

SCONTENTS

Stores \LaTeX CONTENTS

V2.3 2025-04-23*

©2019–2025 by Pablo González†

CTAN: <https://www.ctan.org/pkg/scontents>

 <https://github.com/pablgonz/scontents>

Abstract

This package allows to store \LaTeX code, including “*verbatim*”, in $\langle sequences \rangle$ using the `l3seq` module of `expl3`. The $\langle stored\ content \rangle$ can be used as many times as desired in the document, additionally you can write to $\langle external\ files \rangle$ or show it in $\langle verbatim\ style \rangle$.

Contents

1	Description of the package	1	6	Other commands provided	7
2	Motivation and Acknowledgments	1	6.1	The command <code>\meaningsc</code>	7
3	License and Requirements	2	6.2	The command <code>\countsc</code>	7
4	The scontents package	2	6.3	The command <code>\cleanseqsc</code>	7
4.1	Installation	2	7	The scontents package in action	8
4.2	Loading and options	2	8	Examples	8
4.3	The TAB character	2	8.1	From answers package	8
4.4	Configuration of the options	3	8.2	From filecontentsdef package	9
4.5	Options Overview	3	8.3	From TeX-SX	9
5	User interface	3	8.4	Customization of <code>verbatimsc</code>	11
5.1	The environment <code>scontents</code>	4	8.5	The command <code>\mergesc</code> in action	13
5.2	The command <code>\newenvsc</code>	5	8.6	The tagged PDF example	15
5.3	The command <code>\Scontents</code>	5	9	Change history	16
5.4	The command <code>\getstored</code>	6	10	Index of Documentation	17
5.5	The command <code>\foreachsc</code>	6	11	References	17
5.6	The command <code>\tpestored</code>	6	12	Implementation	18
5.7	The command <code>\mergesc</code>	6	13	Index of Implementation	46
5.8	The environment <code>verbatimsc</code>	7			

1 Description of the package

The `SCONTENTS` package allows to $\langle store\ contents \rangle$ in $\langle sequences \rangle$ or $\langle external\ files \rangle$. In some ways it is similar to the `filecontentsdef` package, with the difference in which the $\langle content \rangle$ is stored. The idea behind this package is to get an approach to ConTeXt “*buffers*” by making use $\langle sequences \rangle$.

2 Motivation and Acknowledgments

In \LaTeX there is no direct way to record content for later use, although you can do this using `\macros`, recording $\langle verbatim\ content \rangle$ is a problem, usually you can avoid this by creating external files or boxes.

The general idea of this package is to try to imitate this implementation “*buffers*” that has ConTeXt which allows you to save content in memory, including *verbatim*, to be used later. The package `filecontentsdef` solves this problem and since `expl3` has an excellent way to manage data, ideas were combined giving rise to this package.

This package would not be possible without the great work of JEAN FRANÇOIS BURNOL who was kind enough to take my requirements into account and add the `filecontentsdefmacro` environment. Also a special thanks to Phelype Oleinik who has collaborated and adapted a large part of the code and all \LaTeX team for their great work and to the different members of the TeX-SX community who have provided great answers and ideas. Here a note of the main ones:

1. Stack datastructure using LaTeX
2. LaTeX equivalent of ConTeXt buffers
3. Storing an array of strings in a command
4. Collecting contents of environment and store them for later retrieval
5. Collect contents of an environment (that contains *verbatim* content)

- Starting with version 2.3 the `SCONTENTS` package is fully compatible with *tagged* PDF and will have \LaTeX release 2024-11-01 (TeX Live 2024) as a minimum requirement.

*This file describes a documentation for v2.3, last revised 2025-04-23.

†E-mail: pablgonz@educarchile.cl.

3 License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (lpl), version 1.3 or later (<https://www.latex-project.org/lpl.txt>). The software has the status “maintained”.

The The `SCONTENTS` package is written (mostly) using `expl3`, it requires an updated version of \LaTeX to work (minimum required to \LaTeX release 2024-11-01). This package can be used with `plain`, `context`, `xelatex`, `lualatex`, `pdflatex` and the classical workflow `latex»dvips»ps2pdf`.

4 The scontents package

4.1 Installation

The package `SCONTENTS` is present in \TeX Live and \MiKTeX , use the package manager to install. For manual installation, download [scontents.zip](#) and unzip it, run `luatex scontents.ins` and move all files to appropriate locations, then run `mktexlsr`. To produce the documentation with source code run `luatex scontents.ins` and `lualatex scontents.dtx` three times.

<code>scontents.tex</code>	»	<code>TDS:tex/generic/scontents/</code>
<code>scontents-code.tex</code>	»	<code>TDS:tex/generic/scontents/</code>
<code>scontents.sty</code>	»	<code>TDS:tex/latex/scontents/</code>
<code>t-scontents.mkiv</code>	»	<code>TDS:tex/context/third/scontents/</code>
<code>scontents.pdf</code>	»	<code>TDS:doc/latex/scontents/</code>
<code>README.md</code>	»	<code>TDS:doc/latex/scontents/</code>
<code>scontents.dtx</code>	»	<code>TDS:source/latex/scontents/</code>
<code>scontents.ins</code>	»	<code>TDS:source/latex/scontents/</code>

4.2 Loading and options

The package is loaded in the usual way:

For \LaTeX users

```
\usepackage{scontents}
```

or

```
\usepackage[⟨key = val⟩]{scontents}
```

The package options are not available for plain \TeX and \ConTeXt , see 4.4.

For plain \TeX users

```
\input scontents.tex
```

For \ConTeXt users

```
\usemodule{scontents}
```

🔗 \ConTeXt users should use `-luatex`, the implementation does not support `LuaMetaTeX`.

4.3 The TAB character

Some users use horizontal TABs “`⇥`” from keyboard to indented the source code of the document and depending on the text editor used, some will use real TABs (“*hard tabs*”), others with “*soft tabs*” (`␣` or `␣␣␣`) or both.

At first glance it may seem the same, but the way in which TABs (“*hard tabs*”) are processed according to the context in which they are found within a file, both in $\langle reading \rangle$ ¹ and $\langle writing \rangle$ ² are different and may have adverse consequences.

In a standard \LaTeX document, the character TAB “`⇥`” are treated as explicit spaces (in most contexts) and is the behavior when $\langle stored contents \rangle$, but when $\langle writing files \rangle$ these are preserved.

With a \TeX Live distribution, the TAB character is “*printable*” for `latex`, `pdflatex` and `lualatex`, but if you use `xelatex` you must add the `-8bit` option on the command line, otherwise you will get \TeX -TAB (`^^I`) in the $\langle output file \rangle$.

As a general recommendation “Do not use TAB character unless strictly necessary”, for example within

¹Check the answer given by Ulrich Diez in [Keyboard TAB character in argument v \(xparse\)](#).

²Check the answer given by Enrico Gregorio in [How to output a tabulation into a file](#).

a *verbatim* environment that supports this character such as `Verbatim` of the package `fancyvrb` or `lstlisting` of the package `listings` or when you want to generate a `MakeFile` file.

4.4 Configuration of the options

Most of the options can be passed directly to the package or using the command `\setupsc`. All boolean keys can be passed using the equal sign “=” or just the name of the key, all unknown keys will return an error. In this section are described some of the options, a summary of all options is shown in section 4.5.

`\setupsc` `\setupsc{⟨keyval list⟩}`

The command `\setupsc` sets the `⟨keys⟩` in a global way, it can be used both in the preamble and in the body of the document as many times as desired. However, options set in the declaration of an environment (with `\newenvsc`) have precedence over options set with `\setupsc`.

Options in the optional arguments of environments and commands have the highest precedence, overriding both options in `\newenvsc`, and in `\setupsc`.

`verb-font = {⟨font family⟩}` default: `\ttfamily`

Sets the `⟨font family⟩` used to display the `⟨stored content⟩` for the `\typestored` and `\meaningsc` commands. This key is only available as a package option or using `\setupsc`.

`store-all = {⟨seq name⟩}` default: `not used`

It is a `⟨meta-key⟩` that sets the `store-env` key of the `scontents` environment and the `store-cmd` key of the `\Scontents` command. This key is only available as a package option or using `\setupsc`.

`overwrite = {⟨true | false⟩}` default: `false`

Sets whether the `⟨files⟩` generated by `write-out` and `write-env` from the `scontents` environment will be rewritten. This key is available as a package option, for `\setupsc`, for `\Scontents*` and for the environment `scontents`.

`print-all = {⟨true | false⟩}` default: `false`

It is a `⟨meta-key⟩` that sets the `print-env` key of the `scontents` environment and the `print-cmd` key of the `\Scontents` command. This key is only available as a package option or using `\setupsc`.

`force-eol = {⟨true | false⟩}` default: `false`

Sets if the end of line for the `⟨stored content⟩` is hidden or not. This key is necessary only if the last line is the closing of some environment defined by the `fancyvrb` package as `\end{Verbatim}` or another environment that does not support a comments “%” after closing `\end{⟨env⟩}%`. This key is available for the `scontents` environment and the `\Scontents` command.

`width-tab = {⟨integer⟩}` default: `1`

Sets the equivalence in `⟨spaces⟩` for the character `TAB` used when displaying stored content in *verbatim style*. The value must be a `⟨positive integer⟩`. This key is available for the `\typestored` and the `\meaningsc` commands.

4.5 Options Overview

Summary of available options:

key	package	<code>\setupsc</code>	<code>scontents</code>	<code>\Scontents</code>	<code>\Scontents*</code>	<code>\typestored</code>	<code>\meaningsc</code>
<code>store-env</code>	✓	✓	✓	✗	✗	✗	✗
<code>store-cmd</code>	✓	✓	✗	✓	✓	✗	✗
<code>print-env</code>	✓	✓	✓	✗	✗	✗	✗
<code>print-cmd</code>	✓	✓	✗	✓	✓	✗	✗
<code>print-all</code>	✓	✓	✗	✗	✗	✗	✗
<code>store-all</code>	✓	✓	✗	✗	✗	✗	✗
<code>write-env</code>	✗	✗	✓	✗	✗	✗	✗
<code>write-cmd</code>	✗	✗	✗	✗	✓	✗	✗
<code>write-out</code>	✗	✗	✓	✗	✓	✗	✗
<code>overwrite</code>	✓	✓	✓	✗	✓	✗	✗
<code>width-tab</code>	✓	✓	✗	✗	✗	✓	✓
<code>force-eol</code>	✓	✓	✓	✓	✓	✗	✗
<code>verb-font</code>	✓	✓	✗	✗	✗	✗	✗

5 User interface

The user interface consists in `scontents` environment, `\Scontents` and `\Scontents*` commands to `⟨stored content⟩` and `\getstored` command to get the `⟨stored content⟩` along with other utilities described in this documentation.

5.1 The environment scontents

```
scontents \begin{scontents}[(keyval list)]
          <env contents>
\end{scontents}
```

The `scontents` environment allows you to *store* and *write* content, including *verbatim* material. After the package has been loaded, the environment can be used both in the preamble and in the body of the document.

For the correct operation `\begin{scontents}` and `\end{scontents}` must be in different lines, all *keys* must be passed separated by commas and “without separation” of the start of the environment.

Comments “%” or “any character” after `\begin{scontents}` or `[(keyval list)]` on the same line are not supported, the package will return an “error” message if this happens. In a similar way comments “%” or “any character” after `\end{scontents}` on the same line the package will return a “warning” message.

The environment can be *nested* if it is properly balanced and does not appear “literally” in commented lines or in some *verbatim* environment or command. As an example:

```
\begin{scontents}[store-env=outer]
This text is in the outer environment (before nested).
\begin{scontents}[store-env=inner]
This text is found in the inner environment (inside of nested).
\end{scontents}
This text is in the outer environment (after nested).
\end{scontents}
```

Of course, content stored in the *inner* sequence is only available after content stored in the *outer* sequence one has been retrieved, either by using the key `print-env` or `getstored` command.

It is advisable to store content within sequences with different names, so as not to get lost in the order in which content is stored.

Notes for plain T_EX and ConT_EXt users

In plain T_EX there is not environments as in L^AT_EX. Instead of using the environment `scontents`, one should use a *pseudo environment* delimited by `\scontents` and `\endscontents`.

```
\scontents \scontents[(keyval list)]
\endscontents <env contents>
\endscontents
```

ConT_EXt users should use `\startcontents` and `\stopcontents`.

```
\startcontents \startcontents[(keyval list)]
\stopcontents <env contents>
\stopcontents
```

Options for environment

The environment options can be configured globally using option in package or the `\setupsc` command and locally using `[(key = val)]` in the environment. The key `force-eol` is available for this environment.

```
store-env = {<seq name>} default: contents
```

Sets the name of the *sequence* in which the contents will be stored. If the sequence does not exist, it will be created globally.

```
print-env = {<true | false>} default: false
```

Sets if the *stored content* is displayed or not at the time of running the environment. The content is extracted from the *sequence* in which it is stored.

```
write-env = {<file.ext>} default: not used
```

Sets the name of the *external file* in which the *contents* of the environment will be written. The *file.ext* will be created in the working directory, relative or absolute paths are not supported. If *file.ext* does not exist, it will be created or overwritten if the `overwrite` key is used.

The characters TABs will be written in *file.ext* and the *contents* will be stored in the *sequence* established at that time. X_YL^AT_EX users using the TAB character must add `-8bit` at the command line, otherwise you will get T_EX-TAB (^^AI) in *file.ext*.

```
write-out = {<file.ext>} default: not used
```

Sets the name of the *external file* in which the *contents* of the environment will be written. The *file.ext* will be created in the working directory, relative or absolute paths are not supported. If *file.ext* does not exist, it will be created or overwritten if the `overwrite` key is used.

The characters TABs will be written in $\langle file.ext \rangle$, the rest of the $\langle keys \rangle$ will not be available and the $\langle contents \rangle$ will NOT be stored in any $\langle sequence \rangle$. X_YTeX users using the TAB character must add `-8bit` at the command line, otherwise you will get T_EX-TAB (\^I) in $\langle file.ext \rangle$.

5.2 The command `\newenvsc`

```
\newenvsc \newenvsc{ $\langle env name \rangle$ }[ $\langle initial keys \rangle$ ]
```

The command `\newenvsc` allows you to create $\langle new environments \rangle$ based on the same characteristics of the `scontents` environment. The values entered in [$\langle initial keys \rangle$] will be considered as the default values for this new environment and the valid $\langle keys \rangle$ are `store-env` and `print-env`. For example:

```
\newenvsc{myenvstore}[store-env=myseq,print-env=false]
```

created the `myenvstore` environment that stored the content in the `myseq` sequence and will not display the content when it is executed.

5.3 The command `\Scontents`

```
\Scontents \Scontents[ $\langle key = val \rangle$ ]{ $\langle argument \rangle$ }
\Scontents* [  $\langle key = val \rangle$  ] {  $\langle argument \rangle$  }
\Scontents* [  $\langle key = val \rangle$  ]  $\langle del \rangle$   $\langle argument \rangle$   $\langle del \rangle$ 
```

The `\Scontents` command reads the $\langle argument \rangle$ in standard mode. It is not possible to pass environments such as *verbatim*, but it is possible to use the implementation of `\Verb` provided by the `fvextra` package for contents on one line and `\lstinline` from `listings` package, but it is preferable to use the starred (*) version.

The `\Scontents*` command reads the $\langle argument \rangle$ under *verbatim* category code regimen. If its first delimiter is a brace, it will be assumed that the $\langle argument \rangle$ is nested into braces. Otherwise it will be assumed that the ending of that $\langle argument \rangle$ is delimited by that first delimiter $\langle del \rangle$ like command `\verb`.

Blank lines are preserved, escaped braces “\{” and “\}” must also be balanced if the argument is used with braces and TABs characters typed from the keyboard are converted into spaces. The starred argument (*) and [$\langle key = val \rangle$] must not be separated by horizontal spaces between them and the command.

Both versions can be used anywhere in the document and cannot be used as an $\langle argument \rangle$ for other command.

Options for command

The command options can be configured globally using option in package or the `\setupsc` command and locally using [$\langle key = val \rangle$]. The key `force-eol` is available for this command.

```
store-cmd = {  $\langle seq name \rangle$  } default: contents
```

Sets the name of the $\langle sequence \rangle$ in which the contents will be stored. If the sequence does not exist, it will be created globally.

```
print-cmd = {  $\langle true \mid false \rangle$  } default: false
```

Sets if the $\langle stored content \rangle$ is displayed or not at the time of running the command. The content is extracted from the $\langle sequence \rangle$ in which it is stored.

Options only for the starred version

```
write-cmd = {  $\langle file.ext \rangle$  } default: not used
```

Sets the name of the $\langle external file \rangle$ in which the $\langle contents \rangle$ of the $\langle argument \rangle$ will be written. The $\langle file.ext \rangle$ will be created in the working directory, relative or absolute paths are not supported. If $\langle file.ext \rangle$ does not exist, it will be created or overwritten if the `overwrite` key is used.

The characters TABs will be written in $\langle file.ext \rangle$ and the $\langle contents \rangle$ will be stored in the $\langle sequence \rangle$ established at that time. X_YTeX users using the TAB character must add `-8bit` at the command line, otherwise you will get T_EX-TAB (\^I) in $\langle file.ext \rangle$.

```
write-out = {  $\langle file.ext \rangle$  } default: not used
```

Sets the name of the $\langle external file \rangle$ in which the $\langle contents \rangle$ of the $\langle argument \rangle$ will be written. The $\langle file.ext \rangle$ will be created in the working directory, relative or absolute paths are not supported. If $\langle file.ext \rangle$ does not exist, it will be created or overwritten if the `overwrite` key is used.

The characters TABs will be written in $\langle file.ext \rangle$, the rest of the $\langle keys \rangle$ will not be available and the $\langle contents \rangle$ will NOT be stored in any $\langle sequence \rangle$. X_YTeX users using the TAB character must add `-8bit` at the command line, otherwise you will get T_EX-TAB (\^I) in $\langle file.ext \rangle$.

The key `overwrite` is available for this command.

5.4 The command `\getstored`

```
\getstored <index>[<seq name>]
```

The command `\getstored` gets the content stored in `{<seq name>}` according to the `<index>` in which it was stored. The command is robust and can be used as an `<argument>` for another command. If the optional argument is not passed, the default value is the “last element” stored in `{<seq name>}`.

5.5 The command `\foreachsc`

```
\foreachsc <key = val>[<seq name>]
```

The command `\foreachsc` goes through and executes the command `\getstored` on the contents stored in `{<seq name>}`. If you pass without options run `\getstored` on all contents stored in `{<seq name>}`.

Options for command

- `sep = {<code>}` default: *empty*
 Establishes the separation between each content stored in `{<seq name>}`. For example, you can use `sep={\ [10pt]}` for vertical separation of stored contents.
- `step = {<integer>}` default: *1*
 Sets the increment (`<step>`) applied to the value set by key `start` for each element stored in the `{<seq name>}`. The value must be a `<positive integer>`.
- `start = {<integer>}` default: *1*
 Sets the `<index>` number of the `{<seq name>}` from which execution will start. The value must be a `<positive integer>`.
- `stop = {<integer>}` default: *total*
 Sets the `<index>` number of the `{<seq name>}` from which execution it will finish executing. The value must be a `<positive integer>`.
- `before = {<code>}` default: *empty*
 Sets the `{<code>}` that will be executed `<before>` each content stored in `{<seq name>}`. The `{<code>}` must be passed between braces.
- `after = {<code>}` default: *empty*
 Sets the `{<code>}` that will be executed `<after>` each content stored in `{<seq name>}`. The `{<code>}` must be passed between braces.
- `wrapper = {<code> {#1} more code}` default: *empty*
 Wraps the content stored in `{<seq name>}` referenced by `{#1}`. The `{<code>}` must be passed between braces. For example `\foreachsc[wrapper={\makebox[1em][l]{#1}}]{contents}`.

5.6 The command `\tpestored`

```
\tpestored <index, 1-end, width-tab = number>[<seq name>]
```

The command `\tpestored` internally places the content stored in the `{<seq name>}` into the `verbatimsc` environment. The `<index>` corresponds to the position in which the content is stored in the `{<seq name>}`, if `<1-end>` is used “all” content stored in `{<seq name>}` will be printed.

If the optional argument is not passed it defaults to the first element stored in the `{<seq name>}`. The key `width-tab` is available for this command.

5.7 The command `\mergesc`

```
\mergesc <typestored | meanings, keys>[<seq A>[<index>], <seq B>[<start - stop>], <seq C>[<1-end>]]
```

The command `\mergesc` internally assembles the content stored in the `{<seq A>}[1]`, `{<seq B>}[2-5]` and `{<seq C>}[1-end]` into a temporary internal `<seq temp>`.

The use of the keys `typestored` or `meaningsc` are “mandatory” and disjoint from each other, the rest of the accepted `<keys>` are `print-cmd`, `write-out`, `width-tab` and `overwrite`.

The use of the `write-out` key with this command follows the same rules already described, the main advantage is that it allows to join stored content *without rewriting* the file over and over again, by design \TeX does not have an append mode for writing files, this effectively allows you to write chunks of code and then merge them into a single file.

5.8 The environment `verbatimsc`

`verbatimsc` Internal environment used by `\typestored` to display *verbatim style* contents.

One consideration to keep in mind is that this is a “*representation*” of the *stored content* in a “*verbatim*” environment.

The internal `verbatimsc` environment is compatible with *tagged* PDF and can be customized in the following ways after loading the `SCONTENTS` package:

Using the package `fancyvrb`:

```
\makeatletter
\let\verbatimsc\undefined
\let\endverbatimsc\undefined
\makeatother
\usepackage{fancyvrb}
\DefineVerbatimEnvironment{verbatimsc}{Verbatim}{numbers=left}
```

Using the package `minted`:

```
\makeatletter
\let\verbatimsc\undefined
\let\endverbatimsc\undefined
\makeatother
\usepackage{minted}
\newminted{tex}{linenos}
\newenvironment{verbatimsc}{\VerbatimEnvironment\begin{texcode}}{\end{texcode}}
```

Using the package `listings`:

```
\makeatletter
\let\verbatimsc\undefined
\let\endverbatimsc\undefined
\makeatother
\usepackage{listings}
\lstnewenvironment{verbatimsc}
{
  \lstset{
    basicstyle=\small\ttfamily,
    columns=fullflexible,
    language=[LaTeX]TeX,
    numbers=left,
    numberstyle=\tiny\color{gray},
    keywordstyle=\color{red}
  }
}{}
}
```

At the moment, the `fancyhdr`, `listings`, and `minted` packages are not compatible with *tagged* PDF.

6 Other commands provided

6.1 The command `\meaningsc`

`\meaningsc` `\meaningsc` [*index*, *width-tab = number*] {*seq name*}

The command `\meaningsc` executes `\meaning` on the content stored in {*seq name*}. The *index* corresponds to the position in which the content is stored in the {*seq name*}.

If the optional argument is not passed it defaults to the first element stored in the {*seq name*}. The key *width-tab* is available for this command.

6.2 The command `\countsc`

`\countsc` `\countsc` {*seq name*}

The command `\countsc` count a number of contents stored in {*seq name*}.

6.3 The command `\cleanseqsc`

`\cleanseqsc` `\cleanseqsc` {*seq name*}

The command `\cleanseqsc` remove all contents stored in {*seq name*}.

7 The `SCONTENTS` package in action

Remember the abstract on the first page?, this is it:

Abstract

This package allows to store \LaTeX code, including “*verbatim*”, in `(sequences)` using the `l3seq` module of `expl3`. The `(stored content)` can be used as many times as desired in the document, additionally you can write to `(external files)` or show it in `(verbatim style)`.

And the description of the package?

The `SCONTENTS` package allows to `(store contents)` in `(sequences)` or `(external files)`. In some ways it is similar to the `filecontentsdef` package, with the difference in which the `(content)` is stored. The idea behind this package is to get an approach to ConTeXt “*buffers*” by making use `(sequences)`.

I’ve only written:

```
\begin{abstract}
This package allows to store \hologo{LaTeX} code, including \enquote{\emph{verbatim}},
in \mymeta{sequences} using the \mypkg{l3seq} module of \mypkg{expl3}. The \mymeta{stored
content} can be used as many times as desired in the document, additionally you can write
to \mymeta{external files} or show it in \mymeta{verbatim style}.
\end{abstract}
```

and

The `\mypkg*{scontents}` package allows to `\mymeta{store contents}` in `\mymeta{sequences}` or `\mymeta{external files}`. In some ways it is similar to the `\mypkg{filecontentsdef}` package, with the difference in which the `\mymeta{content}` is stored. The idea behind this package is to get an approach to `\hologo{ConTeXt}` `\enquote{\emph{buffers}}` by making use `\mymeta{sequences}`.

Of course, I didn’t copy and paste. The real code they were written with is:

```
1 \begin{scontents}[store-env=abstract,print-env=true]
2 \begin{abstract}
3 This package allows to store \hologo{LaTeX} code, including \enquote{\emph{verbatim}},
4 in \mymeta{sequences} using the \mypkg{l3seq} module of \mypkg{expl3}. The \mymeta{stored
5 content} can be used as many times as desired in the document, additionally you can write
6 to \mymeta{external files} or show it in \mymeta{verbatim style}.
7 \end{abstract}
8 \end{scontents}
```

and

```
1 \begin{scontents}[store-env=description, print-env=true]
2 The \mypkg*{scontents} package allows to \mymeta{store contents} in \mymeta{sequences}
3 or \mymeta{external files}. In some ways it is similar to the \mypkg{filecontentsdef}
4 package, with the difference in which the \mymeta{content} is stored. The idea behind
5 this package is to get an approach to \hologo{ConTeXt} \enquote{\emph{buffers}} by
6 making use \mymeta{sequences}.
7 \end{scontents}
```

I stored the content in memory and then ran `\getstored` and `\tpestored`. This is one of the ways you can use `SCONTENTS`.

8 Examples


These are some adapted examples that have served as inspiration for the creation of this package. The examples are attached to this documentation and can be extracted from your PDF viewer or from the command line by running:

```
$ pdftdetach -saveall scontents.pdf
```

and then you can use the excellent `arara`³ tool to compile them.

8.1 From answers package

Example 1

Adaptation of example 1 of the package `answers` .

³The cool \TeX automation tool: <https://www.ctan.org/pkg/arara>



```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage[store-cmd=solutions]{scontents}
5 \newtheorem{ex}{Exercise}
6 \setlength{\parindent}{0pt}
7 \pagestyle{empty}
8 \begin{document}
9 \section{Problems}
10 \begin{ex}
11 First exercise
12 \Scontents{First solution.}
13 \end{ex}
14
15 \begin{ex}
16 Second exercise
17 \Scontents{Second solution.}
18 \end{ex}
19
20 \section{Solutions}
21 \foreachsc[sep={\\[10pt]}]{solutions}
22 \end{document}

```

8.2 From filecontentsdef package

Example 2

Adaptation of example from package filecontentsdef .

```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage[store-env=defexercise,store-cmd=defexercise]{scontents}
5 \setlength{\parindent}{0pt}
6 \pagestyle{empty}
7 \begin{document}
8 % not starred
9 \Scontents{
10 Prove that  $x^n+y^n=z^n$  is not solvable in positive integers if  $n$  is at
11 most  $3$ .\par
12 }
13 % starred
14 \Scontents*|Refute the existence of black holes in less than  $140$  characters.|
15 % write environment to \jobname.txt
16 \begin{scontents}[write-env=\jobname.txt]
17 \def\NSA{NSA}%
18 Prove that factorization is easily done via probabilistic algorithms and
19 advance evidence from knowledge of the names of its employees in the
20 seventies that the \NSA\ has known that for  $40$  years.\par
21 \end{scontents}
22 % see all stored
23 \begin{itemize}
24 \foreachsc[before={\item }]{defexercise}
25 \end{itemize}
26 % \getstored are robust :)
27 \section{\getstored[2]{defexercise}}
28 \end{document}

```

8.3 From TeX-SX

Example 3

Adapted from LaTeX equivalent of ConTeXt buffers .

```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage[store-cmd=tikz]{scontents}
5 \usepackage{tikz}
6 \setlength{\parindent}{0pt}
7 \pagestyle{empty}
8 \Scontents{\matrix{ \node (a) {a} ; & \node (b) {b} ; \\ } ;}

```

```

9 \Scontents{\matrix[ampersand replacement=\&]
10 { \node (a) {$a$} ; \& \node (b) {$b$} ; \\ } ;}
11 \Scontents{\matrix{\node (a) {$a$} ; & \node (b) {$b$} ; \\ } ; }
12 \begin{document}
13 \section{tikzpicture}
14 \begin{tikzpicture}
15 \getstored[1]{tikz}
16 \end{tikzpicture}
17
18 \begin{tikzpicture}
19 \getstored[2]{tikz}
20 \end{tikzpicture}
21
22 \begin{tikzpicture}
23 \getstored{tikz}
24 \end{tikzpicture}
25
26 \begin{scontents}[store-env=buffer]
27 Hello World!
28
29 This is a \verb*|fake poor man's buffer :)|.
30 \end{scontents}
31
32 \section{source tikz}
33 \tpestored[1]{tikz}
34 \tpestored[2]{tikz}
35 \tpestored[3]{tikz}
36
37 \section{fake buffer}
38 \subsection{real content}
39 \getstored[1]{buffer}
40 \subsection{verbatim style}
41 \tpestored[1]{buffer}
42 \subsection{meaning}
43 \meaningsc[1]{buffer}
44
45 \section{tikz again}
46 \foreachsc[before={\begin{tikzpicture}},after={\end{tikzpicture}},sep={\[\[10pt]]}{tikz}
47 \end{document}

```

Example 4

Adapted from [Collecting contents of environment and store them for later retrieval](#) .

```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage{scontents}
5 \setlength{\parindent}{0pt}
6 \pagestyle{empty}
7 \begin{document}
8 \begin{scontents}[store-env=main]
9 Something for main A.
10 \end{scontents}
11
12 \begin{scontents}[store-env=main]
13 Something for \verb|main B|.
14 \end{scontents}
15
16 \begin{scontents}[store-env=other]
17 Something for \verb|other|.
18 \end{scontents}
19
20 \textbf{Let's print them}
21
22 This is first stored in main: \getstored[1]{main}\par
23 This is second stored in main: \getstored{main}\par
24 This is stored in other: \getstored{other}
25
26 \textbf{Print all of stored in main}\par
27 \foreachsc[sep={\[\[10pt]]}{main}
28 \end{document}

```

Example 5

Adapted from [Collect contents of an environment \(that contains verbatim content\)](#) .

```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage{scontents}
5 \setlength{\parindent}{0pt}
6 \pagestyle{empty}
7 \begin{document}
8 \section{Problem stated the first time}
9 \begin{scontents}[print-env=true,store-env=problem]
10 This is normal text.
11 \verb|This is from the verb command.|
12 \verb*|This is from the verb* command.|
13 This is normal text.
14 \begin{verbatim}
15 This is from the verbatim environment:
16 &{}~
17 \end{verbatim}
18 \end{scontents}
19 \section{Problem restated}
20 \getstored[1]{problem}
21 \section{Problem restated once more}
22 \getstored[1]{problem}
23 \end{document}

```

Example 6

Adapted from [Environment hiding its content](#) .


```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass[10pt]{article}
4 \usepackage{scontents}
5 \newenvsc{forshort}[store-env=forshort,print-env=false]
6 \setlength{\parindent}{0pt}
7 \pagestyle{empty}
8 \begin{document}
9
10 Something in the whole course.
11
12 \begin{forshort}
13   Just a summary...
14 \end{forshort}
15
16 \end{document}

```

8.4 Customization of verbatimsc

Example 7

Customization of `verbatimsc` using the `fancyvrb` and `tcolorbox` package .

```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage{scontents}
5 \makeatletter
6 \let\verbatimsc\undefined
7 \let\endverbatimsc\undefined
8 \makeatother
9 \usepackage{fvextra}
10 \usepackage{xcolor}
11 \definecolor{mygray}{gray}{0.9}
12 \usepackage{tcolorbox}
13 \newenvironment{verbatimsc}%
14 {\VerbatimEnvironment
15 \begin{tcolorbox}[colback=mygray, boxsep=0pt, arc=0pt, boxrule=0pt]
16 \begin{Verbatim}[fontsize=\scriptsize, breaklines, breakafter=*, breaksymbolsep=0.5em,
17 breakaftersymbolpre={\,\tiny\ensuremath{\lfloor}}}%

```

```

18 {\end{Verbatim}}%
19 \end{tcolorbox}}
20 \setlength{\parindent}{0pt}
21 \pagestyle{empty}
22 \begin{document}
23 \section{Test \texttt{\textbackslash begin\{scontents\}} with \texttt{fancyvrb}}
24 Test \verb+{scontents}+ \par
25
26 \begin{scontents}
27 Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+
28 with index 1.
29
30 Prove new \Verb*{ fancyvrb with braces } and environment \verb+Verbatim*+
31 \begin{verbatim}
32 verbatim environment
33 \end{verbatim}
34 \end{scontents}
35
36 \section{Test \texttt{\textbackslash Scontents} with \texttt{fancyvrb}}
37 \Scontents{ We have coded this in \LaTeX: $E=mc^2$}.
38
39 \section{Test \texttt{\textbackslash getstored}}
40 \getstored[1]{contents}\par
41 \getstored{contents}
42
43 \section{Test \texttt{\textbackslash meaningsc}}
44 \meaningsc[1]{contents}\par
45 \meaningsc[2]{contents}
46
47 \section{Test \texttt{\textbackslash typestored}}
48 \typestored[1]{contents}
49 \typestored[2]{contents}
50 \end{document}

```

Example 8

Customization of `verbatimsc` using the listings package .

```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage{scontents}
5 \makeatletter
6 \let\verbatimsc@undefined
7 \let\endverbatimsc@undefined
8 \makeatother
9 \usepackage{xcolor}
10 \usepackage{listings}
11 \lstnewenvironment{verbatimsc}
12 {
13   \lstset{
14     basicstyle=\small\ttfamily,
15     breaklines=true,
16     columns=fullflexible,
17     language=[LaTeX]TeX,
18     numbers=left,
19     numbersep=1em,
20     numberstyle=\tiny\color{gray},
21     keywordstyle=\color{red}
22   }
23 }{}
24 \setlength{\parindent}{0pt}
25 \pagestyle{empty}
26 \begin{document}
27 \section{Test \texttt{\textbackslash begin\{scontents\}} with \texttt{listings}}
28 Test \verb+{scontents}+ \par
29
30 \begin{scontents}
31 Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+ with index 1.\par
32
33 Prove \lstinline[basicstyle=\ttfamily] | \lstinline | and environment \verb+Verbatim*+
34 \begin{verbatim}

```

```

35   verbatim environment
36 \end{verbatim}
37 \end{scontents}
38
39 \section[Test \texttt{\textbackslash Scontents*} with \texttt{listings}]
40
41 \Scontents*+ We have coded this in \lstinlne[basicstyle=\ttfamily]|\LaTeX: $E=mc^2$|
42 and more.+
43
44 \section[Test \texttt{\textbackslash getstored}]
45 \getstored{contents}\par
46 \getstored[1]{contents}
47
48 \section[Test \texttt{\textbackslash tpestored}]
49 \tpestored[1]{contents}
50 \tpestored[2]{contents}
51 \end{document}

```

Example 9


Customization of `verbatimsc` using the `minted` package .

```

1 % arara: xelatex: {shell: true, options: [-8bit]}
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage{scontents}
5 \makeatletter
6 \let\verbatimsc@undefined
7 \let\endverbatimsc@undefined
8 \makeatother
9 \usepackage{minted}
10 \newminted{tex}{linenos}
11 \newenvironment{verbatimsc}{\VerbatimEnvironment\begin{texcode}}{\end{texcode}}
12 \pagestyle{empty}
13 \setlength{\parindent}{0pt}
14 \begin{document}
15 \section[Test \texttt{\textbackslash begin\{scontents\}} with \texttt{minted}]
16 Test \verb+{scontents}+ \par
17
18 \begin{scontents}[overwrite,write-env=\jobname.tsc,force-eol=true]
19 Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+
20 with index 1.\par
21
22 Prove new \Verb*{ new fvxtra with braces } and environment \verb+Verbatim*+
23 \begin{Verbatim}[obeytabs, showtabs, tab=\rightarrowfill, tabcolor=red]
24 No tab
25     One real tab
26         Two real Tab plus     one tab
27 \end{Verbatim}
28 \end{scontents}
29
30 \section[See \Verb{\jobname.tsc}]
31 Read \Verb{\jobname.tsc} (shows TABS as red arrows):
32 \VerbatimInput[obeytabs, showtabs, tab=\rightarrowfill, tabcolor=red]{\jobname.tsc}
33
34 \section[Test \texttt{\textbackslash Scontents} with \texttt{minted}]
35
36 \Scontents{ We have coded \par this in \LaTeX: $E=mc^2$.)
37
38 \section[Test \texttt{\textbackslash getstored}]
39 \getstored[1]{contents}\par
40 \getstored{contents}
41
42 \section[Test \texttt{\textbackslash tpestored}]
43 \tpestored[1]{contents}
44 \tpestored[2]{contents}
45 \end{document}

```

8.5 The command `\mergesc` in action


The command `\mergesc` in action, adapted from Denis Bitouzé request at <https://github.com/pablgonz/scontents/issues/2> .

```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage{scontents}
5 % Fix part of a MCE that should go before babel's loading
6 \begin{scontents}[store-env=mce]
7 \documentclass[french]{article}
8 \usepackage[T1]{fontenc}
9 \usepackage[utf8]{inputenc}
10 \usepackage{lmodern}
11 \usepackage[a4paper]{geometry}
12 \end{scontents}
13 % Fix part of a MCE that should go after (>=) babel's loading
14 \begin{scontents}[store-env=mce]
15 \usepackage{babel}
16 \begin{document}
17 \end{scontents}
18 % Fix part of a MCE that should go after its body
19 \begin{scontents}[store-env=mce]
20 \end{document}
21 \end{scontents}
22 \begin{document}
23 \section{First answer}
24 % Variable part of a MCE that should added to the fixed preamble, before babel's loading
25 \begin{scontents}[store-env=mce-1]
26 \usepackage{amsmath}
27 \end{scontents}
28 % Variable part of a MCE being the code snippet
29 \begin{scontents}[store-env=mce-1]
30 \begin{align}
31   0 & \neq 1 \\
32   1 & \neq 0
33 \end{align}
34 \end{scontents}
35 \begin{description}
36 \item[Preamble's addition]\leavevmode
37   \typestored[1]{mce-1}
38 \item[Code snippet]\leavevmode
39   \typestored[2]{mce-1}
40 \item[MCE]\leavevmode
41   \mergesc[typestored, print-cmd=true]
42     {
43       {mce}[1], {mce-1}[1], {mce}[2], {mce-1}[2], {mce}[3]
44     }
45 \end{description}
46 \section{Second answer}
47 % Variable part of a MCE that should added to the fixed preamble, before babel's loading
48 \begin{scontents}[store-env=mce-2]
49 \usepackage{amsmath}
50 \end{scontents}
51 % Variable part of a MCE being the code snippet
52 \begin{scontents}[store-env=mce-2]
53 \begin{flalign}
54   0 & \neq 1 \\
55   1 & \neq 0
56 \end{flalign}
57 \end{scontents}
58
59 \begin{description}
60 \item[Preamble's addition]\leavevmode
61   \typestored[1]{mce-2}
62 \item[Code snippet]\leavevmode
63   \typestored[2]{mce-2}
64 \item[MCE]\leavevmode
65   \mergesc[typestored, print-cmd=true, write-out=mce.txt, overwrite=true]
66     {
67       {mce}[1], {mce-2}[1], {mce}[2], {mce-2}[2], {mce}[3]
68     }
69 \end{description}
70 \end{document}

```

8.6 The tagged PDF example

A verification test for the `\typestored` command and the `verbatimsc` environment compatible with tagged PDF .

```
1 % arara: lualatex
2 % arara: clean: { extensions: [ aux, log] }
3 \DocumentMetadata{lang = en-US, pdfversion = 2.0, pdfstandard = ua-2, testphase=latest}
4 \documentclass{article}
5 \usepackage{scontents, unicode-math, hyperref}
6 \hypersetup{pdftitle = {Test scontents package},}
7 \begin{document}
8 % environment
9 \begin{scontents}[print-env=true]
10     First code \verb|\foo|
11
12     And more code \verb|\baz|
13 \end{scontents}
14
15 % \typestored
16 \typestored[1]{contents}
17 \end{document}
```

9 Change history

In this section you will find some (not all) of the changes in `SCONTENTS` development, from the first public implementation using the `filecontentsdef` package to the current version with only `expl3`.

- v2.3 (ctan), 2025-04-23**
 - Adapting the `verbatimsc` environment for *tagged* PDF.
 - Update minimum required to \LaTeX release 2024-11-01.
 - Safer code for replacement `\obeyedline`.
- v2.2 (ctan), 2025-03-26**
 - Fix internal definition for some functions.
 - Replace `\peek_charcode_ignore_spaces:NTF` by `\peek_charcode:NTF`.
 - Set correct code for `\obeyedline` implement in \LaTeX release 2024-06-01.
- v2.1 (ctan), 2024-06-14**
 - Fix `\cleansc` command.
 - Add `\mergesc` command.
 - Fix internal definition for `seq` var.
 - Fix internal code for `\typestored`.
 - Replace `\cs_argument_spec:N` by `\cs_parameter_spec:N`.
 - Detect `l3keys2e` package (obsolete in june 2022 \LaTeX release).
 - Minor adjustments in the documentation.
- v2.0 (ctan), 2022-04-04**
 - Adapting the `verbatimsc` environment (compatibility `verbatim` package).
 - Removed compatibility layer for older \LaTeX releases.
 - Fix loader in plain \TeX and `ConTeXt`.
 - Minor adjustments in the documentation.
- v1.9 (ctan), 2020-01-21**
 - Update and improvements in the internal code.
 - Updating the generic code for I/O verification.
 - Add `write-cmd` and `write-out` keys for `\Scontents*`.
 - Fix `sep` key in `\foreachsc`.
- v1.8 (ctan), 2019-11-18**
 - Add `\newenvsc` command.
 - Fix nested environment in plain \TeX and `ConTeXt`.
 - Modified default value in `\getstored`.
 - Add `overwrite` key to reduce I/O operations.
 - Deleted an unnecessary group in the code.
- v1.7 (ctan), 2019-10-29**
 - The `verbatimsc` environment was rewritten.
 - Minor adjustments in documentation.
- v1.6 (ctan), 2019-10-26**
 - The internal behavior of `\getstored` has been modified.
 - The internal behavior of `\foreachsc` has been modified.
 - Corrected file extension for `ConTeXt`.
 - Remove spurious warning.
- v1.5 (ctan), 2019-10-24**
 - Add support for plain \TeX and `ConTeXt`.
 - Split internal code for optimization.
 - Add support for vertical spaces in `key=val`.
 - Add `\foreachsc` command.
 - Check if `verbatim` package is loaded.
- v1.4 (ctan), 2019-10-03**
 - Add `store-all` key.
 - Messages and keys were separated.
 - Restructuring of documentation.
 - Now the version of `expl3` is checked instead of `xparse`.
 - The internal behavior of `force-eol` has been modified.
- v1.3 (ctan), 2019-09-24**
 - The environment can now nest.
 - Added `force-eol`, `verb-font` and `width-tab` keys.
 - The extra space has been removed when you run `\getstored`.
 - Internal code has been rewritten more efficiently.
 - Remove starred argument for `\typestored`.
 - Remove `filecontentsdef` dependency.
 - Changing `\regex_replace_all:` for `\tl_replace_all:`.
- v1.2 (ctan), 2019-08-28**
 - Restructuring of documentation.
 - Added copy of `\tex_scantokens:`.
- v1.1 (ctan), 2019-08-12**
 - Extension of documentation.
 - Replace `\tex_endinput:D` by `\file_input_stop:`.
- v1.0 (ctan), 2019-07-30**
 - First public release.

10 Index of Documentation

The italic numbers denote the pages where the corresponding entry is described.

C	
Commands provide by SCONTENTS :	
<code>\Scontents*</code>	3, 5
<code>\Scontents</code>	3, 5
<code>\cleanseqsc</code>	7
<code>\countsc</code>	7
<code>\endscontents</code>	4
<code>\foreachsc</code>	6
<code>\getstored</code>	3, 4, 6
<code>\meaningsc</code>	3, 7
<code>\mergesc</code>	6
<code>\newenvsc</code>	3, 5
<code>\scontents</code>	4
<code>\setupsc</code>	3–5
<code>\startscontents</code>	4
<code>\stopscontents</code>	4
<code>\tpestored</code>	3, 6, 7
E	
Environments provide by SCONTENTS :	
<code>scontents</code>	3–5
<code>verbatimsc</code>	6, 7, 11–13
Environments:	
<code>Verbatim</code>	3
<code>filecontentsdefmacro</code>	1
<code>lstlisting</code>	3
K	
Keys provide by SCONTENTS :	
<code>after</code>	6
<code>before</code>	6
<code>force-eol</code>	3–5
<code>meaningsc</code>	6
<code>overwrite</code>	3–6
<code>print-all</code>	3
<code>print-cmd</code>	3, 5, 6
<code>print-env</code>	3–5
<code>sep</code>	6
<code>start</code>	6
<code>step</code>	6
<code>stop</code>	6
<code>store-all</code>	3
<code>store-cmd</code>	3, 5
<code>store-env</code>	3–5
<code>tpestored</code>	6
<code>verb-font</code>	3
<code>width-tab</code>	3, 6, 7
<code>wrapper</code>	6
<code>write-cmd</code>	3, 5
<code>write-env</code>	3, 4
<code>write-out</code>	3–6
L	
<code>\lstinline</code>	5
M	
<code>\meaning</code>	7
P	
Packages:	
<code>answers</code>	8
<code>expl3</code>	1, 2, 8, 16
<code>fancyhdr</code>	7
<code>fancyvrb</code>	3, 7, 11
<code>filecontentsdef</code>	1, 8, 9, 16
<code>fvextra</code>	5
<code>l3seq</code>	1, 8
<code>listings</code>	3, 5, 7, 12
<code>minted</code>	7, 13
<code>scontents</code>	1, 2, 7, 8, 16
<code>tcolorbox</code>	11
V	
<code>\Verb</code>	5
<code>\verb</code>	5

11 References

- [1] The L^AT_EX Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/expl3>, 2023.
- [2] The L^AT_EX Project. “The xparse package”. Available from CTAN, <https://www.ctan.org/pkg/xparse>, 2023.
- [3] The L^AT_EX Project. “The l3keys2e package”. Available from CTAN, <https://www.ctan.org/pkg/l3keys2e>, 2022.
- [4] WRIGHT, JOSEPH. “Programming key–value in expl3”. Available from TUGBOAT, <https://www.tug.org/TUGboat/tb31-1/tb97wright-l3keys.pdf>, 2010.

12 Implementation

The most recent publicly released version of `SCONTENTS` is available at CTAN: <https://www.ctan.org/pkg/scontents>. Historical and developmental versions are available at <https://github.com/pablgonz/scontents>. While general feedback via email is welcomed, specific bugs or feature requests should be reported through the issue tracker: <https://github.com/pablgonz/scontents/issues>.

12.1 Declaration of the package

First we set up the module name for `l3doc`:

```
1 <@@=scontents>
```

Now we define some common macros to hold the package date and version:

```
2 <loader>\def\ScontentsFileDate{2025-04-23}%
3 <core>\def\ScontentsCoreFileDate{2025-04-23}%
4 <*loader>
5 \def\ScontentsFileVersion{2.3}%
6 \def\ScontentsFileDescription{Stores LaTeX contents in memory or files}%
```

The \LaTeX loader is quite simple, we just need to make sure of the minimum version for correct operation and then set interfaces up. The choice of \LaTeX release 2024-11-01 is the latest available in \TeX Live 2024 (frozen) and is necessary to be able to implement the package's full compatibility with *tagged* PDF.

```
7 <*latex>
8 \NeedsTeXFormat{LaTeX2e}[2024-11-01]
9 \ProvidesExplPackage
10   {scontents} {\ScontentsFileDate} {\ScontentsFileVersion} {\ScontentsFileDescription}
11 </latex>
```

The plain \TeX and \ConTeXt loaders are similar (probably because I don't know how to make a proper \ConTeXt module :-). We define a \LaTeX -style `\ver@scontents.sty` macro with version info (just in case) and add `\ExplSyntaxOn` to be able to load `xparse` later.

```
12 <!!latex>
13 <context>\writestatus{loading}{User Module scontents v\ScontentsFileVersion}
14 <context>\unprotect
15 \input expl3-generic.tex
16 \ExplSyntaxOn
17 \tl_gset:ce { ver @ scontents . sty } { \ScontentsFileDate\space
18   v\ScontentsFileVersion\space \ScontentsFileDescription }
19 \iow_log:e { Package: ~ scontents ~ \use:c { ver @ scontents . sty } }
20 </!!latex>
```

In plain \TeX , check that the package isn't being loaded twice (\LaTeX and \ConTeXt already defend against that):

```
21 <*plain>
22 \msg_set:nnn { scontents } { already-loaded }
23   { The~'scontents'~package~is~already~loaded.~Aborting~input~\msg_line_context:. }
24 \cs_if_exist:NT \__scontents_rescan_tokens:n
25   {
26     \msg_warning:nn { scontents } { already-loaded }
27     \ExplSyntaxOff
28     \file_input_stop:
29   }
30 </plain>
```

12.2 Definition of variables by format

We define and set variables that must be handled separately in order to work properly with plain \TeX , \ConTeXt and \LaTeX .

`\g_scontents_end_verbatimsc_tl` A global token list `\g__scontents_end_verbatimsc_tl` match when ending `verbatimsc` environment.

```
31 \tl_new:N \g_scontents_end_verbatimsc_tl
32 \tl_gset_rescan:Nnn \g_scontents_end_verbatimsc_tl
33   {
34     \char_set_catcode_other:N \
35 <*latex>
36     \char_set_catcode_other:N {\
37     \char_set_catcode_other:N \}
38 </latex>
```

```

39 }
40 <latex> { \end{verbatim} }
41 <plain> { \endverbatim }
42 <context> { \stopverbatim }

```

(End of definition for `\g__scontents_end_verbatimsc_tl`.)

`\c__scontents_end_env_tl` A token list `\c__scontents_end_env_tl` match when ending environments defined by `\newenvsc`,
`\l__scontents_env_name_tl` `\l__scontents_env_name_tl` storing the name of environments defined by `\newenvsc`.

```

43 \tl_new:N \l__scontents_env_name_tl
44 \tl_const:Nc \c__scontents_end_env_tl
45 {
46   \c_backslash_str
47 <latex|plain> end
48 <context> stop
49 <latex> \c_left_brace_str
50   \exp_not:N \l__scontents_env_name_tl
51 <latex> \c_right_brace_str
52 }

```

(End of definition for `\c__scontents_end_env_tl` and `\l__scontents_env_name_tl`.)

Now we load the core `SCONTENTS` code:

```

53 \file_input:n { scontents-code.tex }

```

`__scontents_format_case:nnn` Sometimes we need to detect the format from within a macro:

```

54 \cs_new:Npn \__scontents_format_case:nnn #1 #2 #3
55 <latex> {#1} % LaTeX
56 <plain> {#2} % Plain/Generic
57 <context> {#3} % ConTeXt

```

(End of definition for `__scontents_format_case:nnn`.)

Checking that the package was loaded with the proper loader code. This code was copied from `expl3-code.tex`.

```

58 </loader>
59 <*core>
60 \begingroup
61   \catcode32=10
62   \endlinechar=32
63   \def\next{\endgroup}%
64   \expandafter\ifx\csname PackageError\endcsname\relax
65     \begingroup
66       \def\next{\endgroup\endgroup}%
67       \def\PackageError#1#2#3%
68         {%
69           \endgroup
70           \errhelp{#3}%
71           \errmessage{#1 Error: #2!}%
72         }%
73   \fi
74   \expandafter\ifx\csname ScontentsFileDate\endcsname\relax
75     \def\next
76       {%
77         \PackageError{scontents}{No scontents loader detected}
78         {%
79           You have attempted to use the scontents code directly rather than using
80           the correct loader. Loading of scontents will abort.
81         }%
82       \endgroup
83       \endinput
84     }%
85   \else
86     \ifx\ScontentsFileDate\ScontentsCoreFileDate
87     \else
88       \def\next
89         {%
90           \PackageError{scontents}{Mismatched scontents files detected}
91         }%

```

```

92         You have attempted to load scontentts with mismatched files:
93         probably you have one or more files 'locally installed' which
94         are in conflict. Loading of scontentts will abort.
95     }%
96     \endgroup
97     \endinput
98 }%
99 \fi
100 \fi
101 \next

```

12.3 Definition of temporary variables

The token list `\l__scontentts_macro_tmp_tl` is a temporary token list to hold the contents of the macro/environment. `\l__scontentts_temp_tl`, `\g__scontentts_temp_tl`, `\l__scontentts_tmpa_int` and `\l__scontentts_temp_bool` are generic temporary vars.

```

102 \tl_new:N \l__scontentts_macro_tmp_tl
103 \tl_new:N \l__scontentts_temp_tl
104 \tl_new:N \g__scontentts_temp_tl
105 \int_new:N \l__scontentts_tmpa_int
106 \bool_new:N \l__scontentts_temp_bool

```

(End of definition for `\l__scontentts_macro_tmp_tl` and others.)

`\l__scontentts_keys_tl` Stores unused *(keys)* to be forwarded to other commands.

```

107 \tl_new:N \l__scontentts_keys_tl

```

(End of definition for `\l__scontentts_keys_tl`.)

12.4 Compatibility layer with plain T_EX and ConT_EXt

When loading the package outside of L^AT_EX we can't usually use `xparse`. However since `xparse` now `ltxcmds` is part of the L^AT_EX kernel is loadable in any format.

```

108 </core>
109 <*loader&!latex>
110 \int_set:Nn \l__scontentts_tmpa_int { \char_value_catcode:n { \@ } }
111 \char_set_catcode_letter:N \@
112 \file_input:n { xparse-generic.tex }
113 \char_set_catcode:nn { \@ } { \l__scontentts_tmpa_int }
114 </loader&!latex>
115 <*core>

```

12.5 Definition of keys for the package

We create some common *(keys)* that will be used by the options passed to the package as well as by the environments and commands defined.

```

116 \keys_define:nn { scontentts }
117 {
118   store-env .tl_set:N          = \l__scontentts_name_seq_env_tl,
119   store-env .initial:n        = contents,
120   store-env .value_required:n = true,
121   store-cmd .tl_set:N         = \l__scontentts_name_seq_cmd_tl,
122   store-cmd .initial:n        = contents,
123   store-cmd .value_required:n = true,
124   verb-font .tl_set:N         = \l__scontentts_verb_font_tl,
125   verb-font .value_required:n = true,
126   print-env .bool_set:N       = \l__scontentts_print_env_bool,
127   print-env .initial:n        = false,
128   print-env .default:n        = true,
129   print-cmd .bool_set:N       = \l__scontentts_print_cmd_bool,
130   print-cmd .initial:n        = false,
131   print-cmd .default:n        = true,
132   force-eol .bool_set:N       = \l__scontentts_forced_eol_bool,
133   force-eol .initial:n        = false,
134   force-eol .default:n        = true,
135   overwrite .bool_set:N       = \l__scontentts_overwrite_bool,
136   overwrite .initial:n        = false,
137   overwrite .default:n        = true,

```

```

138 width-tab .int_set:N = \l__scontents_tab_width_int,
139 width-tab .initial:n = 1,
140 width-tab .value_required:n = true,
141 print-all .meta:n = { print-env = #1 , print-cmd = #1 },
142 print-all .default:n = true,
143 store-all .meta:n = { store-env = #1 , store-cmd = #1 },
144 store-all .value_required:n = true
145 }
146 </core>
147 <loader>\keys_define:nn { scontents }
148 <latex> { verb-font .initial:n = \ttfamily }
149 <plain|context> { verb-font .initial:n = \tt

```

In \LaTeX mode process the $\langle keys \rangle$ as options passed on to the package and will return an error when they are.

```

150 <*latex>
151 \ProcessKeyOptions [ scontents ]
152 </latex>
153 <*core>

```

12.6 Internal variables and utility functions

$\backslash\l__scontents_fname_out_tl$ The token list $\backslash\l__scontents_fname_out_tl$ is used for store the name of the $\langle output\ file \rangle$, when there's one. Its value is set by the keys `write-env`, `write-out` and `write-cmd`.

$\backslash\l__scontents_every_line_env_tl$ The token list $\backslash\l__scontents_every_line_env_tl$ holds the contents of an environment, `scontents` by default, as it's being read. $\backslash\l__scontents_file_iow$ is an output stream for saving the contents of an environment (or command) to a file.

This variables is used by the function $\backslash\l__scontents_file_tl_write_start:n$ (see 12.10.5).

```

154 \tl_new:N \l__scontents_fname_out_tl
155 \tl_new:N \l__scontents_every_line_env_tl
156 \iow_new:N \l__scontents_file_iow

```

(End of definition for $\backslash\l__scontents_fname_out_tl$, $\backslash\l__scontents_every_line_env_tl$, and $\backslash\l__scontents_file_iow$.)

$\backslash\l__scontents_foreach_name_seq_tl$ $\backslash\l__scontents_foreach_name_seq_tl$ is the name assigned to the sequence on which the loop will be made, $\backslash\l__scontents_foreach_before_tl$ and $\backslash\l__scontents_foreach_after_tl$ are token lists in which the assigned material will be placed before and after the execution of the $\backslash\foreachsc$ loop.

```

157 \tl_new:N \l__scontents_foreach_name_seq_tl
158 \tl_new:N \l__scontents_foreach_before_tl
159 \tl_new:N \l__scontents_foreach_after_tl

```

(End of definition for $\backslash\l__scontents_foreach_name_seq_tl$, $\backslash\l__scontents_foreach_before_tl$, and $\backslash\l__scontents_foreach_after_tl$.)

$\backslash\l__scontents_env_nesting_int$ $\backslash\l__scontents_seq_item_seq$ stores the indexes in the sequence of the items requested to $\backslash\typetored$ or \backslashmeaningcs . $\backslash\l__scontents_env_nesting_int$ stores the current nesting level of the `scontents` environment. $\backslash\l__scontents_foreach_stop_int$ will save the value at which the $\backslash\foreachsc$ loop will stop.

```

160 \int_new:N \l__scontents_foreach_stop_int
161 \seq_new:N \l__scontents_seq_item_seq
162 \int_new:N \l__scontents_env_nesting_int

```

(End of definition for $\backslash\l__scontents_env_nesting_int$ and $\backslash\l__scontents_foreach_stop_int$.)

$\backslash\l__scontents_writing_bool$ The boolean $\backslash\l__scontents_writing_bool$ keeps track of whether we should write to a file, and $\backslash\l__scontents_storing_bool$ determines whether it is in write-only mode when the key `write-out` is used.

```

163 \bool_new:N \l__scontents_writing_bool
164 \bool_set_false:N \l__scontents_writing_bool
165 \bool_new:N \l__scontents_storing_bool
166 \bool_set_true:N \l__scontents_storing_bool
167 \bool_new:N \l__scontents_writable_bool

```

(End of definition for $\backslash\l__scontents_writing_bool$, $\backslash\l__scontents_storing_bool$, and $\backslash\l__scontents_writable_bool$.)

```

\l__scontents_foreach_before_bool
\l__scontents_foreach_after_bool
\l__scontents_foreach_stop_bool
\l__scontents_foreach_wrapper_bool
168 \bool_new:N \l__scontents_foreach_before_bool
169 \bool_set_false:N \l__scontents_foreach_before_bool
170 \bool_new:N \l__scontents_foreach_after_bool
171 \bool_set_false:N \l__scontents_foreach_after_bool
172 \bool_new:N \l__scontents_foreach_stop_bool
173 \bool_set_false:N \l__scontents_foreach_stop_bool
174 \bool_new:N \l__scontents_foreach_wrapper_bool
175 \bool_set_false:N \l__scontents_foreach_wrapper_bool

```

(End of definition for `\l__scontents_foreach_before_bool` and others.)

```

\l__scontents_foreach_print_seq
176 \seq_new:N \l__scontents_foreach_print_seq
177 \seq_new:c { g__scontents_name_sc!internal_seq }

```

(End of definition for `\l__scontents_foreach_print_seq`.)

`\c__scontents_hidden_space_str` `\c__scontents_hidden_space_str` is a constant *string* to used to hide the *forced space* added by T_EX when recording content in a macro. This *string* contains the *reserved phrase* “`%^^Ascheol%`” which is added to the end of the argument stored in `seq` when the key `force-eol` is false.

```

178 \str_const:Ne \c__scontents_hidden_space_str
179 { \c_percent_str \c_circumflex_str \c_circumflex_str A scheol \c_percent_str }

```

(End of definition for `\c__scontents_hidden_space_str`.)

```

\q__scontents_stop
\q__scontents_mark
180 \quark_new:N \q__scontents_stop
181 \quark_new:N \q__scontents_mark

```

(End of definition for `\q__scontents_stop` and `\q__scontents_mark`.)

```

\s__scontents_stop
\s__scontents_mark
182 \scan_new:N \s__scontents_stop
183 \scan_new:N \s__scontents_mark

```

(End of definition for `\s__scontents_stop` and `\s__scontents_mark`.)

```

\l__scontents_cur_seq_name_str
184 \str_new:N \l__scontents_cur_seq_name_str

```

(End of definition for `\l__scontents_cur_seq_name_str`.)

```

\__scontents_use_i_delimit_by_s_stop:nw
\__scontents_use_none_delimit_by_s_stop:w
185 \cs_new:Npn \__scontents_use_delimit_by_s_stop:nw #1 \s__scontents_stop {#1}
186 \cs_new:Npn \__scontents_use_i_delimit_by_s_stop:nw #1 #2 \s__scontents_stop {#1}
187 \cs_new:Npn \__scontents_use_none_delimit_by_s_stop:w #1 \s__scontents_stop { }

```

(End of definition for `__scontents_use_i_delimit_by_s_stop:nw` and `__scontents_use_none_delimit_by_s_stop:w`.)

```

\l__scontents_save_sf_int
\l__scontents_save_skip
188 \int_new:N \l__scontents_save_sf_int
189 \skip_new:N \l__scontents_save_skip

```

(End of definition for `\l__scontents_save_sf_int` and `\l__scontents_save_skip`.)

`__scontents_rescan_tokens:n`
`__scontents_rescan_tokens:x`
`__scontents_rescan_tokens:v`

The function `\tl_rescan:nn` provided by `expl3` doesn’t fit the needs of this package because it does not allow catcode changes inside the argument, so verbatim commands used inside one of `SCONTENTS`’s commands/environments will not work. Here we create a private copy of `\tex_scantokens:D` which will serve our purposes. See the answer by Ulrich Diez in [How do use {<setup>} in \tl_set_rescan:Nnn to replace \scantokens?](#)

```

190 \cs_new_protected:Npn \__scontents_rescan_tokens:n #1 { \tex_scantokens:D {#1} }
191 \cs_generate_variant:Nn \__scontents_rescan_tokens:n { V, x }

```

©2019–2025 by Pablo González

(End of definition for `__scontents_rescan_tokens:n`.)

```

__scontents_tab: Control sequences to replace tab ( $\text{^^I}$ ) and form feed ( $\text{^^L}$ ) characters.
__scontents_par:
192 \cs_new:Npe __scontents_tab: { \c_space_tl }
193 \cs_new:Npn __scontents_par: { ^^J ^^J }

```

(End of definition for `__scontents_tab:` and `__scontents_par:.`)

```

\tl_if_empty: fTF Some nonstandard kernel variant.
194 \prg_generate_conditional_variant:Nnn \tl_if_empty:n { f } { p, TF }

```

(End of definition for `\tl_if_empty: fTF.`)

12.7 Defining keys for the environment and commands

We add the *keys* divided into subgroups to handle errors and *unknown keys* separately.

12.7.1 Keys for environment scontents

We define a set of *keys* for environment `scontents`.

```

195 \keys_define:nn { scontents / scontents }
196 {
197   write-env .code:n          = {
198     \bool_set_true:N \__scontents_writing_bool
199     \tl_set:Nn \__scontents_fname_out_tl {#1}
200   },
201   write-out .code:n         = {
202     \bool_set_false:N \__scontents_storing_bool
203     \bool_set_true:N \__scontents_writing_bool
204     \tl_set:Nn \__scontents_fname_out_tl {#1}
205   },
206   write-env .value_required:n = true,
207   write-out .value_required:n = true,
208   print-env .meta:nn         = { scontents } { print-env = #1 },
209   print-env .default:n       = true,
210   store-env .meta:nn         = { scontents } { store-env = #1 },
211   force-eol .meta:nn         = { scontents } { force-eol = #1 },
212   force-eol .default:n       = true,
213   overwrite .meta:nn         = { scontents } { overwrite = #1 },
214   overwrite .default:n       = true,
215   unknown .code:n           = { \__scontents_parse_environment_keys:n {#1} }
216 }

```

12.7.2 Keys for command \Scontents

We define a set of *keys* for commands `\Scontents` and `\Scontents*`.

```

217 \keys_define:nn { scontents / Scontents }
218 {
219   write-cmd .code:n          = {
220     \bool_set_true:N \__scontents_writing_bool
221     \tl_set:Nn \__scontents_fname_out_tl {#1}
222   },
223   write-out .code:n         = {
224     \bool_set_false:N \__scontents_storing_bool
225     \bool_set_true:N \__scontents_writing_bool
226     \tl_set:Nn \__scontents_fname_out_tl {#1}
227   },
228   write-cmd .value_required:n = true,
229   write-out .value_required:n = true,
230   print-cmd .meta:nn         = { scontents } { print-cmd = #1 },
231   print-cmd .default:n       = true,
232   store-cmd .meta:nn         = { scontents } { store-cmd = #1 },
233   force-eol .meta:nn         = { scontents } { force-eol = #1 },
234   force-eol .default:n       = true,
235   overwrite .meta:nn         = { scontents } { overwrite = #1 },
236   overwrite .default:n       = true,
237   unknown .code:n           = { \__scontents_parse_command_keys:n {#1} }
238 }

```

12.7.3 Keys for command `\foreachsc`

We define a set of *(keys)* for command `\foreachsc`.

```

239 \keys_define:nn { scontents / foreachsc }
240 {
241   before .code:n      = {
242     \bool_set_true:N \l__scontents_foreach_before_bool
243     \tl_set:Nn \l__scontents_foreach_before_tl {#1}
244   },
245   before .value_required:n = true,
246   after  .code:n      = {
247     \bool_set_true:N \l__scontents_foreach_after_bool
248     \tl_set:Nn \l__scontents_foreach_after_tl {#1}
249   },
250   after  .value_required:n = true,
251   start  .int_set:N      = \l__scontents_foreach_start_int,
252   start  .value_required:n = true,
253   start  .initial:n      = 1,
254   stop   .code:n      = {
255     \bool_set_true:N \l__scontents_foreach_stop_bool
256     \int_set:Nn \l__scontents_foreach_stop_int {#1}
257   },
258   stop   .value_required:n = true,
259   step   .int_set:N      = \l__scontents_foreach_step_int,
260   step   .value_required:n = true,
261   step   .initial:n      = 1,
262   wrapper .code:n      = {
263     \bool_set_true:N \l__scontents_foreach_wrapper_bool
264     \cs_set_protected:Npn
265     \l__scontents_foreach_wrapper:n #1 {#1}
266   },
267   wrapper .value_required:n = true,
268   sep     .tl_set:N      = \l__scontents_foreach_sep_tl,
269   sep     .initial:n      = {},
270   sep     .value_required:n = true,
271   unknown .code:n      = { \l__scontents_parse_foreach_keys:n {#1} }
272 }

```

12.7.4 Key for commands `\typestored` and `\meaningsc`

We define a *(key)* for command `\typestored` and `\meaningsc`. Both commands accept the same type of optional arguments, just define a common *(key)*.

```

273 \bool_new:N \l__scontents_print_aux_bool
274 \bool_set_true:N \l__scontents_print_aux_bool
275 \keys_define:nn { scontents / typemeaning }
276 {
277   width-tab .meta:nn = { scontents } { width-tab = #1 },
278   write-out .code:n  = {
279     \bool_set_false:N \l__scontents_storing_bool
280     \bool_set_true:N \l__scontents_writing_bool
281     \tl_set:Nn \l__scontents_fname_out_tl {#1}
282   },
283   overwrite .meta:nn = { scontents } { overwrite = #1 },
284   overwrite .default:n = true,
285   unknown   .code:n  = { \l__scontents_parse_type_meaning_key:n {#1} }
286 }

```

12.8 Handling undefined keys

The *(keys)* are stored in the string variable `\l_keys_key_str`, and the value (if any) is passed as an argument to each *(function)*.

12.8.1 Undefined keys for environment `scontents`

We check the *(keys)* passed to the environment `scontents` and process it with `\l__scontents_parse_environment_keys:n` if the *(key)* is *unknown* we return an error message.

```

287 \cs_new_protected:Npn \l__scontents_parse_environment_keys:n #1
288 { \exp_args:NV \l__scontents_parse_environment_keys:nn \l_keys_key_str {#1} }
289 \cs_new_protected:Npn \l__scontents_parse_environment_keys:nn #1#2
290 {

```



```

291 \tl_if_blank:nTF {#2}
292   { \msg_error:nnn { scontents } { env-key-unknown } {#1} }
293   { \msg_error:nnnn { scontents } { env-key-value-unknown } {#1} {#2} }
294 }

```

(End of definition for `__scontents_parse_environment_keys:n` and `__scontents_parse_environment_keys:nn`.)

12.8.2 Undefined keys for `\Scontents` and `\Scontents*`

`__scontents_parse_command_keys:n` We check the `<keys>` passed to commands `\Scontents` or `\Scontents*` and process it with `__scontents_parse_command_keys:n` if the `<key>` is *unknown* we return an error message.

```

295 \cs_new_protected:Npn \__scontents_parse_command_keys:n #1
296   { \exp_args:NV \__scontents_parse_command_keys:nn \l_keys_key_str {#1} }
297 \cs_new_protected:Npn \__scontents_parse_command_keys:nn #1#2
298   {
299     \tl_if_blank:nTF {#2}
300     { \msg_error:nnn { scontents } { cmd-key-unknown } {#1} }
301     { \msg_error:nnnn { scontents } { cmd-key-value-unknown } {#1} {#2} }
302   }

```

(End of definition for `__scontents_parse_command_keys:n` and `__scontents_parse_command_keys:nn`.)

12.8.3 Undefined keys for `\foreachsc`

`__scontents_parse_foreach_keys:n` We check the `<keys>` passed to command `\foreachsc` and process it with `__scontents_parse_foreach_keys:n`, if the `<key>` is *unknown* we return an error message.

```

303 \cs_new_protected:Npn \__scontents_parse_foreach_keys:nn #1#2
304   {
305     \tl_if_blank:nTF {#2}
306     { \msg_error:nnn { scontents } { for-key-unknown } {#1} }
307     { \msg_error:nnnn { scontents } { for-key-value-unknown } {#1} {#2} }
308   }
309 \cs_new_protected:Npn \__scontents_parse_foreach_keys:n #1
310   { \exp_args:NV \__scontents_parse_foreach_keys:nn \l_keys_key_str {#1} }

```

(End of definition for `__scontents_parse_foreach_keys:n` and `__scontents_parse_foreach_keys:nn`.)

12.8.4 Undefined keys for `\typestored` and `\meaningsc`

`__scontents_parse_type_meaning_key:n` The commands `\typestored` and `\meaningsc` accept an optional argument for setting the `width-tab` to print the stored contents. However their optional argument also contains the number of the item to retrieve from the stored sequence. To avoid the awkward `\typestored[] [options] { ... }` syntax, we'll make the commands have a single optional argument which is processed by `\l3keys`, and the unknown keys are brought here to `__scontents_parse_type_meaning_key:n` to process.

First we check if the `<key>` is an integer using `\int_to_roman:n`. If it is, we check that the value passed to the key is blank (otherwise something odd as `1=1` might have been used). If everything is correct, then set the value of the integer which holds the `<index>`. Otherwise raise an error about an *unknown* option.

```

311 \cs_new_protected:Npn \__scontents_parse_type_meaning_key:n #1
312   { \exp_args:NV \__scontents_parse_type_meaning_key:nn \l_keys_key_str {#1} }
313 \cs_new_protected:Npn \__scontents_parse_type_meaning_key:nn #1#2
314   {
315     \tl_if_blank:nTF {#2}
316     { \__scontents_parse_type_meaning_range:w #1 - \__scontents_mark - \s__scontents_mark }
317     { \msg_error:nnnn { scontents } { type-key-value-unknown } {#1} {#2} }
318   }
319 \cs_new_protected:Npn \__scontents_parse_type_meaning_range:w #1 - #2 - #3 \s__scontents_mark
320   {
321     \__scontents_range_parser:nnxn {#1} {#2}
322     { \seq_count:c { g__scontents_name_\l__scontents_cur_seq_name_str _seq } }
323     { \msg_error:nnn { scontents } { type-key-unknown } }
324   }
325 \cs_new_protected:Npn \__scontents_range_parser:nnnn #1 #2 #3 #4
326   {
327     \exp_args:Nxx \__scontents_range_parser_aux:nnn
328     { \str_if_eq:nnTF {#1} { end } {#3} { \exp_not:n {#1} } }
329     { \str_if_eq:nnTF {#2} { end } {#3} { \exp_not:n {#2} } }
330     { #4 }
331   }
332 \cs_generate_variant:Nn \__scontents_range_parser:nnnn { nnnx }

```

```

333 \cs_new_protected:Npn \__scontents_range_parser_aux:nnn #1 #2 #3
334 {
335   \__scontents_tl_if_head_is_q_mark:nTF {#2}
336   {
337     \tl_if_empty:FTF { \int_to_roman:n { -0 #1 } }
338     { \seq_put_right:Ne \l__scontents_seq_item_seq { \int_eval:n {#1} } }
339     { #3 {#1} }
340   }
341   {
342     \bool_lazy_and:nnTF
343     { \tl_if_empty_p:f { \int_to_roman:n { -0 #1 } } }
344     { \tl_if_empty_p:f { \int_to_roman:n { -0 #2 } } }
345     {
346       \int_compare:nNnTF {#2} > {#1}
347       { \int_step_inline:nnnn {#1} { 1 } {#2} }
348       { \int_step_inline:nnnn {#1} { -1 } {#2} }
349       { \seq_put_right:Nn \l__scontents_seq_item_seq {##1} }
350     }
351     { #3 { #1-#2 } }
352   }
353 }

```

(End of definition for `__scontents_parse_type_meaning_key:n` and `__scontents_parse_type_meaning_key:nn`.)

12.9 Programming of the sequences

The storage of the package is done using seq variables. Here we set up the macros that will manage the variables.

```

\__scontents_append_contents:nn
\__scontents_append_contents:Vx

```

The function `__scontents_append_contents:nn` creates a seq variable if one didn't exist and appends the contents in the argument to the right of the sequence.

```

354 \cs_new_protected:Npn \__scontents_append_contents:nn #1#2
355 {
356   \tl_if_blank:nT {#1}
357   { \msg_error:nn { scontents } { empty-store-cmd } }
358   \seq_if_exist:cF { g__scontents_name_#1_seq }
359   { \seq_new:c { g__scontents_name_#1_seq } }
360   \seq_gput_right:cn { g__scontents_name_#1_seq } {#2}
361 }
362 \cs_generate_variant:Nn \__scontents_append_contents:nn { Vx }

```

(End of definition for `__scontents_append_contents:nn`.)

```

\__scontents_getfrom_seq:nn
\__scontents_getfrom_seq:Nn
\__scontents_getfrom_seq:nnn

```

The function `__scontents_getfrom_seq:nn` retrieves the saved item from the sequence.

```

363 \cs_new:Npn \__scontents_getfrom_seq:Nn #1#2
364 {
365   \seq_if_exist:cTF { g__scontents_name_#2_seq }
366   {
367     \exp_args:Nf \__scontents_getfrom_seq:nNn
368     { \seq_count:c { g__scontents_name_#2_seq } } #1 {#2}
369   }
370   { \msg_expandable_error:nnn { scontents } { undefined-storage } {#2} }
371 }
372 \cs_new:Npn \__scontents_getfrom_seq:nNn #1 #2 #3
373 { \seq_map_tokens:Nn #2 { \__scontents_getfrom_seq_aux:nnn {#1} {#3} } }
374 \cs_new:Npn \__scontents_getfrom_seq_aux:nnn #1 #2 #3
375 { \exp_args:Nnf \use:n { \__scontents_getfrom_seq:nnn {#1} } { \int_eval:n {#3} } {#2} }
376 \cs_new:Npn \__scontents_getfrom_seq:nn #1#2
377 {
378   \seq_if_exist:cTF { g__scontents_name_#2_seq }
379   {
380     \exp_args:Nf \__scontents_getfrom_seq:nnn
381     { \seq_count:c { g__scontents_name_#2_seq } }
382     {#1} {#2}
383   }
384   { \msg_expandable_error:nnn { scontents } { undefined-storage } {#2} }
385 }
386 \cs_new:Npn \__scontents_getfrom_seq:nnn #1#2#3
387 {

```

```

388 \bool_lazy_or:nnTF
389   { \int_compare_p:nNn {#2} = { 0 } }
390   { \int_compare_p:nNn { \int_abs:n {#2} } > {#1} }
391   { \msg_expandable_error:nnnnn { scontents } { index-out-of-range } {#2} {#3} {#1} }
392   { \seq_item:cn { g__scontents_name_#3_seq } {#2} }
393 }

```

(End of definition for `__scontents_getfrom_seq:nn`, `__scontents_getfrom_seq:Nn`, and `__scontents_getfrom_seq:nnn`.)

`__scontents_lastfrom_seq:n` The function `__scontents_lastfrom_seq:n` retrieves the last saved item from the sequence when `\l__scontents_print_env_bool` or `\l__scontents_print_cmd_bool` is true.

```

394 \cs_new_protected:Npn \__scontents_lastfrom_seq:n #1
395   {
396     \tl_gset:Ne \g__scontents_temp_tl { \seq_item:cn { g__scontents_name_#1_seq } {-1} }
397     \group_insert_after:N \__scontents_rescan_tokens:V
398     \group_insert_after:N \g__scontents_temp_tl
399     \group_insert_after:N \tl_gclear:N
400     \group_insert_after:N \g__scontents_temp_tl
401   }
402 \cs_generate_variant:Nn \__scontents_lastfrom_seq:n { V }

```

(End of definition for `__scontents_lastfrom_seq:n`.)

`__scontents_store_to_seq:NN` The function `__scontents_store_to_seq:NN` writes the recorded contents in `#1` to the log and stores it in `#2`.

```

403 \cs_new_protected:Npn \__scontents_store_to_seq:NN #1#2
404   {
405     \tl_log:N #1
406     \__scontents_append_contents:Vx #2 { \exp_not:V #1 }
407   }

```

(End of definition for `__scontents_store_to_seq:NN`.)

12.10 The command `\newenvsc` and environment `scontents`

In order to be able to define environments that behave similarly to `scontents`, we define a generic environment and make all other environment as wrappers around that one.

12.10.1 The command `\newenvsc`

`\newenvsc` The internal function `__scontents_env_setting:nn` defines two functions `__scontents_#1_env_begin:` and `__scontents_#1_env_end:`, which set the current environment's name in `#1` and `\l__scontents_env_name_tl` and default properties in `#2` then call `__scontents_setup_verb_processor:`, the generic `__scontents_env_generic_begin:` and `__scontents_env_generic_end:`. Finally the function `__scontents_env_define:nnn` will create the environments.

```

408 \cs_new_protected:Npn \__scontents_env_setting:nn #1 #2
409   {
410     \cs_new_protected:cpn { __scontents_#1_env_begin: }
411     {
412       \tl_set:Nn \l__scontents_env_name_tl {#1}
413       \keys_set:nn { scontents } {#2}
414       \__scontents_setup_verb_processor:
415       \__scontents_env_generic_begin:
416     }
417     \cs_new_protected:cpn { __scontents_#1_env_end: }
418     { \__scontents_env_generic_end: }
419     \exp_args:Nooo % http://nooooooooooooooooo.com :) jeje
420     \__scontents_env_define:nnn { \tl_to_str:n {#1} }
421     { \cs:w __scontents_#1_env_begin: \cs_end: }
422     { \cs:w __scontents_#1_env_end: \cs_end: }
423   }
424 </core>
425 <*loader>
426 \NewDocumentCommand \newenvsc { m O{} }
427   {
428   <latex|plain> \cs_if_exist:cTF { #1 }
429   <context> \cs_if_exist:cTF { start #1 }
430     { \msg_error:nnn { scontents } { env-already-defined } {#1} }

```

```

431     { \_scontents_env_setting:nn {#1} {#2} }
432   }
433 \cs_new_protected:Npn \_scontents_env_define:nnn #1 #2 #3
434   {
435   <latex|plain>   \NewDocumentEnvironment {#1} { }
436   <context>       \cs_new_protected:cpn { start #1 }
437     {
438     <!latex>       \group_begin:
439                   #2
440     }
441   <context>       \cs_new_protected:cpn { stop #1 }
442     {
443     #3
444     <!latex>       \group_end:
445     }
446   }
447 </loader>
448 <*core>

```

(End of definition for `\newenvsc`, `_scontents_env_setting:nn`, and `_scontents_env_define:nnn`. This function is documented on page 5.)

12.10.2 Generic definition of the environment

`_scontents_env_generic_begin:` Now we define the generic environment functions `_scontents_env_generic_begin:` and `_scontents_env_generic_end:`.

```

449 \cs_new_protected:Npn \_scontents_env_generic_begin:
450   {
451   \char_set_catcode_active:N ^^M
452   \_scontents_start_environment:w
453   }
454 \cs_new_protected:Npn \_scontents_env_generic_end:
455   {
456   \_scontents_stop_environment:
457   \_scontents_finish_storing:NNN \\_scontents_macro_tmp_tl
458   \\_scontents_name_seq_env_tl \\_scontents_print_env_bool
459   }

```

(End of definition for `_scontents_env_generic_begin:` and `_scontents_env_generic_end:`.)

12.10.3 Definition of the environment scontents

Finally defining the `scontents` environment should be easy :)

```

460 </core>
461 <loader>\newenvsc{scontents}
462 <*core>

```

(End of definition for `scontents` and others. These functions are documented on page 4.)

12.10.4 key val for environment

`_scontents_grab_optional:n` The macro `_scontents_grab_optional:w` is called from the `scontents` environment with the tokens following the `\begin{scontents}` when the next character is a `[`. This function is defined using `xparse` to exploit its delimited argument processor.

The function is called from a context where `^^M` is active, so `_scontents_normalise_line_ends:N` is used to replace active `^^M` characters by spaces.

```

463 </core>
464 <*loader>
465 \NewDocumentCommand \_scontents_grab_optional:w { r[] }
466   { \_scontents_grab_optional:n {#1} }
467 </loader>
468 <*core>
469 \cs_new_protected:Npn \_scontents_grab_optional:n #1
470   {
471   \tl_if_novalue:nF {#1}
472     {
473     \tl_set:Nn \\_scontents_temp_tl {#1}
474     \_scontents_normalise_line_ends:N \\_scontents_temp_tl
475     \keys_set:nV { scontents / scontents } \\_scontents_temp_tl
476     }

```

```

477   \_scontents_start_after_option:w
478 }

```

(End of definition for `_scontents_grab_optional:n` and `_scontents_grab_optional:w`.)

12.10.5 The environment itself

```

\_scontents_start_environment:w
\_scontents_start_after_option:w
\_scontents_check_line_process:xn
  \_scontents_stop_environment:

```

Here we make `^^I`, `^^L` and `^^M` active characters so that the end of line can be “seen” to be used as a delimiter, and \TeX doesn’t try to eliminate space-like characters.

First we check if the immediate next token after `\begin{scontents}` is a `[`. If it is, then `_scontents_grab_optional:w` is called to do the heavy lifting. `_scontents_grab_optional:w` processes the optional argument and calls `_scontents_start_after_option:w`.

The function `_scontents_start_after_option:w` also checks for trailing tokens after the optional argument and issues an error if any.

In all cases, the function `_scontents_check_line_process:xn` checks that everything past `\begin{scontents}` is empty and then process the environment.

The function `_scontents_check_line_process:xn` calls the function `_scontents_file_tl_write_start:V` which will then read the contents of the environment and optionally store them in a token list or write to an external file.

When that’s done, the function `_scontents_file_write_stop:N` does the cleanup. This part of the code is inspired and adapted from the code of the package `xsimverb` by Clemens Niederberger.

```

479 \group_begin:
480   \char_set_catcode_active:N \^^I
481   \char_set_catcode_active:N \^^L
482   \char_set_catcode_active:N \^^M
483   \cs_new_protected:Npn \_scontents_normalise_line_ends:N #1
484     { \tl_replace_all:Nnn #1 { ^^M } { ~ } }
485   \cs_new_protected:Npn \_scontents_start_environment:w #1 ^^M
486     {
487       \tl_if_head_is_N_type:nTF {#1}
488         {
489           \str_if_eq:eeTF { \tl_head:n {#1} } { [ ]
490             { \_scontents_grab_optional:w #1 ^^M }
491             { \_scontents_check_line_process:xn { } {#1} }
492           }
493           { \_scontents_check_line_process:xn { } {#1} }
494         }
495   \cs_new_protected:Npn \_scontents_start_after_option:w #1 ^^M
496     { \_scontents_check_line_process:xn { [...] } {#1} }
497   \cs_new_protected:Npn \_scontents_check_line_process:xn #1 #2
498     {
499       \tl_if_blank:nF {#2}
500         {
501           \msg_error:nnxn { scontents } { junk-after-begin }
502           { after~\c_backslash_str begin { \_scontents_env_name_tl } #1 } {#2}
503         }
504       \_scontents_make_control_chars_active:
505       \_scontents_file_tl_write_start:V \_scontents_fname_out_tl
506     }
507   \cs_new_protected:Npn \_scontents_stop_environment:
508     {
509       \_scontents_file_write_stop:N \_scontents_macro_tmp_tl
510       \bool_lazy_and:nnT
511         { \_scontents_storing_bool }
512         { \tl_if_empty_p:N \_scontents_macro_tmp_tl }
513         {
514           \msg_warning:nnx { scontents } { empty-environment } { \_scontents_env_name_tl }
515         }
516     }

```

(End of definition for `_scontents_start_environment:w` and others.)

```

\_scontents_file_tl_write_start:n
\_scontents_file_tl_write_start:V
\_scontents_verb_processor_iterate:w
\_scontents_verb_processor_iterate:nnn
\_scontents_setup_verb_processor:
  \_scontents_file_write_stop:N
  \_scontents_remove_leading_nl:n

```

This is the main macro to collect the contents of a verbatim environment. The macro starts a group, opens the *(output file)*, if necessary, sets verbatim catcodes, and then issues `^^M` (set equal to `_scontents_ret:w`) to read the environment line by line until reaching its end. The output token list will be appended with an active `^^J` character and the line just read, and this line is written to the output file, if any. At the end of the environment the *(output file)* is closed (if it was open), and the output token list is smuggled

out of the verbatim group. A leading ^^J is removed from the token list using `__scontents_remove_leading_n\l:n` (which expects an active ^^J token at the head of the token list; a low level TeX error is raised otherwise).

```

517 \cs_new_protected:Npn \__scontents_file_tl_write_start:n #1
518 {
519   \group_begin:
520   \__scontents_file_if_writable:nTF {#1}
521   {
522     \bool_set_true:N \__scontents_writable_bool
523     \iow_open:Nn \__scontents_file_iow {#1}
524   }
525   { \bool_set_false:N \__scontents_writable_bool }
526   \tl_clear:N \__scontents_every_line_env_tl
527   \seq_map_function:NN \l_char_special_seq \char_set_catcode_other:N
528   \int_step_function:nnnN { 128 } { 1 } { 255 } \char_set_catcode_letter:n
529   \cs_set_protected:Npx \__scontents_ret:w ##1 ^^M
530   {
531     \exp_not:N \__scontents_verb_processor_iterate:w
532     ##1 \c__scontents_end_env_tl
533     \c__scontents_end_env_tl
534     \exp_not:N \q__scontents_stop
535   }
536   \__scontents_make_control_chars_active:
537   \__scontents_ret:w
538 }
539 \cs_new:Npn \__scontents_setup_verb_processor:
540 {
541   \use:x
542   {
543     \cs_set:Npn \exp_not:N \__scontents_verb_processor_iterate:w
544     ###1 \c__scontents_end_env_tl
545     ###2 \c__scontents_end_env_tl
546     ###3 \exp_not:N \q__scontents_stop
547   } { \__scontents_verb_processor_iterate:nnn {##1} {##2} {##3} }
548 }
549 \cs_new:Npn \__scontents_verb_processor_iterate:nnn #1 #2 #3
550 {
551   \tl_if_blank:nTF {#3}
552   {
553     \__scontents_analyse_nesting:n {#1}
554     \__scontents_verb_processor_output:n {#1}
555   }
556   {
557     \__scontents_if_nested:TF
558     {
559       \__scontents_nesting_decr:
560       \__scontents_verb_processor_output:x
561       { \exp_not:n {#1} \c__scontents_end_env_tl \exp_not:n {#2} }
562     }
563     {
564       \tl_if_blank:nF {#1}
565       { \__scontents_verb_processor_output:n {#1} }
566       \cs_set_protected:Npx \__scontents_ret:w
567       {
568         \__scontents_env_end_function:
569         \bool_lazy_or:nnF
570         { \tl_if_blank_p:n {#2} }
571         { \str_if_eq_p:ee {#2} { \c_percent_str } }
572         {
573           \str_if_eq:VnF \c__scontents_hidden_space_str {#2}
574           {
575             \msg_warning:nnnn { scontents } { rescanning-text }
576             {#2} { \tl_use:N \l__scontents_env_name_tl }
577           }
578           \__scontents_rescan_tokens:n {#2}
579         }
580       }
581       \char_set_active_eq:NN ^^M \__scontents_ret:w
582     }
583   }

```

```

584     ^^M
585   }
586   \cs_new:Npn \__scontents_env_end_function:
587   {
588     \__scontents_format_case:nnn
589     { \exp_not:N \end { \if_false: } \fi: }
590     { \exp_after:wN \exp_not:N \cs:w end }
591     { \exp_after:wN \exp_not:N \cs:w stop }
592     \tl_use:N \l__scontents_env_name_tl
593     \__scontents_format_case:nnn
594     { \if_false: { \fi: } }
595     { \cs_end: }
596     { \cs_end: }
597   }
598   \cs_new_protected:Npn \__scontents_file_write_stop:N #1
599   {
600     \bool_if:NT \l__scontents_writable_bool
601     { \iow_close:N \l__scontents_file_iow }
602     \use:x
603     {
604       \group_end:
605       \bool_if:NT \l__scontents_storing_bool
606       {
607         \tl_set:Nn \exp_not:N #1
608         { \exp_args:NV \__scontents_remove_leading_nl:n \l__scontents_every_line_env_tl }
609       }
610     }
611   }
612   \cs_new:Npn \__scontents_remove_leading_nl:n #1
613   {
614     \tl_if_head_is_N_type:nTF {#1}
615     {
616       \exp_args:Nf
617       \__scontents_remove_leading_nl:nn
618       { \tl_head:n {#1} } {#1}
619     }
620     { \exp_not:n {#1} }
621   }
622   \cs_new:Npn \__scontents_remove_leading_nl:nn #1 #2
623   {
624     \token_if_eq_meaning:NNTF ^^J #1
625     { \exp_not:o { \__scontents_remove_leading_nl:w #2 } }
626     { \exp_not:n {#2} }
627   }
628   \cs_new:Npn \__scontents_remove_leading_nl:w ^^J { }

```

(End of definition for `__scontents_file_tl_write_start:n` and others.)

`__scontents_verb_processor_output:n` The function `__scontents_verb_processor_output:n` does the output of each line read, to a token list and to a file, depending on the booleans `\l__scontents_writing_bool` and `\l__scontents_storing_bool`.

```

629   \cs_new_protected:Npn \__scontents_verb_processor_output:n #1
630   {
631     \bool_if:NT \l__scontents_writable_bool
632     { \iow_now:Nn \l__scontents_file_iow {#1} }
633     \bool_if:NT \l__scontents_storing_bool
634     { \tl_put_right:Nn \l__scontents_every_line_env_tl { ^^J #1 } }
635   }
636   \group_end:
637   \cs_generate_variant:Nn \__scontents_verb_processor_output:n { x }
638   \cs_generate_variant:Nn \__scontents_file_tl_write_start:n { V }

```

(End of definition for `__scontents_verb_processor_output:n`.)

`__scontents_analyse_nesting:n` `__scontents_analyse_nesting:w` `__scontents_nesting_decr:` `__scontents_use_none_delimit_by_q_stop:w` `__scontents_if_nested:TF` `__scontents_analyse_nesting:n` scans nested `\begin{scontents}` and steps a `\l__scontents_env_nesting_int` counter. The `__scontents_if_nested:` conditional tests if we're in a nested environment, and `__scontents_nesting_decr:` reduces the nesting level, if an `\end{scontents}` is found.

Multiple `\end{scontents}` in the same line are not supported...

```

639 \cs_new_protected:Npn \__scontents_analyse_nesting:n #1
640 {
641   \int_zero:N \l__scontents_tmpa_int
642   \__scontents_analyse_nesting_format:n {#1}
643   \int_compare:nNnT { \l__scontents_tmpa_int } > { 1 }
644     { \msg_warning:nn { scontents } { multiple-begin } }
645 }
646 \cs_new_protected:Npn \__scontents_nesting_incr:
647 {
648   \int_incr:N \l__scontents_env_nesting_int
649   \int_incr:N \l__scontents_tmpa_int
650 }
651 \cs_new_protected:Npn \__scontents_nesting_decr:
652 { \int_decr:N \l__scontents_env_nesting_int }
653 \prg_new_protected_conditional:Npnn \__scontents_if_nested: { TF }
654 {
655   \int_compare:nNnTF { \l__scontents_env_nesting_int } > { \c_zero_int }
656     { \prg_return_true: }
657     { \prg_return_false: }
658 }
659 \cs_new:Npn \__scontents_use_none_delimit_by_q_stop:w #1 \q__scontents_stop { }

```

In \LaTeX , environments start with `\begin{«env»}`, so checking if a string contains `\begin{scontents}` is straightforward. Since no `}` can appear inside `«env»`, then just a macro delimited by `}` is enough.

```

660 \use:x
661 {
662   \cs_new_protected:Npn \exp_not:N \__scontents_analyse_nesting_latex:w ##1
663     \c_backslash_str begin \c_left_brace_str ##2 \c_right_brace_str
664 } {
665   \__scontents_tl_if_head_is_q_mark:nTF {#2}
666     { \__scontents_use_none_delimit_by_q_stop:w }
667     {
668       \str_if_eq:VnT \l__scontents_env_name_tl {#2}
669         { \__scontents_nesting_incr: }
670         \__scontents_analyse_nesting_latex:w
671     }
672 }
673 \cs_new_protected:Npx \__scontents_analyse_nesting_latex:n #1
674 {
675   \__scontents_analyse_nesting_latex:w #1
676   \c_backslash_str begin
677     \c_left_brace_str \exp_not:N \q__scontents_mark \c_right_brace_str
678   \exp_not:N \q__scontents_stop
679 }

```

In other formats, however, we don't have an "end anchor" to delimit the environment name, so a delimited macro won't help. We have to search for the entire environment command (usually `\scontents` and `\startscontents`).

```

680 \cs_new_protected:Npn \__scontents_analyse_nesting_generic_process:nn #1 #2
681 {
682   \tl_if_head_is_N_type:nTF {#2}
683     {
684       \__scontents_tl_if_head_is_q_mark:nF {#2}
685       {
686         \__scontents_nesting_incr:
687         \__scontents_analyse_nesting_generic:w #2 \q__scontents_stop
688       }
689     }
690   { \__scontents_analyse_nesting_generic:w #2 \q__scontents_stop }
691 }
692 \cs_new_protected:Npn \__scontents_analyse_nesting_generic:nn #1 #2
693 {
694   \__scontents_define_generic_nesting_function:n {#1}
695   \use:e
696   {
697     \exp_not:N \__scontents_analyse_nesting_generic:w #2
698     \c_backslash_str #1 \tl_use:N \l__scontents_env_name_tl
699     \exp_not:N \q__scontents_mark \exp_not:N \q__scontents_stop
700   }
701 }
702 \cs_new_protected:Npn \__scontents_define_generic_nesting_function:n #1

```



```

703 {
704   \use:x
705   {
706     \cs_set_protected:Npn \exp_not:N \__scontents_analyse_nesting_generic:w ###1
707       \c_backslash_str #1 \tl_use:N \__scontents_env_name_tl
708       ###2 \exp_not:N \q__scontents_stop
709   } { \__scontents_analyse_nesting_generic_process:nn {##1} {##2} }
710 }
711 \</core>
712 \*loader)
713 \<latex>\cs_new_eq:NN \__scontents_analyse_nesting_format:n
714 \<latex> \__scontents_analyse_nesting_latex:n
715 \<!latex>\cs_new_protected:Npn \__scontents_analyse_nesting_format:n
716 \<plain> { \__scontents_analyse_nesting_generic:nn { } }
717 \<context> { \__scontents_analyse_nesting_generic:nn { start } }
718 \</loader>
719 \*core)

```

(End of definition for `__scontents_analyse_nesting:n` and others.)

12.10.6 Recording of the content in the sequence

`__scontents_finish_storing:NNN` Finishes the environment by optionally calling `__scontents_store_to_seq:` and then clearing the temporary token list.

```

720 \cs_new_protected:Npn \__scontents_finish_storing:NNN #1 #2 #3
721 {
722   \bool_if:NT \__scontents_storing_bool
723   {
724     \bool_if:NF \__scontents_forced_eol_bool
725     { \tl_put_right:Ne #1 { \c__scontents_hidden_space_str } }
726     \__scontents_store_to_seq:NN #1 #2
727     \bool_if:NT #3 { \__scontents_lastfrom_seq:V #2 }
728   }
729 }
730 \</core>

```

(End of definition for `__scontents_finish_storing:NNN`.)

12.11 Code for plain \TeX off verbatimsc

`\verbatimsc` In plain \TeX we emulate \LaTeX 's verbatim environment.

```

\endverbatimsc
\__scontents_verbatimsc_aux: 731 \*plain)
\__scontents_vobeyspaces: 732 \cs_new_protected:Npn \verbatimsc
  \__scontents_xverb: 733 {
\__scontents_nolig_list: 734   \group_begin:
  \__scontents_xobeysp: 735     \__scontents_verbatimsc_aux: \frenchspacing \__scontents_vobeyspaces:
  736     \__scontents_xverb:
  737   }
  738 \cs_new_protected:Npn \endverbatimsc
  739   { \group_end: }
  740 \cs_new_protected:Npn \__scontents_verbatimsc_aux:
  741   {
  742     \skip_vertical:N \parskip
  743     \dim_zero:N \parindent
  744     \skip_set:Nn \parfillskip { \opt plus 1fil }
  745     \skip_set:Nn \parskip { \opt plus\opt minus\opt }
  746     \tex_par:D
  747     \bool_set_false:N \__scontents_temp_bool
  748     \cs_set:Npn \par
  749     {
  750       \bool_if:NTF \__scontents_temp_bool
  751       {
  752         \mode_leave_vertical:
  753         \null
  754         \tex_par:D
  755         \penalty \interlinepenalty
  756       }
  757       {
  758         \bool_set_true:N \__scontents_temp_bool
  759         \mode_if_horizontal:T

```

```

760         { \tex_par:D \penalty \interlinepenalty }
761     }
762 }
763 \cs_set_eq:NN \do \char_set_catcode_other:N
764 \dospecials \obeylines
765 \tl_use:N \l__scontents_verb_font_tl
766 \cs_set_eq:NN \do \__scontents_do_noligs:N
767 \__scontents_nolig_list:
768 \tex_everypar:D \exp_after:wN
769   { \tex_the:D \tex_everypar:D \tex_unpenalty:D }
770 }
771 \cs_new_protected:Npn \__scontents_nolig_list:
772   { \do\` \do\<\do\>\do\,\do\'\do\ - }
773 \cs_new_protected:Npn \__scontents_vobeyspaces:
774   { \__scontents_set_active_eq:NN \ \__scontents_xobeysp: }
775 \cs_new_protected:Npn \__scontents_xobeysp:
776   { \mode_leave_vertical: \nobreak \ }
777 </plain>

```

(End of definition for `\verbatimsc` and others.)

`\dospecials` xparse also requires `ETEX`'s `\dospecials`. In case it doesn't exist (at the time `scontents` is loaded) we define `\dospecials` to use the `\l_char_special_seq`.

```

778 <!*latex>
779 \cs_if_exist:NF \dospecials
780   {
781     \cs_new:Npn \dospecials
782       { \seq_map_function:NN \l_char_special_seq \do }
783   }
784 </!*latex>

```

(End of definition for `\dospecials`.)

12.12 The command `\Scontents`

User command to *stored content*, adapted from code by Ulrich Diez in *Stringify input - \string on token list* and code by user *siracusa* in *Convert a macro from Latexze to expl3*

`__scontents_bsphack:` We emulate `\@bsphack` and `\@esphack` for plain `TEX`. This is necessary to prevent unwanted spaces when the `print-cmd` key is false.

```

785 <*core>
786 \cs_new_protected:Npn \__scontents_bsphack:
787   {
788     \scan_stop:
789     \mode_if_horizontal:T
790     {
791       \skip_set_eq:NN \l__scontents_save_skip \tex_lastskip:D
792       \int_set_eq:NN \l__scontents_save_sf_int \tex_spacefactor:D
793     }
794   }
795 \cs_new_protected:Npn \__scontents_esphack:
796   {
797     \scan_stop:
798     \mode_if_horizontal:T
799     {
800       \int_set_eq:NN \tex_spacefactor:D \l__scontents_save_sf_int
801       \dim_compare:nNnT { \l__scontents_save_skip } > { \c_zero_skip }
802       {
803         \skip_if_eq:nnT { \tex_lastskip:D } { \c_zero_skip }
804         {
805           \nobreak
806           \skip_horizontal:n { \c_zero_skip }
807         }
808       }
809       \tex_ignorespaces:D
810     }
811   }
812 </core>
813 <!*latex>
814 \cs_gset_eq:NN \__scontents_bsphack: \@bsphack

```

```

815 \cs_gset_eq:NN \__scontents_esphack: \@esphack
816 \</latex>

```

(End of definition for __scontents_bsphack: and __scontents_esphack:.)

The `\Scontents` command starts by parsing an optional argument to the function `__scontents_Scontents_internal:n` then delegates to `__scontents_verb_arg:w` or `__scontents_norm_arg:n` depending whether a star (*) argument is present.

```

\__scontents_Scontents_internal:n
\__scontents_norm_arg:n
\__scontents_verb_arg:w
817 \<loader>
818 \NewDocumentCommand \Scontents { !s !0{} }
819 { \__scontents_Scontents_internal:n {#1} {#2} }
820 \</loader>
821 \<core>
822 \cs_new_protected:Npn \__scontents_Scontents_internal:n #1 #2
823 {
824   \__scontents_bsphack:
825   \group_begin:
826     \tl_if_novalue:nF {#2}
827     { \keys_set:nn { scontents / Scontents } {#2} }
828     \char_set_catcode_active:n { 9 }
829     \bool_if:NTF #1
830     { \__scontents_verb_arg:w }
831     { \__scontents_norm_arg:n }
832 }

```

The function `__scontents_norm_arg:n` grabs a normal argument, adds it to the `seq` variable and optionally prints it.

```

833 \cs_new_protected:Npn \__scontents_norm_arg:n #1
834 {
835   \tl_set:Nn \l__scontents_temp_tl {#1}
836   \__scontents_Scontents_finish:
837 }

```

The function `__scontents_verb_arg:w` grabs a verbatim argument using `xparse`'s `+v` argument parser.

```

838 \</core>
839 \<loader>
840 \NewDocumentCommand \__scontents_verb_arg:w { +v }
841 { \__scontents_verb_arg_internal:n {#1} }
842 \</loader>
843 \<core>

```

(End of definition for `\Scontents` and others. This function is documented on page 5.)

The function `__scontents_verb_arg_internal:n` replace all `\^^M` and `\obeyedline` added by `+v` argument added in L^AT_EX release 2024-06-01 by `\^^J` then adds it to the `seq` variable. Here we will apply `\RenewDocumentCommand` since `\obeyedline` can be modified by the user and if so the code would return a low-level error.

```

844 \cs_new_protected:Npn \__scontents_verb_arg_internal:n #1
845 {
846   \tl_set:Nn \l__scontents_temp_tl {#1}
847   \cs_if_exist:NT \obeyedline
848   {
849     \RenewDocumentCommand \obeyedline { } { \iow_char:N \^^M }
850     \tl_replace_all:Nee \l__scontents_temp_tl { \obeyedline } { \iow_char:N \^^M }
851   }
852   \tl_replace_all:Nee \l__scontents_temp_tl { \iow_char:N \^^M } { \iow_char:N \^^J }
853   \__scontents_Scontents_finish:
854 }
855 \cs_new_protected:Npn \__scontents_Scontents_finish:
856 {
857   \__scontents_file_write_cmd:VV \l__scontents_fname_out_tl \l__scontents_temp_tl
858   \__scontents_finish_storing:NNN
859   \l__scontents_temp_tl
860   \l__scontents_name_seq_cmd_tl
861   \l__scontents_print_cmd_bool
862   \use:x
863   {
864     \group_end:

```

```

865   \bool_if:NF \l__scontents_print_cmd_bool { \l__scontents_espack: }
866   }
867 }
868 \cs_new_protected:Npn \l__scontents_file_write_cmd:nn #1#2
869 {
870   \l__scontents_file_if_writable:nT {#1}
871   {
872     \iow_open:Nn \l__scontents_file_iow {#1}
873     \iow_now:Nn \l__scontents_file_iow {#2}
874     \iow_close:N \l__scontents_file_iow
875   }
876 }
877 \cs_generate_variant:Nn \l__scontents_file_write_cmd:nn { VV }
878 \prg_new_protected_conditional:Npnn \l__scontents_file_if_writable:n #1 { T, F, TF }
879 {
880   \bool_if:NTF \l__scontents_writing_bool
881   {
882     \file_if_exist:nTF {#1}
883     {
884       \bool_if:NTF \l__scontents_overwrite_bool
885       {
886         \msg_warning:nne { scontents } { overwrite-file } {#1}
887         \prg_return_true:
888       }
889       {
890         \msg_warning:nne { scontents } { not-writing } {#1}
891         \prg_return_false:
892       }
893     }
894     {
895       \msg_warning:nne { scontents } { writing-file } {#1}
896       \prg_return_true:
897     }
898   }
899   { \prg_return_false: }
900 }

```

(End of definition for `\l__scontents_verb_arg_internal:n`, `\l__scontents_Scontents_finish:`, and `\l__scontents_file_write_cmd:nn`.)

12.13 The command `\getstored`

`\getstored` User command `\getstored` to extract *stored content* in seq (robust).

```

\l__scontents_getstored_internal:nn
901 </core>
902 <*loader>
903 \NewDocumentCommand \getstored { 0{-1} m }
904 { \l__scontents_getstored_internal:nn {#1} {#2} }
905 </loader>
906 <*core>
907 \cs_new_protected:Npn \l__scontents_getstored_internal:nn #1 #2
908 {
909   \group_begin:
910   \int_set:Nn \tex_newlinechar:D { ``^^J }
911   \l__scontents_rescan_tokens:x
912   {
913     \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
914     \l__scontents_getfrom_seq:nn {#1} {#2}
915   }
916 }
917 </core>

```

(End of definition for `\getstored` and `\l__scontents_getstored_internal:nn`. This function is documented on page 6.)

12.14 The command `\foreachsc`

`\foreachsc` User command `\foreachsc` to loop over *stored content* in seq.

```

\l__scontents_foreachsc_internal:nn
\l__scontents_foreach_add_body:n
918 <*loader>
919 \NewDocumentCommand \foreachsc { o m }
920 { \l__scontents_foreachsc_internal:nn {#1} {#2} }
921 </loader>

```

```

922 (*core)
923 \cs_new_protected:Npn \__scontents_foreachsc_internal:nn #1 #2
924 {
925   \group_begin:
926   \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / foreachsc } {#1} }
927   \tl_set:Nn \__scontents_foreach_name_seq_tl {#2}
928   \seq_clear:N \__scontents_foreach_print_seq
929   \bool_if:NF \__scontents_foreach_stop_bool
930   {
931     \int_set:Nn \__scontents_foreach_stop_int
932     { \seq_count:c { g__scontents_name_#2_seq } }
933   }
934   \int_step_function:nnnN
935   { \__scontents_foreach_start_int }
936   { \__scontents_foreach_step_int }
937   { \__scontents_foreach_stop_int }
938   \__scontents_foreach_add_body:n
939   \tl_gset:Ne \g__scontents_temp_tl
940   {
941     \exp_args:NNV \seq_use:Nn
942     \__scontents_foreach_print_seq \__scontents_foreach_sep_tl
943   }
944   \group_end:
945   \exp_after:wN \tl_gclear:N
946   \exp_after:wN \g__scontents_temp_tl
947   \g__scontents_temp_tl
948 }
949 \cs_new_protected:Npn \__scontents_foreach_add_body:n #1
950 {
951   \seq_put_right:Ne \__scontents_foreach_print_seq
952   {
953     \bool_if:NT \__scontents_foreach_before_bool
954     { \exp_not:V \__scontents_foreach_before_tl }
955     \bool_if:NTF \__scontents_foreach_wrapper_bool
956     { \__scontents_foreach_wrapper:n }
957     { \use:n }
958     { \getstored [#1] { \tl_use:N \__scontents_foreach_name_seq_tl } }
959     \bool_if:NT \__scontents_foreach_after_bool
960     { \exp_not:V \__scontents_foreach_after_tl }
961   }
962 }
963 </core>

```

(End of definition for `\foreachsc`, `__scontents_foreachsc_internal:nn`, and `__scontents_foreach_add_body:n`. This function is documented on page 6.)

12.15 The command `\typestored`

`\typestored` The `\typestored` commands fetches a buffer from memory, prints it to the log file, and then calls `__scontents_verb_print:N`.

```

\__scontents_typestored_internal:nn
\__scontents_verb_print:N
\__scontents_xverb:w
964 (*loader)
965 \NewDocumentCommand \typestored { o m }
966 { \__scontents_typestored_internal:nn {#1} {#2} }
967 </loader>
968 (*core)
969 \cs_new_protected:Npn \__scontents_typestored_internal:nn #1 #2
970 {
971   \__scontents_bsphack:
972   \group_begin:
973   \seq_clear:N \__scontents_seq_item_seq
974   \str_set:Ne \__scontents_cur_seq_name_str {#2}
975   \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
976   \seq_if_empty:NT \__scontents_seq_item_seq
977   { \seq_set_from_clist:Nn \__scontents_seq_item_seq { 1 } }
978   \tl_set:Ne \__scontents_temp_tl
979   { \__scontents_getfrom_seq:Nn \__scontents_seq_item_seq {#2} }
980   \__scontents_remove_trailing_eol:N \__scontents_temp_tl
981   \tl_replace_all:Nxn \__scontents_temp_tl \c__scontents_hidden_space_str { ^^J }
982   \tl_log:N \__scontents_temp_tl
983   \tl_if_empty:NF \__scontents_temp_tl
984   {

```

```

985         \bool_if:NT \l__scontents_print_aux_bool
986         {
987             \__scontents_verb_print:N \l__scontents_temp_tl
988         }
989     }
990     \__scontents_file_write_cmd:VV \l__scontents_fname_out_tl \l__scontents_temp_tl
991     \use:e
992     {
993         \group_end:
994         \bool_if:NF \l__scontents_print_aux_bool { \__scontents_espack: }
995     }
996 }

```

The `__scontents_verb_print:N` macro is defined with active carriage return (ASCII 13) characters to mimic an actual verbatim environment “on the loose”. The contents of the environment are placed in a `verbatimsc` environment and rescanned using `__scontents_rescan_tokens:x`.

```

997 \group_begin:
998   \char_set_catcode_active:N ^^M
999   \cs_new_protected:Npn \__scontents_verb_print:N #1
1000   {
1001     \tl_if_blank:VT #1
1002     { \msg_error:nnn { scontents } { empty-variable } {#1} }
1003     \cs_set_eq:NN \__scontents_verb_print_EOL: ^^M
1004     \cs_set_eq:NN ^^M \scan_stop:
1005     \cs_set_eq:cN { do@noligs } \__scontents_do_noligs:N
1006     \int_set:Nn \tex_newlinechar:D { `^^J }
1007     \__scontents_rescan_tokens:x
1008     {
1009       \__scontents_format_case:nnn
1010       { \exp_not:N \begin{verbatimsc} } % LaTeX
1011       { \verbatimsc } % Plain/Generic
1012       { \startverbatimsc } % ConTeXt
1013       ^^M
1014       \exp_not:V #1 ^^M
1015       \g__scontents_end_verbatimsc_tl
1016     }
1017     \cs_set_eq:NN ^^M \__scontents_verb_print_EOL:
1018   }
1019 \group_end:
1020 \cs_new_protected:Npn \__scontents_xverb:
1021 {
1022   \char_set_catcode_active:n { 9 }
1023   \char_set_active_eq:nN { 9 } \__scontents_tabs_to_spaces:
1024   \__scontents_xverb:w
1025 }
1026 </core>

```

(End of definition for `\typestored` and others. This function is documented on page 6.)

12.16 The command `\mergesc`

```

\mergesc
\__scontents_mergesc_output_cmd:nn
\__scontents_mergesc_internal:nn
\__scontents_mergesc_parse_list:n
\__scontents_remove_trailing_eol:N
\__scontents_remove_trailing_eol:w
\__scontents_parse_mergesc:nw
\__scontents_parse_mergesc_aux:nw
\__scontents_parse_mergesc_range:nw
1027 (*loader)
1028 \NewDocumentCommand \mergesc { o m }
1029 { \__scontents_mergesc_internal:nn {#1} {#2} }
1030 </loader>
1031 (*core)
1032 \keys_define:nn { scontents / mergesc }
1033 {
1034   , typestored .code:n =
1035     { \cs_set_eq:NN \__scontents_mergesc_output_cmd:nn \__scontents_typestored_internal:nn }
1036   , meanings .code:n =
1037     { \cs_set_eq:NN \__scontents_mergesc_output_cmd:nn \__scontents_meaningsc_internal:nn }
1038 }
1039 \cs_new_protected:Npn \__scontents_mergesc_output_cmd:nn #1 #2
1040 { \msg_error:nn { scontents } { mergesc-missing-cmd } }
1041 \msg_new:nnn { scontents } { mergesc-missing-cmd }
1042 { Missing-output~command~for~\iow_char:N\mergesc~\msg_line_context:. }
1043 \cs_new_protected:Npn \__scontents_mergesc_internal:nn #1 #2

```

```

1044 {
1045   \group_begin:
1046     \tl_clear:N \l__scontents_keys_tl
1047     \tl_if_novalue:nF {#1}
1048     {
1049       % Add print-cmd here :D
1050       \keys_define:nn { scontents / typemeaning }
1051       {
1052         print-cmd .bool_set:N = \l__scontents_print_aux_bool,
1053         print-cmd .initial:n = false,
1054         print-cmd .default:n = true,
1055       }
1056       \keys_set_known:nN { scontents / mergesc } {#1} \l__scontents_keys_tl
1057     }
1058     \seq_gclear:c { g__scontents_name_sc!internal_seq }
1059     \__scontents_mergesc_parse_list:n {#2}
1060     \exp_args:Nx \__scontents_mergesc_output_cmd:nn
1061     { 1-end, \exp_not:V \l__scontents_keys_tl } { sc!internal }
1062   \group_end:
1063 }

1064 \cs_new_protected:Npn \__scontents_mergesc_parse_list:n #1
1065 {
1066   \clist_map_inline:nn {#1} { \__scontents_parse_mergesc:nw ##1 \s__scontents_stop }
1067   \seq_gpop_right:cN { g__scontents_name_sc!internal_seq } \l__scontents_temp_tl
1068   \__scontents_remove_trailing_eol:N \l__scontents_temp_tl
1069   \seq_gput_right:cV { g__scontents_name_sc!internal_seq } \l__scontents_temp_tl
1070 }
1071 \cs_new_protected:Npx \__scontents_remove_trailing_eol:N #1
1072 {
1073   \exp_not:N \exp_after:wN \exp_not:N \__scontents_remove_trailing_eol:w
1074   #1 \s__scontents_stop \c__scontents_hidden_space_str \s__scontents_stop \s__scontents_mark #1
1075 }
1076 \use:e
1077 {
1078   \cs_new_protected:Npn \exp_not:N \__scontents_remove_trailing_eol:w #1
1079   \c__scontents_hidden_space_str \s__scontents_stop #2 \s__scontents_mark #3
1080 } {
1081   \tl_set:Ne #3
1082   {
1083     \tl_if_empty:nTF {#2}
1084     { \exp_not:o { \__scontents_use_delimit_by_s_stop:nw #1 } }
1085     { \exp_not:n {#1} }
1086   }
1087 }
1088 \cs_new_protected:Npn \__scontents_parse_mergesc:nw #1
1089 {
1090   \peek_remove_spaces:n
1091   {
1092     \peek_charcode:NTF [ % ]
1093     { \__scontents_parse_mergesc_aux:nw {#1} }
1094     { \__scontents_parse_mergesc_aux:nw {#1} [ 1-\seq_count:c { g__scontents_name_#1_seq } ] }
1095   }
1096 }
1097 \cs_new_protected:Npn \__scontents_parse_mergesc_aux:nw #1 [#2]
1098 {
1099   \seq_clear:N \l__scontents_seq_item_seq
1100   \clist_map_inline:nn {#2}
1101   { \__scontents_parse_mergesc_range:nw {#1} ##1 - \q__scontents_mark - \s__scontents_mark }
1102   \seq_map_inline:Nn \l__scontents_seq_item_seq
1103   {
1104     \seq_gput_right:ce { g__scontents_name_sc!internal_seq }
1105     { \seq_item:cn { g__scontents_name_#1_seq } {##1} }
1106   }
1107   \__scontents_use_none_delimit_by_s_stop:w
1108 }
1109 \cs_new_protected:Npn \__scontents_parse_mergesc_range:nw #1 #2 - #3 - #4 \s__scontents_mark
1110 {
1111   \cs_set_protected:Npn \__scontents_tmp:w ##1
1112   {
1113     \msg_error:nneee { scontents } { index-out-of-range }

```

```

1114         {##1} {#1} { \seq_count:c { g__scontents_name_#1_seq } }
1115     }
1116     \__scontents_range_parser:nxn {#2} {#3}
1117     { \seq_count:c { g__scontents_name_#1_seq } }
1118     { \__scontents_tmp:w }
1119 }
1120 </core>

```

(End of definition for `\mergesc` and others. This function is documented on page 6.)

12.17 The \TeX , \LaTeX and \ConTeXt `verbatimsc`

Finally the \TeX , \LaTeX and \ConTeXt version of `verbatimsc` environment is defined.

```

\endverbatimsc
\end{verbatimsc}
\stopverbatimsc
\startverbatimsc
1121 (*loader)
1122 (*!context)
1123 \use:e
1124 {
1125     \cs_new_protected:Npn \exp_not:N \__scontents_xverb:w
1126     #1 \__scontents_end_verbatimsc_tl
1127 <latex>     { #1 \exp_not:N \end{verbatimsc} }
1128 <plain>     { #1 \exp_not:N \endverbatimsc }
1129 <context>   { #1 \exp_not:N \stopverbatimsc }
1130 }
1131 </!context>

```

In \ConTeXt we use our own tool `\definetying`.

```

1132 <context>\definetying[verbatimsc]

```

(End of definition for `\endverbatimsc` and others.)

`__scontents_declare_instance:` To be compatible with *tagged* PDF we must define the environment `verbatimsc` in terms of the `xtemplate` module integrated into the \LaTeX kernel, this code is adapted directly from Mrs. Ulrike Fischer's answer to [New verbatim environment with block code \(tagged-pdf\)](#).

`verbatimsc`

```

1133 (*!latex)
1134 \cs_new_protected:Nn \__scontents_declare_instance:
1135 {
1136     \DeclareInstance{blockenv}{verbatimsc}{display}
1137     {
1138         env-name      = verbatimsc,
1139         tag-name      = verbatim,
1140         tag-class     = ,
1141         tagging-recipe = standard,
1142         inner-level-counter = ,
1143         level-increase = false,
1144         setup-code    = ,
1145         block-instance = displayblock,
1146         inner-instance = ,
1147         final-code    = \legacyverbatimsetup \tag_tool:n {paratag=codeline},
1148         para-flattened = true
1149     }
1150 }
1151 \NewDocumentEnvironment { verbatimsc } { }
1152 {
1153     \IfDocumentMetadataTF
1154     {
1155         \__scontents_declare_instance:
1156         \UseInstance{blockenv}{verbatimsc}{}
1157         \@setupverbinvisiblespace\frenchspacing\@vobeyspaces %% <--??
1158         \__scontents_xverb:
1159     }
1160     {
1161         \cs_set_eq:cN { @xverbatim } \__scontents_xverb:
1162         \verbatim
1163     }
1164 }

```

The macro `\endverbatim` in the second argument of the `verbatimsc` environment is only needed for compatibility with the `verbatim` package.

```

1165 {

```



```

1166 \IfDocumentMetadataTF
1167 {
1168   \endblockenv
1169 }
1170 { \endverbatim }
1171 }
1172 </latex>
1173 </loader>

```

(End of definition for `__scontents_declare_instance:` and `verbatimsc`. This function is documented on page 7.)

12.17.1 Some auxiliaries functions

`__scontents_tabs_to_spaces:` In a verbatim context the TAB character is made active and set equal to `__scontents_tabs_to_spaces:`, to produce as many spaces as the `width-tab` key was set to.

```

1174 <*core>
1175 \cs_new:Npn \__scontents_tabs_to_spaces:
1176 { \prg_replicate:nn { \l__scontents_tab_width_int } { ~ } }

```

(End of definition for `__scontents_tabs_to_spaces:.`)

`__scontents_do_noligs:N` `__scontents_do_noligs:N` is an alternative definition for $\text{\LaTeX} 2_{\epsilon}$'s `\do@noligs` which makes sure to not consume following space tokens. The $\text{\LaTeX} 2_{\epsilon}$ version ends with `\char`#1`, which leaves \TeX still looking for an *optional space*.

```

1177 \cs_new_protected:Npn \__scontents_do_noligs:N #1
1178 {
1179   \char_set_catcode_active:N #1
1180   \cs_set:cpe { __scontents_active_char_ \token_to_str:N #1 : }
1181   {
1182     \mode_leave_vertical:
1183     \tex_kern:D \c_zero_dim
1184     \tex_char:D ``\exp_not:N #1
1185   }
1186   \char_set_active_eq:Nc #1 { __scontents_active_char_ \token_to_str:N #1 : }
1187 }

```

(End of definition for `__scontents_do_noligs:N`.)

`__scontents_tl_if_head_is_q_mark:nTF` Tests if the head of the token list is `\q__scontents_mark`.

```

1188 \prg_new_protected_conditional:Npnn \__scontents_tl_if_head_is_q_mark:n #1
1189 { T, F, TF }
1190 {
1191   \exp_after:wN \if_meaning:w
1192   \exp_after:wN
1193   \q__scontents_mark \__scontents_use_i_delimit_by_s_stop:nw #1 ? \s__scontents_stop
1194   \prg_return_true:
1195   \else:
1196   \prg_return_false:
1197   \fi:
1198 }

```

(End of definition for `__scontents_tl_if_head_is_q_mark:nTF`.)

`__scontents_set_active_eq:NN` `__scontents_set_active_eq:NN` Shortcut definitions for common catcode changes. The `^^L` needs a special treatment in non- \LaTeX mode because in Plain \TeX it is an `\outer` token.

```

\__scontents_make_control_chars_active:
\__scontents_plain_disable_outer_par:
1199 \cs_new_protected:Npn \__scontents_set_active_eq:NN #1
1200 {
1201   \char_set_catcode_active:N #1
1202   \char_set_active_eq:NN #1
1203 }
1204 </core>
1205 <*loader>
1206 \group_begin:
1207 <plain> \char_set_catcode_active:n { ``* }
1208 \cs_new_protected:Npn \__scontents_plain_disable_outer_par:
1209 <*plain>
1210 {
1211   \group_begin:

```

```

1212     \char_set_lccode:nn { `}* } { `^^L }
1213     \tex_lowercase:D { \group_end:
1214     \tex_let:D * \scan_stop:
1215     }
1216   }
1217 </plain>
1218 <latex|context>   { }
1219 \group_end:
1220 </loader>
1221 <*core>
1222 \group_begin:
1223   \char_set_catcode_active:N \*
1224   \cs_new_protected:Npn \__scontents_make_control_chars_active:
1225   {
1226     \__scontents_plain_disable_outer_par:
1227     \__scontents_set_active_eq:NN \^^I \__scontents_tab:
1228     \__scontents_set_active_eq:NN \^^L \__scontents_par:
1229     \__scontents_set_active_eq:NN \^^M \__scontents_ret:w
1230   }
1231 \group_end:
1232 </core>

```

(End of definition for `__scontents_set_active_eq:NN`, `__scontents_make_control_chars_active:`, and `__scontents_plain_disable_outer_par:`.)

12.18 The command `\setupsc`

User command `\setupsc` to setup module.

`\setupsc` A user-level wrapper for `\keys_set:nn{ scontents }`.

```

1233 <*loader>
1234 \NewDocumentCommand \setupsc { +m }
1235 { \keys_set:nn { scontents } {#1} }

```

(End of definition for `\setupsc`. This function is documented on page 3.)

12.19 The command `\meaningsc`

`\meaningsc` User command `\meaningsc` to see content stored in seq.

```

\__scontents_meaningsc_internal:nn
\__scontents_meaningsc:n

```

```

1236 \NewDocumentCommand \meaningsc { o m }
1237 { \__scontents_meaningsc_internal:nn {#1} {#2} }
1238 </loader>
1239 <*core>
1240 \cs_new_protected:Npn \__scontents_meaningsc_internal:nn #1 #2
1241 {
1242   \group_begin:
1243     \seq_clear:N \l__scontents_seq_item_seq
1244     \str_set:Nx \l__scontents_cur_seq_name_str {#2}
1245     \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
1246     \seq_if_empty:NT \l__scontents_seq_item_seq
1247     { \seq_set_from_clist:Nn \l__scontents_seq_item_seq { 1 } }
1248     \__scontents_meaningsc:n {#2}
1249   \group_end:
1250 }
1251 \group_begin:
1252   \char_set_catcode_active:N \^^I
1253   \cs_new_protected:Npn \__scontents_meaningsc:n #1
1254   {
1255     \tl_set:Nx \l__scontents_temp_tl
1256     { \__scontents_getfrom_seq:Nn \l__scontents_seq_item_seq {#1} }
1257     \tl_replace_all:Nxn \l__scontents_temp_tl { \iow_char:N \^^J } { ~ }
1258     \tl_replace_all:Nxn \l__scontents_temp_tl \c__scontents_hidden_space_str { ~ }
1259     \tl_log:N \l__scontents_temp_tl
1260     \tl_use:N \l__scontents_verb_font_tl
1261     \tl_replace_all:Nxn \l__scontents_temp_tl { ^^I } { \__scontents_tabs_to_spaces: }
1262     \cs_replacement_spec:N \l__scontents_temp_tl
1263   }
1264 \group_end:
1265 </core>

```

(End of definition for `\meaningsc`, `__scontents_meaningsc_internal:nn`, and `__scontents_meaningsc:n`. This function is documented on page 7.)

12.20 The command `\countsc`

`\countsc` User command `\countsc` to count number of contents stored in seq.

```

1266 (*loader)
1267 \NewExpandableDocumentCommand \countsc { m }
1268   { \seq_count:c { g__scontents_name_#1_seq } }

```

(End of definition for `\countsc`. This function is documented on page 7.)

12.21 The command `\cleanseqsc`

`\cleanseqsc` A user command `\cleanseqsc` to clear (remove) a defined seq.

```

1269 \NewDocumentCommand \cleanseqsc { m }
1270   { \seq_gclear_new:c { g__scontents_name_#1_seq } }
1271 (/loader)

```

(End of definition for `\cleanseqsc`. This function is documented on page 7.)

12.22 Warning and error messages

Warning and error messages used throughout the package.

```

1272 (*core)
1273 \msg_new:nnn { scontents } { junk-after-begin }
1274   {
1275     Junk~characters~#1~\msg_line_context: :
1276     \\ \\
1277     #2
1278   }
1279 \msg_new:nnnn { scontents } { env-already-defined }
1280   { Environment~'#1'~already-defined! }
1281   {
1282     You~have~used~\newenvsc
1283     with~an~environment~that~already~has~a~definition. \\ \\
1284     The~existing~definition~of~'#1'~will~not~be~altered.
1285   }
1286 \msg_new:nnn { scontents } { empty-stored-content }
1287   { Empty~value~for~key~'getstored'~\msg_line_context:. }
1288 \msg_new:nnn { scontents } { empty-variable }
1289   { Variable~'#1'~empty~\msg_line_context:. }
1290 \msg_new:nnn { scontents } { overwrite-file }
1291   { Overwriting~file~'#1'. }
1292 \msg_new:nnn { scontents } { writing-file }
1293   { Writing~file~'#1'. }
1294 \msg_new:nnn { scontents } { not-writing }
1295   { File~'#1'~already~exists.~Not~writing. }
1296 \msg_new:nnn { scontents } { rescanning-text }
1297   { Rescanning~text~'#1'~after~\c_backslash_str end{#2}~\msg_line_context:. }
1298 \msg_new:nnn { scontents } { multiple-begin }
1299   { Multiple~\c_backslash_str begin{ \__scontents_env_name_tl }~\msg_line_context:. }
1300 \msg_new:nnn { scontents } { undefined-storage }
1301   { Storage~named~'#1'~is~not~defined. }
1302 \msg_new:nnn { scontents } { index-out-of-range }
1303   {
1304     \int_compare:nNnTF {#1} = { 0 }
1305     { Index~of~sequence~cannot~be~zero. }
1306     {
1307       Index~'#1'~out~of~range~for~'#2'.~
1308       \int_compare:nNnTF {#1} > { 0 }
1309       { Max = } { Min = - } #3.
1310     }
1311   }
1312 \msg_new:nnnn { scontents } { env-key-unknown }
1313   {
1314     The~key~'#1'~is~unknown~by~environment~
1315     '\__scontents_env_name_tl'~and~is~being~ignored.
1316   }

```

```

1317 {
1318   The~environment~'\l__scontents_env_name_tl'~does~not~have~a~key~called~'#1'.\\
1319   Check~that~you~have~spelled~the~key~name~correctly.
1320 }
1321 \msg_new:nnnn { scontents } { env-key-value-unknown }
1322 {
1323   The~key~'#1=#2'~is~unknown~by~environment~
1324   '\l__scontents_env_name_tl'~and~is~being~ignored.
1325 }
1326 {
1327   The~environment~'\l__scontents_env_name_tl'~does~not~have~a~key~called~'#1'.\\
1328   Check~that~you~have~spelled~the~key~name~correctly.
1329 }
1330 \msg_new:nnnn { scontents } { cmd-key-unknown }
1331 { The~key~'#1'~is~unknown~by~'\c_backslash_str Scontents'~and~is~being~ignored. }
1332 {
1333   The~command~'\c_backslash_str Scontents'~does~not~have~a~key~called~'#1'.\\
1334   Check~that~you~have~spelled~the~key~name~correctly.
1335 }
1336 \msg_new:nnnn { scontents } { cmd-key-value-unknown }
1337 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str Scontents'~and~is~being~ignored. }
1338 {
1339   The~command~'\c_backslash_str Scontents'~does~not~have~a~key~called~'#1'.\\
1340   Check~that~you~have~spelled~the~key~name~correctly.
1341 }
1342 \msg_new:nnnn { scontents } { for-key-unknown }
1343 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachsc'~and~is~being~ignored. }
1344 {
1345   The~command~'\c_backslash_str foreachsc'~does~not~have~a~key~called~'#1'.\\
1346   Check~that~you~have~spelled~the~key~name~correctly.
1347 }
1348 \msg_new:nnnn { scontents } { for-key-value-unknown }
1349 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachsc'~and~is~being~ignored. }
1350 {
1351   The~command~'\c_backslash_str foreachsc'~does~not~have~a~key~called~'#1'.\\
1352   Check~that~you~have~spelled~the~key~name~correctly.
1353 }
1354 \msg_new:nnnn { scontents } { type-key-unknown }
1355 { The~key~'#1'~is~unknown~and~is~being~ignored. }
1356 {
1357   This~command~does~not~have~a~key~called~'#1'.\\
1358   This~command~only~accepts~the~key~'width-tab'.
1359 }
1360 \msg_new:nnnn { scontents } { type-key-value-unknown }
1361 { The~key~'#1'~to~which~you~passed~'#2'~is~unknown~and~is~being~ignored. }
1362 {
1363   This~command~does~not~have~a~key~called~'#1'.\\
1364   This~command~only~accepts~the~key~'width-tab'.
1365 }
1366 \msg_new:nnn { scontents } { empty-environment }
1367 { environment~'#1'~empty~\msg_line_context:. }
1368 \msg_new:nnnn { scontents } { verbatim-newline }
1369 { Verbatim~argument~of~#1~ended~by~end~of~line. }
1370 {
1371   The~verbatim~argument~of~the~#1~cannot~contain~more~than~one~line,~
1372   but~the~end~
1373   of~the~current~line~has~been~reached.~You~may~have~forgotten~the~
1374   closing~delimiter.
1375   \\ \\
1376   LaTeX~will~ignore~'#2'.
1377 }
1378 \msg_new:nnnn { scontents } { verbatim-tokenized }
1379 { The~verbatim~#1~cannot~be~used~inside~an~argument. }
1380 {
1381   The~#1~takes~a~verbatim~argument.~
1382   It~may~not~appear~within~the~argument~of~another~function.~
1383   It~received~an~illegal~token \tl_if_empty:nF {#3} { ~'#3' } .
1384   \\ \\
1385   LaTeX~will~ignore~'#2'.
1386 }

```

12.23 Finish package

Finish package implementation.

```
1387 </core>  
1388 <plain|context>\ExplSyntaxOff
```

13 Index of Implementation

The italic numbers denote the pages where the corresponding entry is described, the numbers underlined and all others indicate the line on which they are implemented in the package code.

Symbols	
\'	772
*	1207, 1212, 1223
\,	772
\-	772
\<	772
\>	772
\\	34, 913, 1042, 1276, 1283, 1318, 1327, 1333, 1339, 1345, 1351, 1357, 1363, 1375, 1384
\`	772
B	
\beginngroup	60, 65
bool commands:	
\bool_if:NTF	600, 605, 631, 633, 722, 724, 727, 750, 829, 865, 880, 884, 929, 953, 955, 959, 985, 994
\bool_lazy_and:nnTF	342, 510
\bool_lazy_or:nnTF	388, 569
\bool_new:N	106, 163, 165, 167, 168, 170, 172, 174, 273
\bool_set_false:N	164, 169, 171, 173, 175, 202, 224, 279, 525, 747
\bool_set_true:N	166, 198, 203, 220, 225, 242, 247, 255, 263, 274, 280, 522, 758
C	
\catcode	61, 913
char commands:	
\char_set_active_eq:NN	581, 1186, 1202
\char_set_active_eq:nN	1023
\char_set_catcode:nn	113
\char_set_catcode_active:N	451, 480, 481, 482, 998, 1179, 1201, 1223, 1252
\char_set_catcode_active:n	828, 1022, 1207
\char_set_catcode_letter:N	111
\char_set_catcode_letter:n	528
\char_set_catcode_other:N	34, 36, 37, 527, 763
\char_set_lccode:nn	1212
\l_char_special_seq	34, 527, 782
\char_value_catcode:n	110
\cleanseqsc	7, 43, <u>1269</u>
clist commands:	
\clist_map_inline:nn	1066, 1100
\countsc	7, 43, <u>1266</u>
cs commands:	
\cs:w	421, 422, 590, 591
\cs_end:	421, 422, 595, 596
\cs_generate_variant:Nn	191, 332, 362, 402, 637, 638, 877
\cs_gset_eq:NN	814, 815
\cs_if_exist:NTF	24, 428, 429, 779, 847
\cs_new:Npe	192
\cs_new:Npn	54, 185, 186, 187, 193, 363, 372, 374, 376, 386, 539, 549, 586, 612, 622, 628, 659, 781, 1175
\cs_new_eq:NN	713
\cs_new_protected:Nn	1134
\cs_new_protected:Npn	190, 287, 289, 295, 297, 303, 309, 311, 313, 319, 325, 333, 354, 394, 403, 408, 410, 417, 433, 436, 441, 449, 454, 469, 483, 485, 495, 497, 507, 517, 598, 629, 639, 646, 651, 662, 680, 692, 702, 715, 720, 732, 738, 740, 771, 773, 775, 786, 795, 822, 833, 844, 855, 868, 907, 923, 949, 969, 999, 1020, 1039, 1043, 1064, 1078, 1088, 1097, 1109, 1125, 1177, 1199, 1208, 1224, 1240, 1253
\cs_new_protected:Npx	673, 1071
\cs_replacement_spec:N	1262
\cs_set:Npe	1180
\cs_set:Npn	543, 748
\cs_set_eq:NN	763, 766, 1003, 1004, 1005, 1017, 1035, 1037, 1161
\cs_set_protected:Npn	264, 706, 1111
\cs_set_protected:Npx	529, 566
\csname	64, 74
D	
\DeclareInstance	1136
\def	2, 3, 5, 6, 63, 66, 67, 75, 88
\definetying	1132
dim commands:	
\dim_compare:nNnTF	801
\dim_zero:N	743
\c_zero_dim	1183
\do	763, 766, 772, 782
\dospecials	764, <u>778</u>
E	
\else	85, 87
else commands:	
\else:	1195
\end	40, 589, 1127
\endblockenv	1168
\endcsname	64, 74
\endgroup	63, 66, 69, 82, 96, 913
\endinput	83, 97
\endlinechar	62
\endscontents	4, <u>460</u>
\endverbatim	1170
\endverbatimsc	41, <u>731</u> , <u>1121</u>
\end{verbatimsc}	<u>1121</u>
Environments:	
scontents	27, 28
\errhelp	70
\errmessage	71
exp commands:	
\exp_after:wN	590, 591, 768, 945, 946, 1073, 1191, 1192
\exp_args:Nf	367, 380, 616
\exp_args:Nnf	375
\exp_args:NNV	941
\exp_args:Nooo	419
\exp_args:NV	288, 296, 310, 312, 608
\exp_args:Nx	1060
\exp_args:Nxx	327
\exp_not:N	50, 531, 534, 543, 546, 589, 590, 591, 607, 662, 677, 678, 697, 699, 706, 708, 1010, 1073, 1078, 1125, 1127, 1128, 1129, 1184
\exp_not:n	328, 329, 406, 561, 620, 625, 626, 954, 960, 1014, 1061, 1084, 1085
\expandafter	64, 74
\ExplSyntaxOff	27, 1388
\ExplSyntaxOn	16

F

`\fi` 73, 99, 100
 fi commands:
 `\fi`: 589, 594, 1197
 file commands:
 `\file_if_exist:nTF` 882
 `\file_input:n` 53, 112
 `\file_input_stop:` 28
`\foreachsc` 6, 24, 25, 36, 918
`\frenchspacing` 735, 1157

G

`\getstored` 6, 36, 901, 958
 group commands:
 `\group_begin:` . 438, 479, 519, 734, 825, 909, 925, 972,
 997, 1045, 1206, 1211, 1222, 1242, 1251
 `\group_end:` . . 444, 604, 636, 739, 864, 944, 993, 1019,
 1062, 1213, 1219, 1231, 1249, 1264
 `\group_insert_after:N` 397, 398, 399, 400

I

if commands:
 `\if_false:` 589, 594
 `\if_meaning:w` 1191
`\IfDocumentMetadataTF` 1153, 1166
`\ifx` 64, 74, 86
`\input` 15
 int commands:
 `\int_abs:n` 390
 `\int_compare:nNnTF` 346, 643, 655, 1304, 1308
 `\int_compare_p:nNn` 389, 390
 `\int_decr:N` 652
 `\int_eval:n` 338, 375
 `\int_incr:N` 648, 649
 `\int_new:N` 105, 160, 162, 188
 `\int_set:Nn` 110, 256, 910, 931, 1006
 `\int_set_eq:NN` 792, 800
 `\int_step_function:nnnN` 528, 934
 `\int_step_inline:nnnn` 347, 348
 `\int_to_roman:n` 25, 337, 343, 344
 `\int_zero:N` 641
 `\c_zero_int` 655
`\interlinepenalty` 755, 760
 iow commands:
 `\iow_char:N` 849, 850, 852, 1042, 1257
 `\iow_close:N` 601, 874
 `\iow_log:n` 19
 `\iow_new:N` 156
 `\iow_now:Nn` 632, 873
 `\iow_open:Nn` 523, 872

K

keys commands:
 `\keys_define:nn` . . 116, 147, 195, 217, 239, 275, 1032,
 1050
 `\l_keys_key_str` 24, 288, 296, 310, 312
 `\keys_set:nn` . 42, 413, 475, 827, 926, 975, 1235, 1245
 `\keys_set_known:nnN` 1056

L

`\legacyverbatimsetup` 1147

M

`\meaningsc` 7, 24, 25, 42, 1236
`\mergesc` 6, 38, 1027

mode commands:

`\mode_if_horizontal:TF` 759, 789, 798
`\mode_leave_vertical:` 752, 776, 1182

msg commands:

`\msg_error:nn` 357, 1040
`\msg_error:nnn` 292, 300, 306, 323, 430, 1002
`\msg_error:nnnn` 293, 301, 307, 317, 501
`\msg_error:nnnnn` 1113
`\msg_expandable_error:nnn` 370, 384
`\msg_expandable_error:nnnnn` 391
`\msg_line_context:` 23, 1042, 1275, 1287, 1289, 1297,
 1299, 1367
`\msg_new:nnn` 1041, 1273, 1286, 1288, 1290, 1292, 1294,
 1296, 1298, 1300, 1302, 1366
`\msg_new:nnnn` 1279, 1312, 1321, 1330, 1336, 1342, 1348,
 1354, 1360, 1368, 1378
`\msg_set:nnn` 22
`\msg_warning:nn` 26, 644
`\msg_warning:nnn` 514, 886, 890, 895
`\msg_warning:nnnn` 575

N

`\NeedsTeXFormat` 8
`\NewDocumentCommand` 426, 465, 818, 840, 903, 919, 965, 1028,
 1234, 1236, 1269
`\NewDocumentEnvironment` 435, 1151
`\newenvsc` 5, 19, 27, 408, 461, 1282
`\NewExpandableDocumentCommand` 1267
`\next` 63, 66, 75, 88, 101
`\nobreak` 776, 805
`\null` 753

O

`\obeyedline` 847, 849, 850
`\obeylines` 764

P

`\PackageError` 67, 77, 90
 Packages:
 scontents 18, 19, 22
 xparse 28
`\par` 748
`\parfillskip` 744
`\parindent` 743
`\parskip` 742, 745
 peek commands:
 `\peek_charcode:NTF` 1092
 `\peek_remove_spaces:n` 1090
`\penalty` 755, 760
 prg commands:
 `\prg_generate_conditional_variant:Nnn` . . 194
 `\prg_new_protected_conditional:Npnn` . 653, 878,
 1188
 `\prg_replicate:nn` 1176
 `\prg_return_false:` 657, 891, 899, 1196
 `\prg_return_true:` 656, 887, 896, 1194
`\ProcessKeyOptions` 151
`\ProvidesExplPackage` 9

Q

quark commands:

`\quark_new:N` 180, 181

quark internal commands:

`\q_scontents_mark` 41, 180, 316, 677, 699, 1101, 1193
`\q_scontents_stop` 180, 534, 546, 659, 678, 687, 690,
 699, 708

R

\relax 64, 74
 \RenewDocumentCommand 849

S

scan commands:

\scan_new:N 182, 183
 \scan_stop: 788, 797, 1004, 1214

scan internal commands:

\s__scontents_mark [182](#), [316](#), [319](#), [1074](#), [1079](#), [1101](#),
[1109](#)
 \s__scontents_stop . [182](#), [185](#), [186](#), [187](#), [1066](#), [1074](#),
[1079](#), [1193](#)

\Scontents 5, [23](#), [25](#), [34](#), [817](#)

\scontents 4, [460](#)

scontents 4, [460](#)

scontents internal commands:

__scontents_analyse_nesting:n [31](#), [553](#), [639](#), [639](#)
 __scontents_analyse_nesting:w [639](#)
 __scontents_analyse_nesting_format:n [642](#), [713](#),
[715](#)
 __scontents_analyse_nesting_generic:nn . [692](#),
[716](#), [717](#)
 __scontents_analyse_nesting_generic:w . . [687](#),
[690](#), [697](#), [706](#)
 __scontents_analyse_nesting_generic_
 process:nn [680](#), [709](#)
 __scontents_analyse_nesting_latex:n [673](#), [714](#)
 __scontents_analyse_nesting_latex:w [662](#), [670](#),
[675](#)
 __scontents_append_contents:nn [26](#), [354](#), [354](#), [362](#),
[406](#)
 __scontents_bspack: . . [22](#), [785](#), [786](#), [814](#), [824](#), [971](#)
 __scontents_check_line_process:nn [29](#), [479](#), [491](#),
[493](#), [496](#), [497](#)
 \l__scontents_cur_seq_name_str . . [184](#), [322](#), [974](#),
[1244](#)
 __scontents_declare_instance: [1133](#), [1134](#), [1155](#)
 __scontents_define_generic_nesting_
 function:n [694](#), [702](#)
 __scontents_do_noligs:N [41](#), [766](#), [1005](#), [1177](#), [1177](#)
 \c__scontents_end_env_tl [19](#), [43](#), [532](#), [533](#), [544](#), [545](#),
[561](#)
 \g__scontents_end_verbatimsc_tl . . [18](#), [31](#), [1015](#),
[1126](#)
 __scontents_env_define:nnn . . . [27](#), [408](#), [420](#), [433](#)
 __scontents_env_end_function: [568](#), [586](#)
 __scontents_env_generic_begin: [27](#), [28](#), [415](#), [449](#),
[449](#)
 __scontents_env_generic_end: [27](#), [28](#), [418](#), [449](#), [454](#)
 \l__scontents_env_name_tl [19](#), [27](#), [43](#), [412](#), [502](#), [514](#),
[576](#), [592](#), [668](#), [698](#), [707](#), [1299](#), [1315](#), [1318](#), [1324](#), [1327](#)
 \l__scontents_env_nesting_int . . [21](#), [31](#), [160](#), [648](#),
[652](#), [655](#)
 __scontents_env_setting:nn . . . [27](#), [408](#), [408](#), [431](#)
 __scontents_espack: . . [22](#), [785](#), [795](#), [815](#), [865](#), [994](#)
 \l__scontents_every_line_env_tl [21](#), [154](#), [526](#), [608](#),
[634](#)
 __scontents_file_if_writable:n [878](#)
 __scontents_file_if_writable:nTF . . . [520](#), [870](#)
 \l__scontents_file_iow [21](#), [154](#), [523](#), [601](#), [632](#), [872](#),
[873](#), [874](#)
 __scontents_file_tl_write_start:n . [21](#), [29](#), [505](#),
[517](#), [517](#), [638](#)

__scontents_file_write_cmd:nn [844](#), [857](#), [868](#), [877](#),
[990](#)
 __scontents_file_write_stop:N [29](#), [509](#), [517](#), [598](#)
 __scontents_finish_storing:NNN . [457](#), [720](#), [720](#),
[858](#)
 \l__scontents_fname_out_tl . [21](#), [154](#), [199](#), [204](#), [221](#),
[226](#), [281](#), [505](#), [857](#), [990](#)
 \l__scontents_forced_eol_bool [132](#), [724](#)
 __scontents_foreach_add_body:n . [918](#), [938](#), [949](#)
 \l__scontents_foreach_after_bool . [168](#), [247](#), [959](#)
 \l__scontents_foreach_after_tl [21](#), [157](#), [248](#), [960](#)
 \l__scontents_foreach_before_bool [168](#), [242](#), [953](#)
 \l__scontents_foreach_before_tl [21](#), [157](#), [243](#), [954](#)
 \l__scontents_foreach_name_seq_tl . [21](#), [157](#), [927](#),
[958](#)
 \l__scontents_foreach_print_seq [22](#), [176](#), [928](#), [942](#),
[951](#)
 \l__scontents_foreach_sep_tl [268](#), [942](#)
 \l__scontents_foreach_start_int [251](#), [935](#)
 \l__scontents_foreach_step_int [259](#), [936](#)
 \l__scontents_foreach_stop_bool . [168](#), [255](#), [929](#)
 \l__scontents_foreach_stop_int [21](#), [160](#), [256](#), [931](#),
[937](#)
 __scontents_foreach_wrapper:n [265](#), [956](#)
 \l__scontents_foreach_wrapper_bool [168](#), [263](#), [955](#)
 __scontents_foreachsc_internal:nn [918](#), [920](#), [923](#)
 __scontents_format_case:nnn [54](#), [54](#), [588](#), [593](#), [1009](#)
 __scontents_getfrom_seq:Nn . [363](#), [363](#), [979](#), [1256](#)
 __scontents_getfrom_seq:nn . . . [26](#), [363](#), [376](#), [914](#)
 __scontents_getfrom_seq:nNn [367](#), [372](#)
 __scontents_getfrom_seq:nnn . [363](#), [375](#), [380](#), [386](#)
 __scontents_getfrom_seq_aux:nnn [373](#), [374](#)
 __scontents_getstored_internal:nn [901](#), [904](#), [907](#)
 __scontents_grab_optional:n [463](#), [466](#), [469](#)
 __scontents_grab_optional:w [28](#), [29](#), [463](#), [465](#), [490](#)
 \c__scontents_hidden_space_str [22](#), [178](#), [573](#), [725](#),
[981](#), [1074](#), [1079](#), [1258](#)
 __scontents_if_nested: [31](#), [653](#)
 __scontents_if_nested:TF [557](#), [639](#)
 \l__scontents_keys_tl [107](#), [1046](#), [1056](#), [1061](#)
 __scontents_lastfrom_seq:n [27](#), [394](#), [394](#), [402](#), [727](#)
 \l__scontents_macro_tmp_tl . [20](#), [102](#), [457](#), [509](#), [512](#)
 __scontents_make_control_chars_active: . [504](#),
[536](#), [1199](#), [1224](#)
 __scontents_meaningsc:n [1236](#), [1248](#), [1253](#)
 __scontents_meaningsc_internal:nn . [1037](#), [1236](#),
[1237](#), [1240](#)
 __scontents_mergesc_internal:nn . . . [1027](#), [1029](#),
[1043](#)
 __scontents_mergesc_output_cmd:nn . [1027](#), [1035](#),
[1037](#), [1039](#), [1060](#)
 __scontents_mergesc_parse_list:n . . [1027](#), [1059](#),
[1064](#)
 \l__scontents_name_seq_cmd_tl [121](#), [860](#)
 \l__scontents_name_seq_env_tl [118](#), [458](#)
 __scontents_nesting_decr: [31](#), [559](#), [639](#), [651](#)
 __scontents_nesting_incr: [646](#), [669](#), [686](#)
 __scontents_nolig_list: [731](#), [767](#), [771](#)
 __scontents_norm_arg:n [35](#), [817](#), [831](#), [833](#)
 __scontents_normalise_line_ends:N [28](#), [474](#), [483](#)
 \l__scontents_overwrite_bool [135](#), [884](#)
 __scontents_par: [192](#), [193](#), [1228](#)
 __scontents_parse_command_keys:n . [25](#), [237](#), [295](#),

295
 __scontents_parse_command_keys:nn 295, 296, 297
 __scontents_parse_environment_keys:n 24, 215,
287, 287
 __scontents_parse_environment_keys:nn . . 287,
288, 289
 __scontents_parse_foreach_keys:n . 25, 271, 303,
309
 __scontents_parse_foreach_keys:nn 303, 303, 310
 __scontents_parse_mergesc:nw . 1027, 1066, 1088
 __scontents_parse_mergesc_aux:nw . . 1027, 1093,
1094, 1097
 __scontents_parse_mergesc_range:nw 1027, 1101,
1109
 __scontents_parse_type_meaning_key:n 285, 311,
311
 __scontents_parse_type_meaning_key:nn . . 311,
312, 313
 __scontents_parse_type_meaning_range:w . 316,
319
 __scontents_parse_typemeaning_key:n 25
 __scontents_plain_disable_outer_par: . 1199,
1208, 1226
 \l__scontents_print_aux_bool . 273, 274, 985, 994,
1052
 \l__scontents_print_cmd_bool . . 27, 129, 861, 865
 \l__scontents_print_env_bool 27, 126, 458
 __scontents_range_parser:nnnn . . 321, 325, 332,
1116
 __scontents_range_parser_aux:nnn . . . 327, 333
 __scontents_remove_leading_nl:n . . 30, 517, 608,
612
 __scontents_remove_leading_nl:nn . . . 617, 622
 __scontents_remove_leading_nl:w . 517, 625, 628
 __scontents_remove_trailing_eol:N . 980, 1027,
1068, 1071
 __scontents_remove_trailing_eol:w . 1027, 1073,
1078
 __scontents_rescan_tokens:n 38, 24, 190, 190, 191,
397, 578, 911, 1007
 __scontents_ret:w . . . 29, 529, 537, 566, 581, 1229
 \l__scontents_save_sf_int 188, 792, 800
 \l__scontents_save_skip 188, 791, 801
 __scontents_Scontents_finish: 836, 844, 853, 855
 __scontents_Scontents_internal:nn 35, 817, 819,
822
 \l__scontents_seq_item_seq . 21, 161, 338, 349, 973,
976, 977, 979, 1099, 1102, 1243, 1246, 1247, 1256
 __scontents_set_active_eq:NN . 774, 1199, 1199,
1227, 1228, 1229
 __scontents_setup_verb_processor: 27, 414, 517,
539
 __scontents_start_after_option:w . 29, 477, 479,
495
 __scontents_start_environment:w . 452, 479, 485
 __scontents_stop_environment: . . 456, 479, 507
 __scontents_store_to_seq: 33
 __scontents_store_to_seq:NN . . 27, 403, 403, 726
 \l__scontents_storing_bool . 21, 31, 163, 202, 224,
279, 511, 605, 633, 722
 __scontents_tab: 192, 192, 1227
 \l__scontents_tab_width_int 138, 1176
 __scontents_tabs_to_spaces: 41, 1023, 1174, 1175,
1261
 \l__scontents_temp_bool . . . 20, 102, 747, 750, 758
 \g__scontents_temp_tl . 20, 102, 396, 398, 400, 939,
946, 947
 \l__scontents_temp_tl . 20, 102, 473, 474, 475, 835,
846, 850, 852, 857, 859, 978, 980, 981, 982, 983, 987,
990, 1067, 1068, 1069, 1255, 1257, 1258, 1259, 1261,
1262
 __scontents_tl_if_head_is_q_mark:n 1188
 __scontents_tl_if_head_is_q_mark:nTF 335, 665,
684, 1188
 __scontents_tmp:w 1111, 1118
 \l__scontents_tmpa_int 20, 102, 110, 113, 641, 643,
649
 __scontents_tpestored_internal:nn . 964, 966,
969, 1035
 __scontents_use_delimit_by_s_stop:nw 185, 1084
 __scontents_use_i_delimit_by_s_stop:nw . 185,
186, 1193
 __scontents_use_none_delimit_by_q_stop:w 639,
659, 666
 __scontents_use_none_delimit_by_s_stop:w 185,
187, 1107
 __scontents_verb_arg:w 35, 817, 830, 840
 __scontents_verb_arg_internal:n . . 35, 841, 844,
844
 \l__scontents_verb_font_tl 124, 765, 1260
 __scontents_verb_print:N . . . 37, 38, 964, 987, 999
 __scontents_verb_print_EOL: 1003, 1017
 __scontents_verb_processor_iterate:nnn . 517,
547, 549
 __scontents_verb_processor_iterate:w 517, 531,
543
 __scontents_verb_processor_output:n . 31, 554,
560, 565, 629, 629, 637
 __scontents_verbatimsc_aux: 731, 735, 740
 __scontents_vobeyspaces: 731, 735, 773
 \l__scontents_writable_bool 163, 522, 525, 600, 631
 \l__scontents_writing_bool . 21, 31, 163, 198, 203,
220, 225, 280, 880
 __scontents_xobeysp: 731, 774, 775
 __scontents_xverb: 731, 736, 1020, 1158, 1161
 __scontents_xverb:w 964, 1024, 1125
 \Scontents* 25
 \ScontentsCoreFileDate 3, 86
 \ScontentsFileDate 2, 10, 17, 86
 \ScontentsFileDescription 6, 10, 18
 \ScontentsFileVersion 5, 10, 13, 18
 seq commands:
 \seq_clear:N 928, 973, 1099, 1243
 \seq_count:N 322, 368, 381, 932, 1094, 1114, 1117, 1268
 \seq_gclear:N 1058
 \seq_gclear_new:N 1270
 \seq_gpop_right:NN 1067
 \seq_gput_right:Nn 360, 1069, 1104
 \seq_if_empty:NTF 976, 1246
 \seq_if_exist:NTF 358, 365, 378
 \seq_item:Nn 392, 396, 1105
 \seq_map_function:NN 527, 782
 \seq_map_inline:Nn 1102
 \seq_map_tokens:Nn 373
 \seq_new:N 161, 176, 177, 359
 \seq_put_right:Nn 338, 349, 951
 \seq_set_from_clist:Nn 977, 1247
 \seq_use:Nn 941

<code>\setupsc</code>	3, 42, 1233	tl commands:	
skip commands:		<code>\c_space_tl</code>	192
<code>\skip_horizontal:n</code>	806	<code>\tl_clear:N</code>	526, 1046
<code>\skip_if_eq:nnTF</code>	803	<code>\tl_const:Nn</code>	44
<code>\skip_new:N</code>	189	<code>\tl_gc_clear:N</code>	399, 945
<code>\skip_set:Nn</code>	744, 745	<code>\tl_gset:Nn</code>	17, 396, 939
<code>\skip_set_eq:NN</code>	791	<code>\tl_gset_rescan:Nnn</code>	32
<code>\skip_vertical:N</code>	742	<code>\tl_head:n</code>	489, 618
<code>\c_zero_skip</code>	801, 803, 806	<code>\tl_if_blank:nTF</code> 291, 299, 305, 315, 356, 499, 551, 564, 1001	
<code>\space</code>	17, 18	<code>\tl_if_blank_p:n</code>	570
<code>\startscontents</code>	4, 460	<code>\tl_if_empty:n</code>	194
<code>\startverbatimsc</code>	1012, 1121	<code>\tl_if_empty:NTF</code>	983
<code>\stopscontents</code>	4, 460	<code>\tl_if_empty:nTF</code>	194, 337, 1083, 1383
<code>\stopverbatimsc</code>	42, 1121	<code>\tl_if_empty_p:N</code>	512
str commands:		<code>\tl_if_empty_p:n</code>	343, 344
<code>\c_backslash_str</code> .. 46, 502, 663, 676, 698, 707, 1297, 1299, 1331, 1333, 1337, 1339, 1343, 1345, 1349, 1351		<code>\tl_if_head_is_N_type:nTF</code>	487, 614, 682
<code>\c_circumflex_str</code>	179	<code>\tl_if_novalue:nTF</code> .. 471, 826, 926, 975, 1047, 1245	
<code>\c_left_brace_str</code>	49, 663, 677	<code>\tl_log:N</code>	405, 982, 1259
<code>\c_percent_str</code>	179, 571	<code>\tl_new:N</code> 31, 43, 102, 103, 104, 107, 154, 155, 157, 158, 159	
<code>\c_right_brace_str</code>	51, 663, 677	<code>\tl_put_right:Nn</code>	634, 725
<code>\str_const:Nn</code>	178	<code>\tl_replace_all:Nnn</code> .. 484, 850, 852, 981, 1257, 1258, 1261	
<code>\str_if_eq:nnTF</code>	328, 329, 489, 573, 668	<code>\tl_rescan:nn</code>	22
<code>\str_if_eq_p:nn</code>	571	<code>\tl_set:Nn</code> 199, 204, 221, 226, 243, 248, 281, 412, 473, 607, 835, 846, 927, 978, 1081, 1255	
<code>\str_new:N</code>	184	<code>\tl_to_str:n</code>	420
<code>\str_set:Nn</code>	974, 1244	<code>\tl_use:N</code>	576, 592, 698, 707, 765, 958, 1260
		token commands:	
		<code>\token_if_eq_meaning:NNTF</code>	624
		<code>\token_to_str:N</code>	1180, 1186
		<code>\tt</code>	149
		<code>\ttfamily</code>	148
		<code>\typestored</code>	6, 24, 25, 37, 964
		U	
		<code>\unprotect</code>	14
		use commands:	
		<code>\use:N</code>	19
		<code>\use:n</code> 375, 541, 602, 660, 695, 704, 862, 957, 991, 1076, 1123	
		<code>\UseInstance</code>	1156
		V	
		<code>\verbatim</code>	1162
		<code>\verbatimsc</code>	731, 1011
		<code>verbatimsc</code>	7, 1133
		W	
		<code>\writestatus</code>	13
T			
tag commands:			
<code>\tag_tool:n</code>	1147		
TeX and \LaTeX commands:			
<code>\@</code>	110, 111, 113		
<code>\@bsphack</code>	814		
<code>\@esphack</code>	815		
<code>\@setupverbinvisiblespace</code>	1157		
<code>\@vobeyspaces</code>	1157		
tex commands:			
<code>\tex_char:D</code>	1184		
<code>\tex_everypar:D</code>	768, 769		
<code>\tex_ignorespaces:D</code>	808		
<code>\tex_kern:D</code>	1183		
<code>\tex_lastskip:D</code>	791, 803		
<code>\tex_let:D</code>	1214		
<code>\tex_lowercase:D</code>	1213		
<code>\tex_newlinechar:D</code>	910, 1006		
<code>\tex_par:D</code>	746, 754, 760		
<code>\tex_scantokens:D</code>	22, 190		
<code>\tex_spacefactor:D</code>	792, 800		
<code>\tex_the:D</code>	769		
<code>\tex_unpenalty:D</code>	769		