```
\ref{⟨label⟩}
\refL{⟨label⟩}
\refS{⟨label⟩}
\refN{⟨label⟩}
```

give a full reference to clause, paragraph and sentence. `\refL` is a long text, `\refS` a short text, and `\refN` an abbreviated, numeric form. `\ref` defaults to `\refL`.

`\refClause`
`\refClauseN`

The commands

```
\refClause{⟨label⟩}
\refClauseN{⟨label⟩}
```

reference a clause without displaying the paragraph or sentences. `\refClause` puts a section mark (§) in front of the reference, while `\refClauseN` does not.

`\refPar` `\refParL` `\refParS` `\refParN` The commands

```
\refPar{⟨label⟩}
\refParL{⟨label⟩}
\refParS{⟨label⟩}
\refParN[⟨number format⟩]{⟨label⟩}
```

reference a paragraph of a clause. The differences between the forms correspond to the differences between `\refL`, `\refN` and `\refS`. A feature worth noting is the optional argument of `\refParN`. Usually the numeric reference to a paragraph uses a Roman number. You can, however, specify a different ⟨*number format*⟩ in the optional argument. This option primarily makes sense to use Arabic numbers. By default, `\refPar` is `\refParL`.

`\refSentence` `\refSentenceL` `\refSentenceS` `\refSentenceN` The commands

```
\refSentence{⟨label⟩}
\refSentenceL{⟨label⟩}
\refSentenceS{⟨label⟩}
\refSentenceN{⟨label⟩}
```

reference a sentence of a paragraph. Again, there is a long text form, a short text form, and a numerical form. By default, `\refSentence` is `\refSentenceL`.

`ref` *(opt.)* The results of `\ref`, `\refPar`, and `\refSentence` depend on the option

```
ref=⟨value⟩
```

that can be used as an optional argument of `\documentclass`, `\usepackage` when loading package contract, or as an argument of `\contractSetup`. The default is `ref=long` and therefore `\refL`, `\refParL` and `\refSentenceL`. You can find the available ⟨*value*⟩s for his option and their meaning in Table 3.

Example: Suppose you always want to reference paragraphs in the form “paragraph 1 in clause 1”. As there is no predefined command for this, you have to create your own definition from the available options. You can achieve this easily with:

```
\newcommand*{\refParM}[1]{%
  paragraph~\refParN[arabic]{#1}
  in clause~\refClauseN{#1}%
}
```

This new command can be used in the same way as `\refParL`.

You can find examples of results of the basic commands — this means the commands, that are independent from option `ref` — in Table 4.

Table 3: Available values for the `ref` option to configure the cross reference format of `\ref`, `\refPar`, and `\refSentence`

`long`	A combination of `parlong` and `sentencelong`.
`numeric`	A combination of `parnumeric` and `sentencenumeric`.
`clauseonly`, `onlyclause`, `ClauseOnly`, `OnlyClause`	A combination of `paroff` and `sentenceoff`. Note that `\refPar` and `\refSentence` produce empty results!
`parlong`, `longpar`, `ParL`	Paragraphs are referenced in long textual form.
`parnumeric`, `numericpar`, `ParN`	Paragraphs are referenced in simple numerical form.
`paroff`, `nopar`	Paragraphs have no reference. Note that `\refPar` produces an empty result!
`parshort`, `shortpar`, `ParS`	Paragraphs are referenced in short textual form.
`sentencelong`, `longsentence`, `SentenceL`	Sentences are referenced in long textual form.
`sentencenumeric`, `numericsentence`, `SentenceN`	Sentences are referenced in simple numeric form.
`sentenceoff`, `nosentence`	Sentences have no reference. Note that `\refSentence` produces an empty result!
`sentenceshort`, `shortsentence`, `SentenceS`	Sentences are referenced in short textual form.
`short`	A combination of `parshort` and `sentenceshort`.

Table 4: Example outputs of the `ref`-independent cross reference commands

Command	Example output
`\refL{`⟨*label*⟩`}`	§ 1 paragraph 1 sentence 1
`\refS{`⟨*label*⟩`}`	§ 1 par. 1 sent. 1
`\refN{`⟨*label*⟩`}`	§ 1 I 1.
`\refClause{`⟨*label*⟩`}`	§ 1
`\refClauseN{`⟨*label*⟩`}`	1
`\refParL{`⟨*label*⟩`}`	paragraph 1
`\refParS{`⟨*label*⟩`}`	par. 1
`\refParN{`⟨*label*⟩`}`	I
`\refParN[arabic]{`⟨*label*⟩`}`	1
`\refParN[roman]{`⟨*label*⟩`}`	i
`\refSentenceL{`⟨*label*⟩`}`	sentence 1
`\refSentenceS{`⟨*label*⟩`}`	sent. 1
`\refSentenceN{`⟨*label*⟩`}`	1.

6 Additional Contract Environments

Some users do not use contract to draft contracts or commentaries on individual laws but to examine different types of laws, which may not necessarily use the section prefix (§) before the title of each clause but perhaps something like “Art.” or “IAS”, and so forth. An independent counter is also required for each of these different clause types.

`\DeclareNewJuraEnvironment`

You can use:

```
\DeclareNewJuraEnvironment{⟨environment⟩}[⟨property list⟩]
     {⟨start commands⟩}{⟨end commands⟩}
```

to define new and independent environments for contracts or other legal texts. The argument ⟨*environment*⟩ is the name of the new environment, of course. The ⟨*start commands*⟩ are commands which will be executed at the beginning of the environment, as if they were added directly after `\begin{`⟨*environment*⟩`}`. Correspondingly ⟨*end commands*⟩ will be executed at the end of the environment, as if added directly before `\end{`⟨*environment*⟩`}`. Without a ⟨*property list*⟩ the new environment behaves like the `contract` environment, but with its own counters. You can use several ⟨*key*⟩=⟨*value*⟩ properties as a comma-separated ⟨*property list*⟩. See Table 5 for the currently supported ⟨*options*⟩.

Example: To define the environment for articles we mentioned in the preface of this section, it is sufficient to write:

```
\DeclareNewJuraEnvironment{Article}[ClauseNumberFormat=Art.~]{}{}
```

If we are using a KOMA-Script class and want to separate the paragraphs in this environment with space instead of using paragraph indentation, we can use:

```
\DeclareNewJuraEnvironment{Article}[ClauseNumberFormat=Art.~]
                           {\KOMAoptions{parskip}}{}
```

In cross references, “Art.” will of course be used instead of “§”.

The new environment is used like `contract`:

```
\begin{Article}
  \Clause{}
  Human dignity is inviolable. To respect and protect people is a
  duty of all state authority.
\end{Article}
```

7 Support for Different Languages

The contract package has been developed in cooperation with a German lawyer. Therefore it initially supported only the languages `german`, `ngerman`, `austrian`, and `naustrian`. Nevertheless, it has been designed to support common language packages like babel. Users can easily make changes by using `\providecaptionname`. If you have definitive information about the correct legal terms and conventions of a language, please contact the KOMA-Script author. Support for English has been added in this way, and so contract now also provides terms for the languages `english`, `american`, `british`, `canadian`, `USenglish`, and `UKenglish`.

`\parname`
`\partshortname`
`\sentencename`
`\sentenceshortname`

These are the language-dependent terms used by contract. The meaning of the terms and their English defaults are shown in Table 6. The package itself defines them by using

Table 5: Properties provided by `\DeclareNewJuraEnvironment` for new contract environments

`Clause=`⟨*command*⟩	Defines the ⟨*command*⟩ to which the `\Clause` command is mapped within the environment. This ⟨*command*⟩, like the one documented for `contract`, expects exactly one argument. To use it correctly requires advanced knowledge of the contract's internal functioning. Furthermore, the requirements for the ⟨*command*⟩ may change in future versions. Therefore it is recommended not to use this option!
`ClauseFont=`⟨*commands*⟩	If this property is used, a new ⟨*environment*⟩`.Clause` element is defined with the ⟨*commands*⟩ used as its default setting. If the element was previously defined as an alias, it will become an independent element instead. If it has already been defined as an independent element, the ⟨*commands*⟩ are used as new font settings. Please note the limitations for font settings in section 2.
`SubClause=`⟨*command*⟩	Defines the ⟨*command*⟩ to which the `\SubClause` command is mapped within the environment. This ⟨*command*⟩, like the one documented for `contract`, expects exactly one argument. To use it correctly requires advanced knowledge of the contract's internal functioning. Furthermore, the requirements for the ⟨*command*⟩ may change in future versions. Therefore it is recommended not to use this property!
`Sentence=`⟨*command*⟩	Defines the ⟨*command*⟩ to which the `\Sentence` is mapped within the environment. This ⟨*command*⟩ must not have an argument. Typically it should add one to the `sentence` (using `\refstepcounter`) counter and display it appropriately. It is particularly important to avoid adding unwanted spaces.
`ClauseNumberFormat=`⟨*command*⟩	Formats the numbers of clauses within the environment. The ⟨*command*⟩ should expect exactly one argument: the number of the clause. If the ⟨*command*⟩ implements a series of commands and the number is the last argument of a that series, you can directly use the series of commands as ⟨*command*⟩.

Table 6: Meanings and English defaults of language-dependent terms, if not already defined

Command	Meaning	Default
`\parname`	long form “paragraph”	paragraph
`\parshortname`	short form “paragraph”	par.
`\sentencename`	long form “sentence”	sentence
`\sentenceshortname`	short form “sentence”	sent.

`\providecaptionname` inside `\begin{document}` only if other requirements have not already been met. If you use contract with an unsupported language, the package will throw an error.

8 A Detailed Example

You may remember the letter from the scrlttr2 chapter of the KOMA-Script manual [Koh23b], in which a club member wanted to remind the board about an overdue meeting that was prescribed by the club’s by-laws. Such club by-laws are a kind of contract, and you can create them using contract.

```
\documentclass[fontsize=12pt,parskip=half]
              {scrartcl}
```

We use class scrartcl. Because paragraph distance instead of paragraph indentation is usual in club by-laws, we load the class with option `parskip=half`.

```
\usepackage[british]{babel}
```

The club rules are in British English. Therefore we load the babel package with the `british` option too.

```
\usepackage[T1]{fontenc}
\usepackage{lmodern}
```

We make some default font settings. Earlier versions of the example also loaded the textcomp package here for an improved section mark (§). Since LaTeX 2020/02/01, however, the desired functionality is directly integrated in the LaTeX kernel.

```
\usepackage{enumerate}
```

Later in the document, we want lists numbered not with Arabic numbers but with lower-case letters. We can do this easily with the enumerate package. Alternatively, we could have used the enumitem package.

```
\usepackage[clausemark=forceboth,
             juratotoc,
             juratocnumberwidth=2.5em]
            {contract}
\useshorthands{'}
\defineshorthand{'S}{\Sentence\ignorespaces}
\defineshorthand{'.}{. \Sentence\ignorespaces}

\pagestyle{myheadings}
```

Now it is time for contract. The `clausemark=forceboth` option forces clauses to create left and right marks for the running head. On the other hand, we do not want `\section` to change the marks for the running head. Therefore we use the `myheadings` page style. This page style generally does not provide automatic running heads.

Later, we also want a table of contents with the clauses. This can be achieved with the `juratotoc` option. Doing so we will see that the default width for these numbers is insufficient for the clause numbers in the table of contents. With `juratocnumberwidth=2.5em`, we reserve more space.

The definition of shorthands has already been explained in section 4.3. In this example we do the same thing to simplify the input.

```
\begin{document}
```

It is time to begin the actual document.

```
\subject{By-Laws}
\title{CfCH}
\subtitle{Club for Club Hoppers}
\date{11.\,11.\,2011}
\maketitle
```

Like other documents, the by-laws have a title. We created it with the usual KOMA-Script commands.

```
\tableofcontents
```

As already mentioned, we want to create a table of contents.

```
\addsec{Preamble}

The club landscape in England is diverse. But we have
unfortunately been forced to conclude that it often
suffers seriously when dealing with seriousness.
```

Preambles are not unusual in club by-laws. Here we use `\addsec` to create one because we want it to have an entry in the table of contents.

```
\appendix
```

Here we use a small trick. The articles of the club by-laws should be numbered with uppercase letters instead of Arabic numbers, just as the appendix sections of an article using scrartcl are.

```
\section{Overview}

\begin{contract}
```

We begin the contract with the first article.

```
\Clause[title={Name, Legal Form, Headquarters}]

The name of this club shall be the ``Club for Club
Hoppers'' and is not registered in any club register.

'S The club is a non-economic, useless club'. It has no
headquarters because its members heads are in their
hindquarters.

The fiscal year is from March 31st through April 1st.
```

The first clause has a number and a title. We will do the same with all following clauses.

The first paragraph of the clause contains nothing unusual. Because it is not the only paragraph, every paragraph will be automatically preceded by a paragraph number. Note that the numbering the first paragraph requires at least two LaTeX runs. Since this is the case for the table of contents as well, this does not create any additional problems.

In the second paragraph we have two sentences. Here we can see the shorthands `'S` and `'.` in action. The first one only generates the sentence number. The second one generates not only the full stop but also the sentence number. With this, both sentences are numbered.

```
\Clause[title={Purpose of the Club}]

'S The club is pointless but not useless'. Rather,
it should put the serious handling of seriousness on a
sound footing.

For this purpose, the club members can
\begin{enumerate}[\qquad a)]
\item pick their noses,
\item crack nuts,
\item such their thumbs.
\end{enumerate}

The club is selfish and stands by it.

The club has no financial means.\label{a:mittel}
```

The second clause: again this contains several paragraphs, some of which include several sentences. The second paragraph also has a numbered list. In the last paragraph, we set a label, because we want to reference it later.

```
\Clause[title={Club Officers}]

The club officers hold honorary positions.

'S If the club had resources (see \ref{a:mittel}), it
could afford a full-time manager'. Without the necessary
funds, this is not possible.
```

The third clause contains something special: a cross reference. Here we use the long form with clause, paragraph, and sentence. If we decided later that sentences should not be included in the reference, we could use the `ref=nosentence` option to set this globally.

```
\Clause[title={Club Hopper},dummy]
\label{p.maier}
```

Here we have a special kind of clause. In earlier versions of the club by-laws, this was a real clause, but it was later removed. However, the numbering of the following clauses should not be changed by removing this one. Therefore the `\Clause` statement has not been removed but supplemented by `dummy` property. With this, we also can set a label even though the clause will not be printed.

```
\end{contract}

\section{Membership}

\begin{contract}
```

Another article begins. To avoid problems with the paragraph numbering, we interrupt the `contract` environment.

```
\Clause[title={Types of Members},dummy]
```

The first clause of the next article also has been deleted.

```
\Clause[title={Becoming a Member}]

Everyone can purchase a membership from one of the
associations listed in \refClause{p.maier}.\label{a.preis}

'S To become a member, an informal application is
required'. This application should be submitted in green
ink on pink paper.

Membership applications cannot be rejected.
```

Here we have a real clause again. We cross reference one of the deleted clauses and also set a label.

```
\SubClause[title={Amendment to the Previous Clause}]

'S With the repeal of \refClause{p.maier},
\ref{a.preis} has become impractical'. In its place,
memberships can be inherited.
```

Once more, this is a special kind of clause. This time we have not removed a clause but added one without renumbering the following clauses. To do so, we use `\SubClause`. Therefore the clause number is the same like the previous one but with an appended “a”.

```
\Clause[title={Termination of Membership}]

'S Membership ends with one's life'. For non-living
members, membership does not end.

\Clause[title={General Meeting}]

A general meeting shall take place twice per year.

The interval between two general meetings shall be
no more than 6~months, 1~week, and 2~days.

The invitation to the next general meeting shall be sent
no earlier than 6~months from the previous general
meeting.

\SubClause[title={Amendment to the General Meeting}]

The general meeting may be held at the earliest 2~weeks after
the invitation is received.
\end{contract}
```

The other clauses of this article are very usual. You already know all the features used for them.

```
\section{Validity}
```

```
\begin{contract}
\Clause[title={Effective Date}]

These articles will enter into force on 11.\,11.\,2011 at
11:11~am.

'S If any provision of these by-laws is in conflict with
any other, the by-laws will be repealed on
11.\,11.\,2011 at 11:11~am and 11~seconds'. The club is
considered to be dissolved in this case.

\end{contract}
```

There follows another article no special features.

```
\end{document}
```

Then the LaTeX document ends. You can see first three pages in Figure 1.

9 From scrjura to contract

If you have been using the scrjura package and are now switching to the contract package, there are a few things you need to be aware of:

- Instead of

  `\usepackage[⟨options⟩]{scrjura}` ,

  should now use

  `\usepackage[⟨options⟩]{contract}` .

  The same ⟨*options*⟩ are allowed as the scrjura package understands.

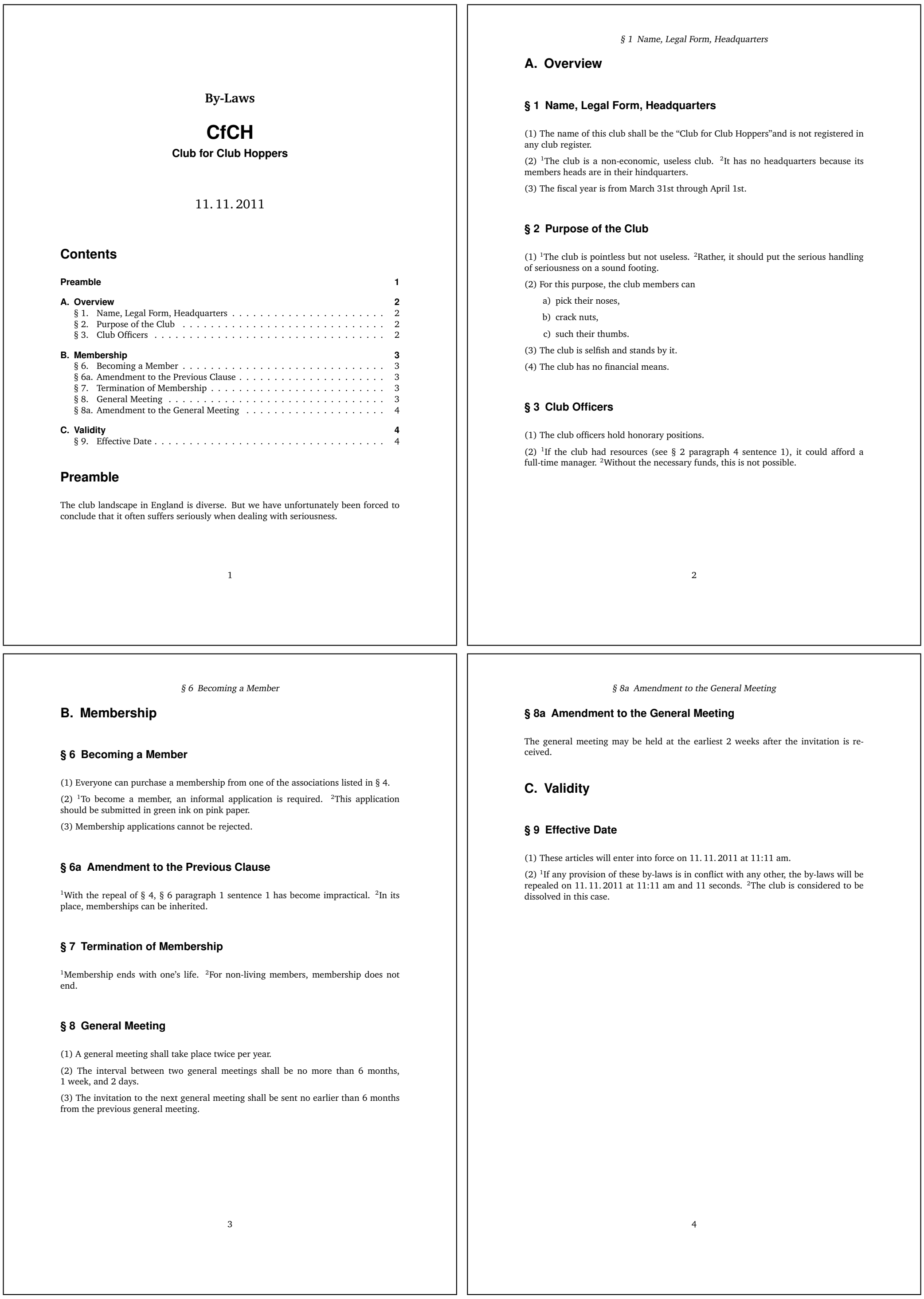
- Instead of using `\KOMAoptions` or `\KOMAoption` to set ⟨*options*⟩ for the contract package, use `\contractSetup`.

- The `\Clause` and `\SubClause` commands now have an optional argument instead of a mandatory argument. This makes more sense because it is allowed to use `\Clause` or `\SubClause` without an argument. Instead of braces, the list of properties must be given in square brackets, for example:

  ```
  \Clause[title={title of clause}]
  ```

  Therefore, even more care should be taken to ensure that the values of each property are enclosed in braces. Otherwise, there may be problems not only with commas, but also with square brackets in the values.

- Check your code if you have used or redefined any internal macros as several of them have been renamed.



By-Laws

CfCH

Club for Club Hoppers

11. 11. 2011

Contents

Preamble

The club landscape in England is diverse. But we have unfortunately been forced to conclude that it often suffers seriously when dealing with seriousness.

A. Overview

§ 1 Name, Legal Form, Headquarters

(1) The name of this club shall be the "Club for Club Hoppers"and is not registered in any club register.

(2) [1]The club is a non-economic, useless club. [2]It has no headquarters because its members heads are in their hindquarters.

(3) The fiscal year is from March 31st through April 1st.

§ 2 Purpose of the Club

(1) [1]The club is pointless but not useless. [2]Rather, it should put the serious handling of seriousness on a sound footing.

(2) For this purpose, the club members can

a) pick their noses,

b) crack nuts,

c) such their thumbs.

(3) The club is selfish and stands by it.

(4) The club has no financial means.

§ 3 Club Officers

(1) The club officers hold honorary positions.

(2) [1]If the club had resources (see § 2 paragraph 4 sentence 1), it could afford a full-time manager. [2]Without the necessary funds, this is not possible.

B. Membership

§ 6 Becoming a Member

(1) Everyone can purchase a membership from one of the associations listed in § 4.

(2) [1]To become a member, an informal application is required. [2]This application should be submitted in green ink on pink paper.

(3) Membership applications cannot be rejected.

§ 6a Amendment to the Previous Clause

[1]With the repeal of § 4, § 6 paragraph 1 sentence 1 has become impractical. [2]In its place, memberships can be inherited.

§ 7 Termination of Membership

[1]Membership ends with one's life. [2]For non-living members, membership does not end.

§ 8 General Meeting

(1) A general meeting shall take place twice per year.

(2) The interval between two general meetings shall be no more than 6 months, 1 week, and 2 days.

(3) The invitation to the next general meeting shall be sent no earlier than 6 months from the previous general meeting.

§ 8a Amendment to the General Meeting

The general meeting may be held at the earliest 2 weeks after the invitation is received.

C. Validity

§ 9 Effective Date

(1) These articles will enter into force on 11. 11. 2011 at 11:11 am.

(2) [1]If any provision of these by-laws is in conflict with any other, the by-laws will be repealed on 11. 11. 2011 at 11:11 am and 11 seconds. [2]The club is considered to be dissolved in this case.

Figure 1: The four pages of the CfCH example of section 8

10 State of Development

Since KOMA-Script 3.24, the scrjura package has shared the version number of the classes and other important packages of KOMA-Script. Package contract on the other hand now has a new version number independent from KOMA-Script.

Independent from the version number you should note that so far, the interaction of the `contract` environment with the many different settings possible with other LaTeX environments, packages, or classes has not been tested. The main reason for this is that contract is very specialized and far beyond the author's ordinary practice. So the author mostly relies on detailed user feedback.

Implementation

```
⟨*package⟩
```

11 Cooperation with hyperref

If hyperref has already loaded before contract the package cannot work correctly. So we throw an error. Maybe it would be a good idea to make this error fatal. But currently it is only an error.

```
⟨*init⟩
\@ifpackageloaded{hyperref}{%
  \PackageError{contract}{Package hyperref already loaded}{%
    If you want to use package contract with package hyperref, you have
    to\MessageBreak
    load package contract before package hyperref.\MessageBreak
    To solve the problem, you just should move the loading of package
    hyperref\MessageBreak
    behind the loading of package contract.}%
}
⟨/init⟩
```

12 Prerequisites

We need package scrkbase. We could load this also together with tocbasic, which is loaded later. But loading it on its own, we can require a minimum version.

Todo: Package contract is no longer a KOMA-Script package. So it should not use the internal KOMA-Script package scrkbase. For keys of contract environments we can just load scrbase. For package options we should either also switch to scrbase or use native key-value options of LaTeX. After doing so we would be able to replace scrkbase at least by scrextend. With this package we still could use commands like `\setkomafont`. Theoretically we would also be able to still use KOMA-Script options, but this would still not be recommended.

```
⟨*init⟩
\RequirePackage{scrkbase}[2013/03/26]
```

```
⟨/init⟩
```

And now, tocbasic.

```
⟨*init⟩
\RequirePackage{tocbasic}
⟨/init⟩
```

13 Options

`contract`
`\if@documentcontract`
`\@documentcontractfalse`
`\@documentcontracttrue`

Option `contract` can be used to make the whole document to be a contract. But in this case you are not allowed to reuse the `contract` environment in the document. Nor is it allowed to stop or restart the contract. With older LaTeX this is done by adding `\contract` to the end of `\document`. With an up to date LaTeX we us a hook.

```
⟨*options⟩
\KOMA@ifkey{contract}{@documentcontract}
\IfLTXAtLeastTF{2020/10/01}{%
  \AddToHook{begindocument/end}{%
    \RelaxFamilyKey[.contract.sty]{KOMA}{contract}%
    \if@documentcontract\expandafter\contract\fi
  }%
}{%
  \g@addto@macro\document{%
    \RelaxFamilyKey[.contract.sty]{KOMA}{contract}%
    \if@documentcontract\expandafter\contract\fi
  }%
}
⟨/options⟩
```

`juratotoc`

Allow to set the toc level of the entries. Value `true` is the same like 2, value `false` is the same like `\maxdimen`.

`\if@juratotoc`
`juratoclevel` (*cnt.*)
`\toclevel@cpar`

```
⟨*options⟩
\KOMA@key{juratotoc}[true]{%
  \KOMA@set@ifkey{juratotoc}{@tempswa}{#1}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed
    \if@tempswa
      \DeclareTOCStyleEntry[level=2]{default}{cpar}%
    \else
      \DeclareTOCStyleEntry[level=\maxdimen]{default}{cpar}%
    \fi
  \else
    \DeclareTOCStyleEntry[level=#1]{default}{cpar}%
  \fi
  \KOMA@kav@xreplacevalue{contract.sty}{juratotoc}{\cpartocdepth}%
}
\KOMA@kav@xadd{contract.sty}{juratotoc}{\cpartocdepth}%
⟨/options⟩
```

`juratocnumberwidth`
`juratocindent`
`\cpar@numberwidth` (*ilen.*)
`\cpar@indent` (*ilen.*)

Indent and number width of the toc entries.

Todo: Since we are using package tocbasic for the ToC entries these options (and lengths) are not needed any longer, but users should use `\DeclareTOCStyleEntry` to setup the number width and indent of the entries.

```
⟨*options⟩
\KOMA@key{juratocnumberwidth}{%
  \DeclareTOCStyleEntry[numwidth=#1]{default}{cpar}%
  \FamilyKeyStateProcessed
  \KOMA@kav@replacevalue{contract.sty}{juratocnumberwidth}{#1}%
}
\KOMA@kav@add{contract.sty}{juratocnumberwidth}{2em}
\KOMA@key{juratocindent}{%
  \DeclareTOCStyleEntry[indent=#1]{default}{cpar}%
  \FamilyKeyStateProcessed
  \KOMA@kav@replacevalue{contract.sty}{juratocindent}{#1}%
}
\KOMA@kav@add{contract.sty}{juratocindent}{1.5em}%
⟨/options⟩
```

`juratitlepagebreak` The options sets the boolean `\if@juratitlepagebreak`.

`\if@juratitlepagebreak`
`\@juratitlepagebreaktrue`
`\@juratitlepagebreakfalse`

If the boolean is `\iftrue` page breaks inside clause headings are allowed (which is not recommended).

Todo: Re-implementation either using a native key-value option (easy) or a new scrbase family.

```
⟨*options⟩
\KOMA@ifkey{juratitlepagebreak}{@juratitlepagebreak}
⟨/options⟩
```

`parnumber` The options switches the (automatic) paragraph numbering.

Todo: Re-implementation either using a native key-value option (easy) or a new scrbase family.

```
⟨*options⟩
\newif\ifparnumber
\KOMA@key{parnumber}[true]{%
  \Ifstr{#1}{auto}{%
    \AutoPar
    \FamilyKeyStateProcessed
    \KOMA@kav@remove{contract.sty}{parnumber}{manual}%
    \KOMA@kav@remove{contract.sty}{parnumber}{auto}%
    \KOMA@kav@add{contract.sty}{parnumber}{auto}%
  }{%
    \Ifstr{#1}{manual}{%
      \ManualPar
      \FamilyKeyStateProcessed
      \KOMA@kav@remove{contract.sty}{parnumber}{manual}%
      \KOMA@kav@remove{contract.sty}{parnumber}{auto}%
      \KOMA@kav@add{contract.sty}{parnumber}{manual}%
    }{%
      \KOMA@set@ifkey{parnumber}{parnumber}{#1}%
      \KOMA@kav@replacebool{contract.sty}{parnumber}{parnumber}%
    }%
  }%
}
\KOMA@kav@add{contract.sty}{parnumber}{true}
\KOMA@kav@add{contract.sty}{parnumber}{auto}
⟨/options⟩
```

`paragraphmark`
`clausemark`

Todo: Re-implementation either using a native key-value option or a new scrbase family (easy).

`markright`
`markboth`
`\Clausemark`

Todo: Remove deprecated options, because they need an internal KOMA-Script macro.

The options are used to activate either `\markright` or `\markboth` for clauses. `\Clausemark` expects not only the title but also the number. So it differs from, e.g., `\chaptermark`, which uses the counter automatically. But maybe I will change this some time.

```
⟨*options⟩
\newcommand*{\Clausemark}[1]{}
\KOMA@key{clausemark}{%
  \begingroup
    \KOMA@set@ncmdkey{clausemark}{@tempa}{%
      {false}{0},{off}{0},{no}{0},%
      {forceright}{1},%
      {forceboth}{2},%
      {right}{3},%
      {both}{4}%
    }{#1}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed
    \ifcase\number\@tempa
      \endgroup
      \let\Clausemark\@gobble
    \or
      \endgroup
      \renewcommand*{\Clausemark}[1]{%
        \markright{\csname MakeMarkcase\endcsname{##1}}}%
    \or
      \endgroup
      \renewcommand*{\Clausemark}[1]{%
        \markboth{\csname MakeMarkcase\endcsname{##1}}%
                 {\csname MakeMarkcase\endcsname{##1}}}%
    \or
      \endgroup
      \renewcommand*{\Clausemark}[1]{%
        \ifx
          \@mkboth\@gobbletwo
        \else
          \markright{\csname MakeMarkcase\endcsname{##1}}%
        \fi}%
    \or
      \endgroup
      \renewcommand*{\Clausemark}[1]{%
        \@mkboth{\csname MakeMarkcase\endcsname{##1}}%
```

```
                {\csname MakeMarkcase\endcsname{##1}}}%
    \else
      \endgroup
    \fi
    \FamilyKeyStateProcessed
  \else
    \endgroup
    \FamilyKeyStateUnknownValue
  \fi
  \KOMA@kav@xreplacevalue{contract.sty}{clausemark}{#1}%
}
\KOMA@kav@add{contract.sty}{clausemark}{false}
\@ifundefined{KOMA@DeclareDeprecatedOption}{}{%
  \KOMA@DeclareDeprecatedOption[contract]{markright}{clausemark=forceright}%
  \KOMA@DeclareDeprecatedOption[contract]{markboth}{clausemark=forceboth}%
}
\KOMA@key{paragraphmark}{%
  \PackageWarningNoLine{contract}{%
    You've used obsolete option `paragraphmark'.\MessageBreak
    Usage of this option is deprecated.\MessageBreak
    You should simply replace `paragraphmark'\MessageBreak
    by `clausemark'%
  }%
  \KOMAExecuteOptions[.contract.sty]{clausemark=#1}%
}
⟨/options⟩
```

`ref` **Todo:** Re-implementation either using a native key-value option or a new scrbase family (easy).

`parcitename`
`sentencecitename` The formatting of the references of paragraphs and sentences. There are a long a short and a numeric form.

Todo: Remove deprecated options, because they need an internal KOMA-Script macro.

`\parcite@format`
`\sentencecite@format` Default is the long form. Corresponding values of the two helper macros are: 0 = long, 1 = short, 2 = numerical, -1 = nothing.

```
⟨*options⟩
\newcommand*{\parcite@format}{0}
\newcommand*{\sentencecite@format}{0}
```

The options can be used to change the default.

```
\KOMA@key{ref}{%
  \begingroup
    \KOMA@set@ncmdkey{ref}{@tempa}{%
      {parlong}{1},{longpar}{1},{ParL}{1},%
      {parshort}{2},{shortpar}{2},{ParS}{2},%
      {parnumeric}{3},{numericpar}{3},{ParN}{3},%
      {paroff}{4},{nopar}{4},%
      {sentencelong}{10},{longsentence}{10},{SentenceL}{10},%
      {sentenceshort}{20},{shortsentence}{20},{SentenceS}{20},%
      {sentencenumeric}{30},{numericsentence}{30},{SentenceN}{30},%
      {sentenceoff}{40},{nosentence}{40},%
      {long}{11},%
      {short}{22},%
      {numeric}{33},%
      {paragraphonly}{44},{onlyparagraph}{44},%
      {ParagraphOnly}{44},{OnlyParagraph}{44}%
    }{#1}%
    \ifx\FamilyKeyState\FamilyKeyStateProcessed
      \aftergroup\FamilyKeyStateProcessed
      \@tempcnta=\@tempa\relax
      \@tempcntb=\z@
      \@whilenum \@tempcnta>9 \do{%
        \advance\@tempcnta -10\relax
        \advance\@tempcntb \@ne\relax
      }%
      \ifcase \@tempcnta
      \or
        \aftergroup\def\aftergroup\parcite@format
        \aftergroup{\aftergroup0\aftergroup}%
      \or
        \aftergroup\def\aftergroup\parcite@format
        \aftergroup{\aftergroup1\aftergroup}%
      \or
        \aftergroup\def\aftergroup\parcite@format
        \aftergroup{\aftergroup2\aftergroup}%
      \or
        \aftergroup\def\aftergroup\parcite@format
        \aftergroup{\aftergroup-\aftergroup1\aftergroup}%
      \fi
      \ifcase \@tempcntb
      \or
        \aftergroup\def\aftergroup\sentencecite@format
        \aftergroup{\aftergroup0\aftergroup}%
      \or
        \aftergroup\def\aftergroup\sentencecite@format
        \aftergroup{\aftergroup1\aftergroup}%
      \or
        \aftergroup\def\aftergroup\sentencecite@format
        \aftergroup{\aftergroup2\aftergroup}%
      \or
        \aftergroup\def\aftergroup\sentencecite@format
        \aftergroup{\aftergroup-\aftergroup1\aftergroup}%
      \fi
    \else
      \aftergroup\FamilyKeyStateUnknownValue
    \fi
  \endgroup
  \ifx\FamilyKeyState\FamilyKeyStateProcessed
    \KOMA@kav@removekey{contract.sty}{ref}%
    \ifcase\parcite@format
      \KOMA@kav@add{contract.sty}{ref}{parlong}%
    \or
```

```
      \KOMA@kav@add{contract.sty}{ref}{parshort}%
    \or
      \KOMA@kav@add{contract.sty}{ref}{parnumeric}%
    \or
      \KOMA@kav@add{contract.sty}{ref}{paroff}%
    \fi
    \ifcase\sentencecite@format
      \KOMA@kav@add{contract.sty}{ref}{sentencelong}%
    \or
      \KOMA@kav@add{contract.sty}{ref}{sentenceshort}%
    \or
      \KOMA@kav@add{contract.sty}{ref}{sentencenumeric}%
    \or
      \KOMA@kav@add{contract.sty}{ref}{sentenceoff}%
    \fi
  \fi
}
\KOMA@kav@add{contract.sty}{ref}{parlong}%
\KOMA@kav@add{contract.sty}{ref}{sentencelong}%
\@ifundefined{KOMA@DeclareDeprecatedOption}{}{%
  \KOMA@DeclareDeprecatedOption[contract]{parcitename}{ref=parlong}
  \KOMA@DeclareDeprecatedOption[contract]{sentencecitename}{ref=sentencelong}
}
⟨/options⟩
```

Execute the options.

```
⟨*postoptions⟩
\KOMAProcessOptions\relax
⟨/postoptions⟩
```

14 Deprecated Paragraph Commands

All these commands are deprecated and now generate error messages instead of code.

\Paragraph
\SubParagraph
\refParagraph
\refParagraphN
\DeprecatedParagraph
\ParagraphCompatibilityHacks

Todo: Remove deprecated commands, because they already throw errors for about nine years.

```
⟨*body⟩
\providecommand*{\DeprecatedParagraph}{%
  \PackageError{contract}{modification of old document needed}{%
    It seem that this document was made for scrjura up to version
    0.7a.\MessageBreak
    Since scrjura version 0.9 \string\Paragraph, \string\SubParagraph, and all
    depending\MessageBreak
    commands, options, and counters have been renamed.\MessageBreak
    You should replace the terms ‘Paragraph’ and ‘paragraph‘ by ‘Clause‘
    and\MessageBreak
    ‘clause‘ if they are part of the name of a contract feature, otherwise
    this\MessageBreak
    document may produce severall additional error messages and maybe the
    wrong\MessageBreak
    result. Sorry for the inconvenience.%
  }%
  \ParagraphCompatibilityHacks
}
\newcommand*{\ParagraphCompatibilityHacks}{%
  \PackageWarning{contract}{compatibility hacks for `\string\Paragraph'
    executed.\MessageBreak
    There is no support for documents using these hacks!\MessageBreak
    There is no warranty for real compatibility!\MessageBreak
    Even if the LaTeX run of the document doesn't report\MessageBreak
    any error, the result may be completely wrong.\MessageBreak
    Therefore it is recommended to solve the problem,\MessageBreak
    instead of trying to work around using the\MessageBreak
    compatibility hacks%
  }%
  \gdef\Paragraph{\Clause}%
  \gdef\SubParagraph{\SubClause}%
  \gdef\c@Paragraph{\c@Clause}%
  \gdef\cl@Paragraph{\cl@Clause}%
  \gdef\c@SubParagraph{\c@SubClause}%
  \gdef\cl@SubParagraph{\cl@SubClause}%
  \gdef\theParagraph{\theClause}%
  \gdef\theSubParagraph{\theSubClause}%
  \gdef\refParagraph{\refClause}%
  \gdef\refParagraphN{\refClauseN}%
  \aliaskomafont{Paragraph}{Clause}%
  \scr@ifundefinedorrelax{Paragraphmark}{}{%
    \global\let\Clausemark\Paragraphmark
  }%
}
\providecommand*{\Paragraph}{\DeprecatedParagraph\Paragraph}
\providecommand*{\SubParagraph}{\DeprecatedParagraph\SubParagraph}
\providecommand*{\refParagraph}{\DeprecatedParagraph\refParagraph}
\providecommand*{\refParagraphN}{\DeprecatedParagraph\refParagraphN}
⟨/body⟩
```

15 Contracts, Clauses, Paragraphs and Sentences

`\contract@env@type` This macro shows the currently active contract environment.

```
⟨*body⟩
\newcommand*{\contract@env@type}{}
⟨/body⟩
```

`\ellipsispar`
`\parellipsis` Count one or more paragraphs given by the optional argument but print `\parellipsis` instead of a real paragraph. The default is either `\dots` or `\textellipsis` if available.

```
⟨*body⟩
\newcommand*{\ellipsispar}[1][1]{%
  \begingroup
    \KOMAoptions{parnumber=manual}\parellipsis\par
    \addtocounter{par}{#1}%
    \if@filesw
      \protected@write\@auxout{}{%
        \string\newmaxpar{\contract@env@type}%
                         {\csname the\contract@env@type
                           AbsoluteClause\endcsname}%
                         {\thepar}%
      }%
    \fi
  \endgroup
  \addtocounter{par}{-1}\refstepcounter{par}%
  \ignorespaces
}
\newcommand*{\parellipsis}{%
  \scr@ifundefinedorrelax{textellipsis}{\dots}{\textellipsis}%
}
⟨/body⟩
```

`contract`
`\if@contract@skiphyperref`
`contractClause` (*cnt.*)
`\thecontractClause`
`\contract@Clauseformat`
`\Clauseformat`
`\paragraphformat`
`contractSubClause` (*cnt.*)
`\thecontractSubClause`
`contractAbsoluteClause` (*cnt.*)

It is not allowed to nest the `contract` environments, but you can end them and start them new. But this would not end the contract and start a new contract but only delay it for some other code.

```
⟨*body⟩
\newenvironment{contract}{%
  \ifx\contract@env@type\@empty
    \let\@doendpe\contract@doendpe
    \let\Clause\contract@paragraph
    \let\c@Clause\c@contractClause
    \edef\cl@Clause{\cl@Clause\cl@contractClause}%
    \let\SubClause\contract@subparagraph
    \let\c@SubClause\c@contractSubClause
    \edef\cl@SubClause{\cl@SubClause\cl@contractSubClause}%
    \let\Sentence\contract@sentence
    \renewcommand*{\contract@env@type}{contract}%
    \aliaskomafont{Clause}{contract.Clause}%
  \else
    \PackageError{contract}{nested ‘contract‘ detected}{%
      You may not use a ‘contract’ environment inside\MessageBreak
      a ‘\contract@env@type’ environment or after loading\MessageBreak
      package ‘contract’ with option ‘\contract@env@type’!}%
  \fi
}{}
\let\if@contract@skiphyperref\iftrue
\let\cl@Clause\@empty
\let\cl@SubClause\@empty
\newcounter{contractClause}
\renewcommand*{\thecontractClause}{%
  {\contract@Clauseformat{\arabic{Clause}}}}
\DeclareRobustCommand*{\contract@Clauseformat}[1]{\Clauseformat{#1}}
\newcommand*{\Clauseformat}[1]{\S~#1}
\newcounter{contractSubClause}
\@addtoreset{SubClause}{Clause}
\renewcommand*{\thecontractSubClause}{%
  {\theClause\alph{SubClause}}}
```

```
\newcounter{contractAbsoluteClause}
⟨/body⟩
```

`\DeclareNewJuraEnvironment` Using `\\c_atsign_strdefjuraenvironment` to define a new juristic environment. This can be done only in document preamble.

```
⟨*body⟩
\newcommand*{\DeclareNewJuraEnvironment}[1]{%
  \@ifundefined{#1}{\expandafter\let\csname #1\expandafter\endcsname
    \csname end#1\endcsname}{}%
  \@ifundefined{#1}{\let\reserved@defjuraenvironment\@defjuraenvironment}{%
    \PackageError{contract}{ignorring declaration of ‘#1’}{%
      You’ve tried to declare jura environment ‘#1’, but
      environment\MessageBreak
      ‘#1’ or command
      \expandafter\string\csname #1\endcsname\space or
      \expandafter\string\csname end#1\endcsname\MessageBreak
      already exists.\MessageBreak
      Declaration will be ignored}%
    \long\def\reserved@defjuraenvironment##1[##2]##3##4{}%
  }%
  \kernel@ifnextchar [%]
    {\reserved@defjuraenvironment{#1}}{\reserved@defjuraenvironment{#1}[]}%
}
\@onlypreamble\DeclareNewJuraEnvironment
```

`\@defjuraenvironment` This command is used to define a new contract environment like `contract`. Several options are provided (see the user manual for details).

```
\DefineFamily{KOMAarg}
\DefineFamilyMember{KOMAarg}
\newcommand{\@defjuraenvironment}{}
\long\def\@defjuraenvironment#1[#2]#3#4{%
  \let\reserved@defjuraenvironment\relax
```

The counters:

```
  \newcounter{#1Clause}%
  \newcounter{#1AbsoluteClause}%
  \newcounter{#1SubClause}%
  \FamilyCSKey[.contract.sty]{KOMAarg}{Clause}{#1@Clause}%
  \FamilyCSKey[.contract.sty]{KOMAarg}{SubClause}{#1@SubClause}%
  \FamilyCSKey[.contract.sty]{KOMAarg}{Sentence}{#1@Sentence}%
  \DefineFamilyKey[.contract.sty]{KOMAarg}{ClauseNumberFormat}{%
    \expandafter\def\csname #1@Clauseformat \endcsname####1{##1{####1}}%
    \expandafter\edef\csname #1@Clauseformat\endcsname{%
      \noexpand\protect\expandafter\noexpand\csname #1@Clauseformat \endcsname
    }%
    \FamilyKeyStateProcessed
  }
  \DefineFamilyKey[.contract.sty]{KOMAarg}{ClauseFont}{%
    \IfExistskomafont{#1.Clause}{%
      \IfIsAliaskomafont{#1.Clause}{%
        \expandafter\let\csname scr@fnt@instead@#1.Clause\endcsname\relax
        \newkomafont{#1.Clause}{##1}%
      }{\setkomafont{#1.Clause}{##1}}%
    }{%
      \newkomafont{#1.Clause}{##1}%
    }%
  }
  \FamilyExecuteOptions[.contract.sty]{KOMAarg}{#2}%
  \RelaxFamilyKey[.contract.sty]{KOMAarg}{ClauseFont}%
  \RelaxFamilyKey[.contract.sty]{KOMAarg}{ClauseNumberFormat}%
  \RelaxFamilyKey[.contract.sty]{KOMAarg}{Sentence}%
  \RelaxFamilyKey[.contract.sty]{KOMAarg}{SubClause}%
  \RelaxFamilyKey[.contract.sty]{KOMAarg}{Clause}%
  \@ifundefined{#1@Clauseformat}{%
    \expandafter\DeclareRobustCommand\expandafter*%
    \csname #1@Clauseformat\endcsname[1]{\Clauseformat{##1}}%
  }{}%
```

```
\expandafter\renewcommand\expandafter*\csname the#1Clause\endcsname{%
  \protect\@nameuse{#1@Clauseformat}{\arabic{#1Clause}}}%
```

Environment:

```
\newenvironment{#1}{%
  \par
  \ifx\contract@env@type\@empty
    \edef\contract@env@type{#1}%
    \let\@doendpe\contract@doendpe
    \expandafter\let\expandafter\c@Clause\csname c@#1Clause\endcsname
    \edef\cl@Clause{\cl@Clause\csname cl@#1Clause\endcsname}%
    \expandafter\let\expandafter\c@SubClause
      \csname c@#1SubClause\endcsname
    \edef\cl@SubClause{\cl@SubClause
      \csname cl@#1SubClause\endcsname}%
    \@ifundefined{#1@Clause}{%
      \let\Clause\contract@paragraph
    }{%
      \expandafter\let\expandafter\Clause
      \csname #1@Clause\endcsname
    }%
    \@ifundefined{#1@SubClause}{%
      \let\SubClause\contract@subparagraph
    }{%
      \expandafter\let\expandafter\SubClause
      \csname #1@SubClause\endcsname
    }%
    \@ifundefined{#1@Sentence}{%
      \let\Sentence\contract@sentence
    }{%
      \expandafter\let\expandafter\Sentence\csname #1@Sentence\endcsname
    }%
    \@ifundefined{\contract@env@type @everypar}{%
      \expandafter\let
      \csname \contract@env@type @everypar\endcsname
      \contract@everypar
      }{}%
```

Font alias for `Clause`. If neither a font not an alias is defined for the new environment `contract.Clause` is used.

```
      \IfExistskomafont{#1.Clause}{%
        \IfIsAliaskomafont{#1.Clause}{%
          \aliaskomafont{Clause}{\csname scr@fnt@instead@#1.Clause\endcsname}%
        }{%
          \aliaskomafont{Clause}{#1.Clause}%
        }%
      }{%
        \aliaskomafont{Clause}{contract.Clause}%
      }%
      #3%
    \else
      \PackageError{contract}{nested contract environments detected}{%
        You must not use a `#1' environment inside\MessageBreak
        a `\contract@env@type' environment or after loading\MessageBreak
        package `contract' with option `\contract@env@type'!}%
    \fi
  }{%
    #4%
    \par
  }%
}
⟨/body⟩
```

`\contract@paragraph`

This is the `\Clause` used by contracts. A contract consists (usually) of several clauses. Each clause has optional elements managed by ⟨*key*⟩=⟨*value*⟩ pairs handled by scrkbase and last but not least by keyval.

`title`
`head`
`nohead`
`entry`
`noentry`
`tocentry`
`notocentry`

Title, running head and toc entry of the clause. The title is the default for running head and toc entry. But you can also use an empty value for each of them or use the `no...` options to switch them off.

```
⟨*body⟩
\define@key{contract}{title}{%
  \def\contract@title{#1}%
  \ifx\contract@entry\relax\def\contract@entry{\contract@title}\fi
  \ifx\contract@head\relax\def\contract@head{\contract@title}\fi
}
\define@key{contract}{entry}{%
  \PackageWarning{contract}{deprecated option `entry'.\MessageBreak
    You should use option `tocentry' instead of\MessageBreak
    option `entry'%
  }%
  \def\contract@entry{#1}}
\define@key{contract}{tocentry}{\def\contract@entry{#1}}
\define@key{contract}{noentry}[]{%
  \PackageWarning{contract}{deprecated option `noentry'.\MessageBreak
    You should use option `notocentry' instead of\MessageBreak
    option `noentry'%
  }%
  \let\contract@entry\relax}
\define@key{contract}{notocentry}[]{\let\contract@entry\relax}
\define@key{contract}{head}{\def\contract@head{#1}}
\define@key{contract}{nohead}[]{\let\contract@head\relax}
```

`number`

The number can be changed manually. But clauses without numbers are not allowed. So if you use an empty value, the number is automatically set.

```
\define@key{contract}{number}{\def\contract@number{#1}}
```

`\contract@preskip`
`\contract@postskip`
`preskip`
`postskip`

The options are used to specify the distance before and after the clause. The preset value of these options are the global settings done by `\setkeys{contract}{...}`.

```
\newcommand*{\contract@preskip}{2\baselineskip}
\newcommand*{\contract@postskip}{\baselineskip}
\define@key{contract}{preskip}{\def\contract@preskip{#1}}
\define@key{contract}{postskip}{\def\contract@postskip{#1}}
```

`dummy`

The option switches the boolean `\ifcontract@dummy`.

`\ifcontract@dummy`

If the boolean is `\iftrue` the clause will not be printed. But note: you cannot use this to remove the paragraphs or sentences of the clause. But you can use this option to generate holes in the numbering without manually manipulating the counters.

```
\newif\ifcontract@dummy
\define@key{contract}{dummy}[true]{\csname contract@dummy#1\endcsname}
```

`contract.Clause` (*font*)
`\contract@paragraph@font`

Correctly this macro should be named `\contract@paragraph@format`. But it is already used by some users for ugly tricks. So I will not rename it to avoid problems for existing documents. Additionally it would be better to use a new macro per environment. However the same reason not to change this.

```
\newkomafont{contract.Clause}{\sffamily\bfseries\large}
\newcommand*{\contract@paragraph@font}{\usekomafont{Clause}%
  \@hangfrom}
```

`@AbsClause` (*cnt.*)
`\theH@AbsClause`
`\theHClause`
`\theHSubClause`

```
% Here we have some not good tested code for \pkg{hyperref}.
\newcounter{@AbsClause}
\def\theH@AbsClause{P-\arabic{@AbsClause}}
\def\theHClause{\theH@AbsClause}
\def\theHSubClause{\theH@AbsClause}
```

For the headings we use manual paragraph numbering, because we don't want any paragraph numbering inside the heading. After initializing the options they are processed.

```
\NewDocumentCommand\contract@paragraph {o} {%
  \stepcounter{\contract@env@type AbsoluteClause}%
  \ManualPar\parnumbertrue
  \let\contract@title\relax
  \let\contract@entry\relax
  \let\contract@head\relax
  \let\contract@number\relax
  \contract@dummyfalse
  \IfValueT{#1}{\setkeys{contract}{#1}}%
```

Unless this is a dummy clause, the headings will be initialized and vertical skips will be done.

```
\ifcontract@dummy\else
  \par
  \@afterindentfalse
  \addvspace{\contract@preskip}%
\fi
```

If there isn't a manual number, we use the next number. If there is a manual number, this number is printed and we take care that labels and hyperref also use the manual number.

```
\ifx\contract@number\relax
  \let\p@Clause\@empty
  \expandafter\let\expandafter\theClause
    \csname the\contract@env@type Clause\endcsname
  \refstepcounter{Clause}%
\else
  \begingroup
    \let\@elt\@stpelt
    \cl@Clause
  \endgroup

  \protected@edef\theClause{%
    \protect\@nameuse{\contract@env@type @Clauseformat}{\contract@number}%
  }%
  \protected@edef\@currentlabel{\theClause}%
  \def\@currentcounter{Clause}%
\fi
\stepcounter{@AbsClause}%
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname if@skiphyperref\endcsname\relax
\else
  \expandafter\let\csname if@contract@skiphyperref\expandafter\endcsname
  \csname if@skiphyperref\endcsname
\fi
\if@contract@skiphyperref\else
  \hyper@refstepcounter{@AbsClause}%
⟨+trace⟩    \typeout{absolute Number: \the@AbsClause^^JLabel: `\@currentHref'}%
\fi
```

For simplification we use the code of clauses for sub-clauses.

```
\let\theSubClause\theClause
```

Unless for dummy clauses, the heading is printed, the toc entry is done and also the running head.

```
\ifcontract@dummy\else
  \begingroup
    \if@juratitlepagebreak\else\interlinepenalty\@M\fi
    \contract@paragraph@font{\theClause
      \ifx\contract@title\relax\else\enskip\fi}%
    \contract@title
    \ifx\contract@entry\relax\else
      \expandafter\addxcontentsline\expandafter{\ext@toc}%
      {cpar}[\theClause]\contract@entry
```

```
        \addxcontentsline{cpa}{cpar}[\theClause]\contract@entry
      \fi
      \ifx\contract@head\relax\else
        \expandafter\Clausemark\expandafter{%
          \expandafter\theSubClause\expandafter\enskip\contract@head}%
      \fi
      \par
    \endgroup\nobreak\vskip\contract@postskip
```

Last but not least paragraph numbering is initialized.

```
    \contract@afterheading
  \fi
}
⟨/body⟩
```

`\contract@subparagraph` This is almost the same like `\contract@paragraph`.

```
⟨*body⟩
\NewDocumentCommand \contract@subparagraph {o}{%
  \stepcounter{\contract@env@type AbsoluteClause}%
  \ManualPar\parnumbertrue
  \let\contract@title\relax
  \let\contract@entry\relax
  \let\contract@head\relax
  \let\contract@number\relax
  \contract@dummyfalse
  \IfValueT{#1}{\setkeys{contract}{#1}}%
  \ifcontract@dummy\else
    \par
    \@afterindentfalse
    \vskip\contract@preskip
  \fi
  \ifx\contract@number\relax
    \let\p@SubClause\@empty
    \let\theSubClause\thecontractSubClause
    \refstepcounter{SubClause}%
  \else
    \begingroup
      \let\@elt\@stpelt
      \cl@SubClause
    \endgroup
    \protected@edef\theSubClause{\theClause\contract@number}%
    \protected@edef\@currentlabel{\theSubClause}%
    \def\@currentcounter{SubClause}%
  \fi
  \stepcounter{@AbsClause}%
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname if@skiphyperref\endcsname\relax
  \else
    \expandafter\let\csname if@contract@skiphyperref\expandafter\endcsname
    \csname if@skiphyperref\endcsname
  \fi
  \if@contract@skiphyperref\else
```

```
    \hyper@refstepcounter{@AbsClause}%
⟨+trace⟩     \typeout{absolute Number: \the@AbsClause^^JLabel: ‘\@currentHref’}%
  \fi
  \ifcontract@dummy\else
    \begingroup
      \if@juratitlepagebreak\else\interlinepenalty\@M\fi
      \contract@paragraph@font{\theSubClause
        \ifx\contract@title\relax\else\enskip\fi}%
      \contract@title
      \ifx\contract@entry\relax\else
        \expandafter\addxcontentsline\expandafter{\ext@toc}%
        {cpar}[\theSubClause]\contract@entry
        \addxcontentsline{cpa}{cpar}[\theSubClause]\contract@entry
      \fi
      \ifx\contract@head\relax\else
        \expandafter\Clausemark\expandafter{%
          \expandafter\theSubClause\expandafter\enskip\contract@head}%
      \fi
      \par
    \endgroup
    \nobreak\vskip\contract@postskip
    \contract@afterheading
  \fi
}
⟨/body⟩
```

`\AutoPar` `\ManualPar` Switching between automatic or manual paragraph numbers for all contract environments.

```
⟨*body⟩
\newcommand*{\AutoPar}{%
  \expandafter\let\expandafter\contract@used@everypar
  \csname \contract@env@type @everypar\endcsname
}
\newcommand*{\ManualPar}{%
  \let\contract@used@everypar\relax
}
⟨/body⟩
```

`\contract@afterheading` Similar to `\afterheading` but with automatic paragraph numbers.

ToDo: Test if this can be done using LaTeX hooks, depending on the LaTeX release.

```
⟨*body⟩
\CheckCommand*{\@afterheading}{%
  \@nobreaktrue
  \everypar{%
    \if@nobreak
      \@nobreakfalse
      \clubpenalty \@M
      \if@afterindent \else
        {\setbox\z@\lastbox}%
      \fi
    \else
```

```
      \clubpenalty \@clubpenalty
      \everypar{}%
    \fi}%
}
\newcommand*{\contract@afterheading}{%
  \@nobreaktrue
  \everypar{%
    \if@nobreak
      \@nobreakfalse
      \clubpenalty \@M
      \if@afterindent \else
        {\setbox\z@\lastbox}%
      \fi
    \else
      \clubpenalty \@clubpenalty
      \everypar{%
        \contract@used@everypar
      }%
    \fi
    \contract@used@everypar
  }%
  \AutoPar
}
```

`\contract@used@everypar` The macro to be used at the very beginning of every paragraph to add the number. To be used only inside contract environments, so empty outside.

```
\newcommand*{\contract@used@everypar}{}
```

`\@doendpe` LaTeX used this macro, to reset all paragraph actions at the end of environments. To avoid unwanted switching-off of the paragraph number it will be reinitialized.

`\contract@doendpe`
`\IncludeInRelease`
`\@gobble@IncludeInRelease`
`\EndIncludeInRelease`

From LaTeX 2015/01/01 a different definition of `\\c_atsign_strdoendpe` is used. So we also have to use different versions depending on the release. We do so with some tricks. Maybe this should be replaced by usage of `\IfLTXAtLeastTF` from already loaded scrbase.

Todo: Using `\everypar` is evil and should be replaced by using a generic paragraph hook.

```
\providecommand*{\IncludeInRelease}[3]{%
  \PackageInfo{contract}{temporary definition of \string\IncludeInRelease}%
  \Ifstr{#1}{0000/00/00}{%
    \let\IncludeInRelease\@undefined
    \def\EndIncludeInRelease{\let\EndIncludeInRelease\@undefined}%
  }{%
    \let\EndIncludeInRelease\relax
    \long\def\@gobble@IncludeInRelease##1\EndIncludeInRelease{%
      \let\@gobble@IncludeInRelease\@undefined
    }%
    \expandafter\@gobble@IncludeInRelease
  }%
}
\IncludeInRelease{2015/01/01}{\@doendpe}{clubpenalty fix}
\CheckCommand*\@doendpe{\@endpetrue
     \def\par{\@restorepar
```

```
              \clubpenalty\@clubpenalty
              \everypar{}\par\@endpefalse}\everypar
               {{\setbox\z@\lastbox}%
                \everypar{}\@endpefalse}}
\newcommand*{\contract@doendpe}{%
  \@endpetrue
  \def\par{%
    \@restorepar
    \clubpenalty\@clubpenalty
    \everypar{%
      \csname contract@used@everypar\endcsname
    }%
    \par\@endpefalse
  }%
  \everypar{%
    {\setbox\z@\lastbox}\everypar{%
      \csname contract@used@everypar\endcsname
    }%
    \@endpefalse
  }%
}
\EndIncludeInRelease
\IncludeInRelease{0000/00/00}{\@doendpe}{clubpenalty fix}
\CheckCommand*\@doendpe{\@endpetrue
  \def\par{\@restorepar\everypar{}\par\@endpefalse}\everypar
  {{\setbox\z@\lastbox}\everypar{}\@endpefalse}}
\newcommand*{\contract@doendpe}{%
  \@endpetrue
  \def\par{%
    \@restorepar\everypar{%
      \csname contract@used@everypar\endcsname
    }%
    \par\@endpefalse
  }%
  \everypar{%
    {\setbox\z@\lastbox}\everypar{%
      \csname contract@used@everypar\endcsname
    }%
    \@endpefalse
  }%
}
\EndIncludeInRelease
⟨/body⟩
```

16 Entry to Table of Contents

`\l@cpar` Toc entry of contract clauses. This is done using tocbasic. The definition has to be part of the initialization of the package, otherwise package options wouldn’t be able to change the setting.

```
⟨*init⟩
```

```
\DeclareTOCStyleEntry[%
  indent=1.5em,
  numwidth=2em,
  level=\maxdimen
]{default}{cpar}
⟨/init⟩
```

17 Numbering of Paragraphs and Sentences

`\contract@separator`

Used to make it possible to remove white spaces at the beginning or end.

```
⟨*body⟩
\DeclareRobustCommand*{\contract@separator}[1]{#1}
⟨/body⟩
```

`\contract@usetype`

By default it is robust but does only call `\contract@@usetype` with the only argument.

`\contract@@usetype`

This second command is not robust and can easily be redefined. But by default it also does nothing but eating the argument.

```
⟨*body⟩
\DeclareRobustCommand*{\contract@usetype}[1]{\contract@@usetype{#1}}
\newcommand*{\contract@@usetype}[1]{}
⟨/body⟩
```

`\contract@everypar`

The `\contract@everpar` used by contracts.

`\ifparnumber`
`\parnumbertrue`
`\parnumberfalse`
`par` (*cnt.*)
`\thepar`
`\theHpar`
`\parformat`
`\parformatseparation`
`\p@par`
`\withoutparnumber`

The boolean defines if paragraph numbers have to be used. If they are deactivated also manual paragraph numbers are deactivated and the paragraphs are not counted. Otherwise the paragraphs are numbered using `\thepar`. It is important to reset the paragraph counter with every clause and sub-clause. And for labels the parent object the clause has to be used.

```
⟨*body⟩
\newcounter{par}
\renewcommand*{\thepar}{\arabic{par}}
\def\theHpar{\theH@AbsClause-\Roman{par}}
\newcommand*{\parformat}{(\thepar)}
\newcommand*{\parformatseparation}{\nobreakspace}
\newkomafont{parnumber}{}
\renewcommand*\p@par{{\contract@usetype{\contract@env@type}\theSubClause\contract@separator
\@addtoreset{par}{Clause}
\@addtoreset{par}{SubClause}
\newcommand*{\withoutparnumber}{}
⟨/body⟩
```

```
⟨*body⟩
\newcommand*{\contract@everypar}{%
  \ifparnumber
    \ifx\contract@special@par\relax
      \ifx\contract@special@reset@par\relax\else
        \global\let\thepar\contract@special@reset@par
        \global\let\contract@special@reset@par\relax
      \fi
      \refstepcounter{par}%
      \refstepcounter{sentence}%
    \else
      \ifx\contract@special@reset@par\relax
        \global\let\contract@special@reset@par\thepar
      \fi
      \global\let\thepar\contract@special@par
      \global\let\contract@special@par\relax
      \setcounter{sentence}{0}\refstepcounter{sentence}%
    \fi
    \begingroup
      \if@filesw
        \protected@write\@auxout{%
          \expandafter\let\csname \contract@env@type @Clauseformat\endcsname
          \@firstofone
        }{%
          \string\newmaxpar{\contract@env@type}%
                           {\csname the\contract@env@type
                             AbsoluteClause\endcsname}%
                           {\thepar}%
        }%
      \fi
      \getmaxpar\@tempa{\contract@env@type}%
                       {\csname the\contract@env@type AbsoluteClause\endcsname}%
⟨+trace⟩     \typeout{Stored max is \@tempa}%
      \def\reserved@a##1\@nnil{\def\@tempa{##1}}%
      \afterassignment\reserved@a\@tempcnta=0\@tempa\relax\@nnil
      \ifnum \@tempcnta>\@ne
        {\usekomafont{parnumber}{\parformat\parformatseparation}}%
      \else
        \def\reserved@a{\relax}%
        \ifx\@tempa\reserved@a
          \withoutparnumber
        \else
          {\usekomafont{parnumber}{\parformat\parformatseparation}}%
        \fi
      \fi
    \endgroup
  \else
    \begingroup\withoutparnumber\endgroup
    \setcounter{sentence}{-1}\refstepcounter{sentence}%
  \fi
}
⟨/body⟩
```

`\thisparnumber`
`\contract@special@par`
`\contract@special@reset@par`

You can use this for manual paragraph numbering. But the number has to be fully expandable!

```
⟨*body⟩
\newcommand*{\thisparnumber}[1]{%
  \def\contract@special@par{#1}%
}
\newcommand*{\contract@special@par}{}
```

```
\let\contract@special@par\relax
\newcommand*{\contract@special@reset@par}{}
\let\contract@special@reset@par\relax
⟨/body⟩
```

18 Referencing

`\refL` `\ref@L`

Similar to `\ref` but always the long form.

```
⟨*body⟩
\newcommand*{\refL}{\kernel@ifstar {\ref@L*}{\ref@L{}}}
\newcommand*{\ref@L}[2]{%
  \begingroup
    \def\parcite@format{0}%
    \let\sentencecite@format\parcite@format
    \ref#1{#2}%
  \endgroup
}
```

`\refS` `\ref@S`

Similar to `\ref` but always the short form.

```
\newcommand*{\refS}{\kernel@ifstar {\ref@S*}{\ref@S{}}}
\newcommand*{\ref@S}[2]{%
  \begingroup
    \def\parcite@format{1}%
    \let\sentencecite@format\parcite@format
    \ref#1{#2}%
  \endgroup
}
```

`\refN` `\ref@N`

Similar to `\ref` but always the numerical form.

```
\newcommand*{\refN}{\kernel@ifstar {\ref@N*}{\ref@N{}}}
\newcommand*{\ref@N}[2]{%
  \begingroup
    \def\parcite@format{2}%
    \let\sentencecite@format\parcite@format
    \ref#1{#2}%
  \endgroup
}
```

`\refClause` `\ref@Clause`

Reference only the clause of a clause, paragraph or sentence. For better compatibility with hyperref there is also a star version if hyperref is used. Without hyperref the star version is nonsense.

```
\newcommand*{\refClause}{%
  \kernel@ifstar {\ref@Clause*}{\ref@Clause{}}
}
\newcommand*{\ref@Clause}[2]{%
  \expandafter\ifx\csname r@#2\endcsname\relax
    \ref#1{#2}%
  \else
    \begingroup
```

Copy all parts of the reference but the first one to `\\c_atsign_strtempb`.

```
      \expandafter\expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\def
      \expandafter\expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\@tempb
      \expandafter\expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter\@gobble\csname r@#2\endcsname}%
```

Copy the first part of the reference to `\\c_atsign_strtempa`.

```
      \def\@tempc##1##2\@nil{##1}%
      \let\contract@separator\@gobble
      \protected@edef\@tempa{\expandafter\expandafter\expandafter\@tempc
        \csname r@#2\endcsname\noexpand\@nil}%
```

Copy the first part of `\\c_atsign_strtempa` to `\\c_atsign_strtempb`.

```
      \protected@edef\@tempa{\expandafter\expandafter\expandafter\@tempc
        \@tempa\@nil}%
      \let\@@protect\protect
      \let\protect\noexpand
      \expandafter\edef\csname r@#2\endcsname{{\@tempa}\@tempb}%
      \let\protect\@@protect
      \ref#1{#2}%
    \endgroup
  \fi
}
```

`\refClauseN`
`\ref@ClauseN`

Reference only the clause number of a clause, a paragraph or a sentence. For improved compatibility with hyperref there is also a star version if hyperref is used. Without hyperref the star version is nonsense.

```
\newcommand*{\refClauseN}{%
  \kernel@ifstar {\ref@ClauseN*}{\ref@ClauseN{}}
}
\newcommand*{\ref@ClauseN}[2]{%
  \begingroup
    \let\Clauseformat\relax
    \ref@Clause{#1}{#2}%
  \endgroup
}
```

`\refPar`
`\ref@Par`

References only the paragraph of a paragraph or sentence. For improved compatibility with hyperref there is also a star version if hyperref is used. Without hyperref the star version is nonsense.

```
\newcommand*{\refPar}{%
  \kernel@ifstar {\ref@Par*}{\ref@Par{}}
}
\newcommand*{\ref@Par}[2]{%
  \expandafter\ifx\csname r@#2\endcsname\relax
    \ref#1{#2}%
  \else
    \begingroup
```

Copy all parts of the reference but the first one to \\c_atsign_strtempb.

```
      \expandafter\expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\def
      \expandafter\expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\@tempb
      \expandafter\expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter\@gobble\csname r@#2\endcsname}%
```

Copy the first part of the reference to \\c_atsign_strtempa.

```
      \def\@tempc##1##2\@nil{##1}%
      \let\contract@separator\@gobble
      \protected@edef\@tempa{\expandafter\expandafter\expandafter\@tempc
        \csname r@#2\endcsname\noexpand\@nil}%
```

Copy the second part of \\c_atsign_strtempa to \\c_atsign_strtempa ablegen.

```
      \def\@tempc##1##2##3\@nil{##2}%
      \protected@edef\@tempa{\expandafter\expandafter\expandafter\@tempc
        \@tempa{%
          \protect\G@refundefinedtrue
          \nfss@text{\reset@font\bfseries ??}%
          \@latex@warning{Reference `#2' on page \thepage \space
            with undefined par number}%
        }\noexpand\@nil}%
      \let\@@protect\protect
      \let\protect\noexpand
      \expandafter\edef\csname r@#2\endcsname{{\@tempa}\@tempb}%
      \let\protect\@@protect
      \ref#1{#2}%
    \endgroup
  \fi
}
```

\refParL
\ref@ParX

The same but long.

```
\newcommand*{\refParL}{%
  \kernel@ifstar {\ref@ParX0*}{\ref@ParX0{}}
}
\newcommand*{\ref@ParX}[3]{%
  \begingroup
    \def\parcite@format{#1}%
    \let\sentencecite@format\parcite@format
    \ref@Par{#2}{#3}%
  \endgroup
}
```

\refParS

The same but short.

```
\newcommand*{\refParS}{%
  \kernel@ifstar {\ref@ParX1*}{\ref@ParX1{}}
}
```

`\refParN`
`\ref@ParN`
`\ref@@ParN`

The same but numerical.

```
\newcommand*{\refParN}{%
  \kernel@ifstar {\ref@ParN2*}{\ref@ParN2{}}
}
\newcommand*{\ref@ParN}[2]{%
  \kernel@ifnextchar [%]
    {\ref@@ParN{#1}{#2}}%
    {\ref@ParX{#1}{#2}}%
}
\newcommand*{\ref@@ParN}{}
\def\ref@@ParN#1#2[#3]#4{%
  \begingroup
    \renewcommand*{\parnumericformat}[1]{%
      \csname @#3\endcsname{\number ##1\relax}%
    }%
    \ref@ParX{#1}{#2}{#4}%
  \endgroup
}
```

`\refSentence`
`\ref@Sentence`

Reference only the sentence of a sentence. For improved compatibility with hyperref there is also a star version if hyperref is used. Without hyperref the star version is nonsense.

```
\newcommand*{\refSentence}{%
  \kernel@ifstar {\ref@Sentence*}{\ref@Sentence{}}
}
\newcommand*{\ref@Sentence}[2]{%
  \expandafter\ifx\csname r@#2\endcsname\relax
    \ref#1{#2}%
  \else
    \begingroup
```

Copy all parts of the reference to `\\c_atsign_strtempb`.

```
      \expandafter\expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\def
      \expandafter\expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter\@tempb
      \expandafter\expandafter\expandafter\expandafter
      \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter\@gobble\csname r@#2\endcsname}%
```

Copy the first part of the reference to `\\c_atsign_strtempa`.

```
      \def\@tempc##1##2\@nil{##1}%
      \let\contract@separator\@gobble
      \protected@edef\@tempa{\expandafter\expandafter\expandafter\@tempc
        \csname r@#2\endcsname\noexpand\@nil}%
```

Copy the third part of `\\c_atsign_strtempa` to `\\c_atsign_strtempa`.

```
      \def\@tempc##1##2##3##4\@nil{##3}%
      \protected@edef\@tempa{\expandafter\expandafter\expandafter\@tempc
        \@tempa{}{%
          \protect\G@refundefinedtrue
          \nfss@text{\reset@font\bfseries ??}%
          \@latex@warning{Reference ‘#2’ on page \thepage \space
```

```
            with undefined sentence number}%
        }\noexpand\@nil}%
      \let\@@protect\protect
      \let\protect\noexpand
      \expandafter\edef\csname r@#2\endcsname{{\@tempa}\@tempb}%
      \let\protect\@@protect
      \ref#1{#2}%
    \endgroup
  \fi
}
```

`\refSentenceL`
`\ref@SentenceX`
The same but long.

```
\newcommand*{\refSentenceL}{%
  \kernel@ifstar {\ref@SentenceX0*}{\ref@SentenceX0{}}
}
\newcommand*{\ref@SentenceX}[3]{%
  \begingroup
    \def\parcite@format{#1}%
    \let\sentencecite@format\parcite@format
    \ref@Sentence{#2}{#3}%
  \endgroup
}
```

`\refSentenceS`
The same but short.

```
\newcommand*{\refSentenceS}{%
  \kernel@ifstar {\ref@SentenceX1*}{\ref@SentenceX1{}}
}
```

`\refSentenceN`
The same but numeric.

```
\newcommand*{\refSentenceN}{%
  \kernel@ifstar {\ref@SentenceX2*}{\ref@SentenceX2{}}
}
⟨/body⟩
```

`\contract@sentence`
Numbering of sentences.

`sentence` (*cnt.*)
`\thesentence`
`\theHsentence`
`\p@sentence`
The counter is used for numbering the sentences. It is important to add the paragraph as parent object to labels. The original method to make it possible to use `\thesentence` as an argument of `\p@sentence` does not work any longer using LaTeX 2019-10-01 or newer. It would result in an error message. So the code has to be adapted to the new definition of `\refstepcounter` in LaTeX 2019-10-01. From this version it uses `\labelformat`. Don't ask me, what I think about the fact, that every new versions of LaTeX can break existing packages and package authors have to find out such incompatibilities on their own.

```
⟨*body⟩
\newcounter{sentence}[par]
\renewcommand*{\thesentence}{\arabic{sentence}}
\def\theHsentence{\theHpar-\arabic{sentence}}
\scr@ifundefinedorrelax{labelformat}{%
  \renewcommand*{\p@sentence}{\expandafter\p@@sentence}
  \newcommand*{\p@@sentence}[1]{\p@par{{\par@cite{\thepar}}%
      \contract@separator{\nobreakspace}}{\sentence@cite{#1}}}%
}{%
  \labelformat{sentence}{\p@par{{\par@cite{\thepar}}%
      \contract@separator{\nobreakspace}}{\sentence@cite{#1}}}%
}
\newcommand*{\contract@sentence}{%
```

For the numbering it is important not to increase the paragraph number at the very beginning, because the paragraph already does so. To make this work, the paragraph has to start before we print the number. But immediately after a `minipage`, a list or a `\parbox` we should behave as not being at the beginning of a paragraph.

```
  \ifvmode
    \if@endpe
      \refstepcounter{sentence}%
    \else
      \leavevmode
    \fi
  \else
    \refstepcounter{sentence}%
  \fi
  {\usekomafont{sentencenumber}{\sentencenumberformat}}%
  \nobreak\hskip\z@
}
```

`sentencenumber` (*font*)
`\sentencenumberformat`

Formatting an font can be changed using font element `sentencenumber` and command `\sentencenumberformat`. The last has the preset `\textsuperscript`.

```
\newkomafont{sentencenumber}{}
\newcommand*{\sentencenumberformat}{\textsuperscript{\thesentence}}
⟨/body⟩
```

`\par@cite`
`\parciteformat`

Reference style for paragraphs.

```
⟨*body⟩
\DeclareRobustCommand*{\par@cite}[1]{\parciteformat{#1}}
\newcommand*{\parciteformat}[1]{%
  \ifcase \parcite@format
    \expandafter\parlongformat
  \or
    \expandafter\parshortformat
  \or
    \expandafter\parnumericformat
  \else
    \unskip\expandafter\@gobble
  \fi
  {#1}%
}
```

`\sentence@cite`
`\sentenceciteformat`

Reference style for sentences. Preset is `\\c_atsign_strarabic`.

```
\DeclareRobustCommand*{\sentence@cite}[1]{\sentenceciteformat{#1}}
\newcommand*{\sentenceciteformat}[1]{%
  \ifcase \sentencecite@format
    \expandafter\sentencelongformat
  \or
    \expandafter\sentenceshortformat
  \or
    \expandafter\sentencenumericformat
  \else
    \unskip\expandafter\@gobble
  \fi
  {#1}%
}
```

\parlongformat
\parshortformat
\parnumericformat
\sentencelongformat
\sentenceshortformat
\sentencenumericformat

The six formattings.

```
\newcommand*{\parlongformat}[1]{\parname~#1}
\newcommand*{\parshortformat}[1]{\parshortname~#1}
\newcommand*{\parnumericformat}[1]{\@Roman{\number #1\relax}}
\newcommand*{\sentencelongformat}[1]{\sentencename~#1}
\newcommand*{\sentenceshortformat}[1]{\sentenceshortname~#1}
\newcommand*{\sentencenumericformat}[1]{\@arabic{\number #1\relax}.}
⟨/body⟩
```

19 Language Dependent Names

\parname
\parshortname
\sentencename
\sentenceshortname
\contract@lang@error

The names of paragraphs and sentences and their short versions. The English names are donated by “m.eik”.

```
⟨*body⟩
\newcommand*{\parname}{Paragraph}
\AtBeginDocument{%
  \providecaptionname{german,ngerman,austrian,naustrian}\parname{Absatz}%
  \providecaptionname{german,ngerman,austrian,naustrian}\parshortname{Abs.}%
  \providecaptionname{german,ngerman,austrian,naustrian}\sentencename{Satz}%
  \providecaptionname{german,ngerman,austrian,naustrian}\sentenceshortname{S.}%
  \providecaptionname{english,american,british,canadian,%
    USenglish,UKenglish,usenglish,ukenglish}\parname{paragraph}%
  \providecaptionname{english,american,british,canadian,%
    USenglish,UKenglish,usenglish,ukenglish}\parshortname{par.}%
  \providecaptionname{english,american,british,canadian,%
    USenglish,UKenglish,usenglish,ukenglish}\sentencename{sentence}%
  \providecaptionname{english,american,british,canadian,%
    USenglish,UKenglish,usenglish,ukenglish}\sentenceshortname{sent.}%
}
\providecommand*{\parname}{\contract@lang@error{\parname}}
\providecommand*{\parshortname}{\contract@lang@error{\parshortname}}
\providecommand*{\sentencename}{\contract@lang@error{\sentencename}}
\providecommand*{\sentenceshortname}{\contract@lang@error{\sentenceshortname}}
\newcommand*{\contract@lang@error}[1]{%
  \PackageError{contract}{%
    current language not supported%
  }{%
    Currently contract only supports languages ‘german’, ‘ngerman’,
    ‘austrian’,\MessageBreak
    ‘naustrian’, ‘english’, ‘american’, ‘british’, ‘canadian’,
    ‘USenglish’,\MessageBreak
    ‘UKenglish’, ‘usenglish‘, and ‘ukenglish’.\MessageBreak
    It seems, that you are using another language (maybe ‘\languagename’) or
    that\MessageBreak
    your language selection isn’t compatible to package ‘babel’.\MessageBreak
    Because of this you have to define ‘\string#1’ by yourself!\MessageBreak
    It would be nice if you’ll send your definitions to the author.%
  }%
  \textbf{??}%
}
⟨/body⟩
```

20 Using Values from last LaTeX Run

\newmaxpar
\getmaxpar

Two helper macros, to save a counter in a aux-file and get the value back or another value of it is not in the aux-file.

```
⟨*body⟩
\newcommand*{\newmaxpar}[3]{%
  \begingroup
    \expandafter\let\csname #1@Clauseformat\endcsname\@firstofone
    \protected@edef\@tempa{#2}\@onelevel@sanitize\@tempa
    \expandafter\xdef\csname max@#1@\@tempa\endcsname{#3}%
  \endgroup
}
\newcommand*{\getmaxpar}[3]{%
  \begingroup
    \expandafter\let\csname #2@Clauseformat\endcsname\@firstofone
    \protected@edef\@tempa{#3}%
    \@onelevel@sanitize\@tempa
    \expandafter\ifx \csname max@#2@\@tempa\endcsname\relax
      \edef\@tempa{\endgroup\edef\noexpand#1{\expandafter\the\value{par}}}%
    \else
      \edef\@tempa{\endgroup
        \edef\noexpand#1{\csname max@#2@\@tempa\endcsname}}%
    \fi
  \@tempa
}
```

Because some users remove `contract` from their documents without deleting the aux-file, we add a fallback definition of `\newmaxpar` to the aux-file. This avoids error messages because of undefined `\newmaxpar`.

```
\AtBeginDocument{%
  \if@filesw
    \immediate\write\@auxout{%
      \string\providecommand*\string\newmaxpar[3]{}
    }%
  \fi
}
⟨/body⟩
```

References

[BB24] Javier Bezos López and Johannes L. Braams. *babel — Multilingual support for LaTeX, LuaLaTeX, XeLaTeX and Plain TeX*. Version 24.1. This package manages culturally-determined typographical (and other) rules for a wide range of languages. A document may select a single language to be supported, or it may select several, in which case the document may switch from one language to another in a variety of ways. Babel uses contributed configuration files that provide the detail of what has to be done for each language, as well as `.ini` files for about 300 languages from around the World, including many written in non-Latin and RTL scripts. Many of them work with pdfLaTeX, as well as with XeLaTeX and LuaLaTeX, out of the box. A few even work with plain formats. Jan. 7, 2024. URL: https://ctan.org/pkg/babel (visited on 01/08/2024).

[Koh20a] Markus Kohm. *KOMA-Script. Eine Sammlung von Klassen und Paketen für LaTeX 2ε*. 7. Aufl. Edition DANTE. Print-Ausgabe. Berlin: Lehmanns Media, 2020. ISBN: 978-3-96543-097-6.

[Koh20b] Markus Kohm. *KOMA-Script. Eine Sammlung von Klassen und Paketen für LaTeX 2ε*. 7. Aufl. Edition DANTE. eBook-Ausgabe. Berlin: Lehmanns Media, 2020. ISBN: 978-3-96543-103-4.

[Koh23a] Markus Kohm. *KOMA-Script. Die Anleitung*. 16. Juni 2023. URL: http://mirrors.ctan.org/macros/latex/contrib/koma-script/scrguide-de.pdf (besucht am 04. 07. 2023).

[Koh23b] Markus Kohm. *KOMA-Script. The Guide*. June 16, 2023. URL: http://mirrors.ctan.org/macros/latex/contrib/koma-script/scrguide-en.pdf (visited on 07/14/2023).

[Koh23c] Markus Kohm. *KOMA-Script — A bundle of versatile classes and packages*. Version 3.41. The KOMA-Script bundle provides replacements for the article, report, and book classes with emphasis on typography and versatility. There is also a letter class. July 7, 2023. URL: https://ctan.org/pkg/koma-script (visited on 07/14/2023).

[MFP21] Frank Mittelbach, Robin Fairbairns, and H. Partl. *parskip — Layout with zero `\parindent`, non-zero `\parskip`*. Version 2.0h. Simply changing `\parskip` and `\parindent` leaves a layout that is untidy; this package (though it is no substitute for a properly-designed class) helps alleviate this untidiness. Mar. 14, 2021. URL: https://www.ctan.org/pkg/parskip (visited on 01/19/2024).

[Mit21] Frank Mittelbach. *The parskip package*. Mar. 14, 2021. URL: http://mirrors.ctan.org/macros/latex/contrib/parskip/parskip.pdf (visited on 01/19/2024).

Change History

scrjura-v0.5b – 2010/04/05

`juratitlepagebreak`: new 22

scrjura-v0.5c – 2010/04/26

`\contract@everypar`:
`\contract@Clauseformat` expands

- `markboth`: deprecated 23
- `parnumber`: usage of renewed interface with `\FamilyKeyState` 22
- `ref`: usage of renewed interface with `\FamilyKeyState` 24

scrjura-v0.7a – 2014/01/28
- `\contract@sentence`:
  - `\nobreak\hskip\z@` added to allow hyphenation of the first word after the sentence mark 44
  - using `\textsuperscript` 44

scrjura-v0.7b – 2014/11/03
- `\contract@@usetype`: `\jura@@usetype` renamed 38
- `\contract@afterheading`:
  - `\jura@afterheading` renamed 35
  - no argument 35
- `\contract@env@type`: `\jura@env@type` renamed 27
- `\contract@everypar`: more flexible replacement of `contractAbsoluteClause` 38
- `\contract@paragraph`: more flexible ⟨*environment*⟩`AbsoluteClause` 32
- `\contract@separator`: `\jura@separator` renamed 38
- `\contract@used@everypar`: `\jura@everypar` renamed 36
- `\contract@usetype`: `\jura@usetype` renamed 38
- `\ellipsispar`: `\thecontractAbsoluteClause` made more flexible 27

scrjura-v0.7b – 2014/11/10
- `\getmaxpar`: added fallback code to `aux`-file 47

scrjura-v0.7b – 2014/11/11
- General: General renaming of “Paragraph” into “Clause” 20

scrjura-v0.9 – 2014/11/04
- `\@defjuraenvironment`: new 29

scrjura-v0.9 – 2014/11/12
- `\DeclareNewJuraEnvironment`: new 29

scrjura-v0.9a – 2015/03/09
- `clausemark`: internal value storage 23
- `juratocindent`: internal value storage 21
- `juratotoc`: internal value storage 21
- `parnumber`: internal value storage 22
- `ref`: internal value storage 24

scrjura-v0.9b – 2015/05/01
- `contractSubClause`: must use counter `SubClause` instead of `contractSubClause` 28
- `\thecontractClause`: must use counter `Clause` instead of `contractClause` 28

scrjura-v0.9c – 2015/05/13
- `\@defjuraenvironment`: defining `\the...Clause` 30
- `\contract@paragraph`: `\thecontractClause` replaced by environment-dependent macro 33

scrjura-v0.9e – 2015/11/03
- `\@gobble@IncludeInRelease`: used temporary 36
- `\EndIncludeInRelease`: used temporary 36
- `\IncludeInRelease`: used temporary 36

scrjura-v0.9e – 2015/11/04
- `\p@sentence`: first start the paragraph then print the number 45

scrjura-v0.9f – 2016/02/06
- `\contract@everypar`: `\nobreakspace` replaced by `\parformatseparation` 38
- `\parformatseparation`: new 38

scrjura-v0.9f – 2016/02/24
- `\@defjuraenvironment`: `\protect\\c_atsign_strnameuse` instead of `\csname...\endcsname` 30
- `\contract@paragraph`: `\protect\\c_atsign_strnameuse` instead of `\csname...\endcsname` 33

scrjura-v0.9g – 2015/03/25
- `\getmaxpar`: write to `aux`-file `\if@filesw` 47

scrjura-v0.9g – 2016/03/25
- `\contract@everypar`: writing to `aux`-file `\if@filesw` 38
- `\ellipsispar`: write to `aux`-file only `\if@filesw` 27

scrjura-v0.9h – 2016/04/11
- `clausemark`: `paragraphmark` renamed to `clausemark` 23

scrjura-v0.9h – 2016/04/12
- `\DeclareNewJuraEnvironment`: `\\c_atsign_strifnextchar` replaced by `\kernel@ifnextchar` 29
- `\ref@ParN`: `\\c_atsign_strifnextchar` replaced by `\kernel@ifnextchar` 43
  - `\\c_atsign_strifstar` replaced by `\kernel@ifstar` 43
- `\refClause`: `\\c_atsign_strifstar` replaced by `\kernel@ifstar` 40
- `\refClauseN`: `\\c_atsign_strifstar` replaced by `\kernel@ifstar` 41

- `\theHsentence`: Ulrike Fischer requested to use `\def` instead of `\newcommand*` . . . 44
- `\theHSubClause`: Ulrike Fischer requested to use `\def` instead of `\newcommand*` . . . 32

v0.0.1 – 2023-10-10

- General: new KOMA-Script spin-off . . . 1
- `\contract@@usetype`: `\scrjura@@usetype` renamed . . . 38
- `\contract@afterheading`: `\scrjura@afterheading` renamed . . . 35
- `\contract@doendpe`: `\scrjura@doendpe` renamed . . . 36
- `\contract@env@type`: `\scrjura@env@type` renamed . . . 27
- `\contract@lang@error`: `\scrjura@lang@error` renamed . . . 46
- `\contract@separator`: `\scrjura@separator` renamed . . . 38
- `\contract@special@reset@par`: `\scrjura@special@par` renamed . . . 39
  - `\scrjura@special@reset@par` renamed . . . 39
- `\contract@used@everypar`: `\scrjura@everypar` renamed . . . 36
- `\contract@usetype`: `\scrjura@usetype` renamed . . . 38
- `\if@contract@skiphyperref`: `\if@scrjura@skiphyperref` renamed . . . 28

v0.0.2 – 2024/01/23

- `\contract@paragraph`: the argument is optional . . . 32
- `\contract@subparagraph`: the argument is optional . . . 34

v0.9 – 2024-02-02

- General: first release as standalone package . . . 1

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.