

# A L<sup>A</sup>T<sub>E</sub>X Package for Typesetting Information Maps\*

Gerd Neugebauer  
Mainzer Str. 8  
56321 Rhens (Germany)  
Net: `gerd.neugebauer@sdm.de`

Documentation date: 2000/03/01

## Abstract

The Information Mapping<sup>®</sup> method provides a different methodology for structuring and presenting information. It claims to be useful for readers who are more concerned about finding the right information than reading the document as a whole. Thus short, highly structured, and context free pieces of information are used.

`limap.dtx` provides a L<sup>A</sup>T<sub>E</sub>X style and a L<sup>A</sup>T<sub>E</sub>X class. The style contains definitions to typeset maps and blocks according to the Information Mapping method. The class provides all definitions to typeset a whole document.

---

\*This file documents `limap.dtx` version 1.2 as of 2000/03/01.

# Contents

<b>1</b>	<b>Motivation</b>	<b>3</b>
<b>2</b>	<b>The User Interface</b>	<b>3</b>
2.1	Package and Class Options . . . . .	3
2.2	Macros and Environments . . . . .	4
2.3	The Configuration Options . . . . .	5
2.4	Changing or Adding Language Specific Settings . . . . .	5
2.5	The Configuration File . . . . .	6
<b>3</b>	<b>The Documentation Driver</b>	<b>7</b>
3.1	The Version Information . . . . .	7
3.2	Producing the Documentation . . . . .	7
<b>4</b>	<b>The Implementation</b>	<b>8</b>
4.1	Preliminaries and Option Processing . . . . .	8
4.1.1	The Package/Class Declarations . . . . .	8
4.1.2	Language Specific Declarations . . . . .	8
4.1.3	Determining the Appropriate Base Class . . . . .	9
4.1.4	Loading Required Packages . . . . .	10
4.2	Layout Parameters . . . . .	10
4.3	Adaptable Macros . . . . .	11
4.4	Language Specific Macros . . . . .	12
4.5	Internal Macros, Lengths, and Counters . . . . .	13
4.6	Typesetting a Map . . . . .	15
4.7	Typesetting a Block . . . . .	16
4.8	Typesetting a Table of Contents . . . . .	16
4.9	Typesetting a Title Page . . . . .	17
4.10	Final Actions . . . . .	18

# 1 Motivation

The information mapping method provides a methodology to structure information in a special way. The aim is to help a reader who uses the document to search for relevant information instead of consuming it from start to end. The information mapping method also claims to raise the productivity of writers.

This document does not include an introduction to the information mapping method. The reader is referred to other documents. Maybe an accompanying document is distributed with this package.

To support the information mapping method several L<sup>A</sup>T<sub>E</sub>X macros and environments are needed which provide a logical description of the relevant concepts. Those macros are provided in the package and class file.

This package provides both a class file as well as a package. The package contains the definitions of maps, blocks, and others. They can be used together with any base class.

The class load a base class and does some other useful stuff. The base class can be determined with two class options.

## 2 The User Interface

The main part of the user interface is inherited from L<sup>A</sup>T<sub>E</sub>X. The major differences are the sectioning commands which are made obsolete in parts by the information mapping method.

### 2.1 Package and Class Options

The package and the class can take several options to influence the behaviour of the package or class.

First, we describe the settings influencing the language specific settings. They do not make provisions to use the appropriate hyphenation patterns. They just arrange things such that the internally used texts are displayed in the chosen language.

**austrian** Activate the language specific text fragments for the austrian language (in fact German with one minor modification).

**german** Activate the language specific text fragments for the german language.

**french** Activate the language specific text fragments for the french language.

**english** Activate the language specific text fragments for the english language.

**USenglish** Activate the language specific text fragments for the american language.

The class has two additional options to determine the base class to be used. The first option is the variant. It can take the following values:

**base** Use the base set of classes. This is the default.

**koma** Use the set of classes from koma-script.

The second option is the class type. It determines which kind of document to typeset. It can take the following values:

**book** Typeset a book type document.

**report** Typeset a report type document.

**article** Typeset a article type document.

**letter** Typeset a letter type document.

The following table shows which base classes are loaded according to the given values:

	<i>base</i>	<i>koma</i>
<i>book</i>	book	scrbook
<i>report</i>	report	scrreprt
<i>article</i>	article	scrartcl
<i>letter</i>	letter	scrlettr

Any option not processed by limap.cls is passed to the base class used. Thus it is possible to customize those classes any further.

## 2.2 Macros and Environments

**Block** The environment **Block** can be used to typeset an IMAP block. It takes one argument which is the block title.

This environment is active inside a map only. It can't easily be redefined.

**\Block** The macro **\Block** can be used to typeset an IMAP block. It takes one argument which is the block title.

This is a shorthand for denoting a block. The end mark can be omitted if you use the macro instead of the environment. Nevertheless this is deprecated.

**\WideBlock** The macro **\WideBlock** can be used to typeset a piece of information on the whole page width. It is normally used after an initiating block containing the title of the whole construction.

The macro **\WideBlock** takes one argument which contains the material to span the whole page width.

**Map** The environment **Map** can be used to typeset an IMAP map.

**\MapTableOfContents** The macro **\MapTableOfContents** can be used to typeset the table of contents for a map. This table of contents includes all submaps of the map it is contained in—not recursively but only one level deeper.

**\maketitle** The macro **\maketitle** is redefined by the class to show a new appearance on a title page. It can be used as in the standard classes.

## 2.3 The Configuration Options

The configuration options are settings which can be easily adapted to suit your personal needs. Nevertheless it is strongly recommended that you touch them only if you are really knowing what you are doing and when you are willing to face the consequences.

<code>\MapRuleWidth</code>	The macro <code>\MapRuleWidth</code> determines the width of the rules drawn between blocks.
<code>\MapFont</code>	The macro <code>\MapFont</code> determines the font changing command to be used when starting a new map.
<code>\MapTitleSize</code>	The macro <code>\MapTitleSize</code> determines the size changing command to be used when typesetting the title of a map.
<code>\MapTitleContinuedSize</code>	This macro determines the font changing command to be used for typesetting the additional text after titles on followup pages of multipage maps.
<code>\MapParskip</code>	The macro <code>\MapParskip</code> determines the distance of the text from the separating rules.
<code>\MapTitlefraction</code>	The macro <code>\MapTitlefraction</code> determines the part of the page width devoted to the title area. It is a fraction in the range from 0 to 1.
<code>\MapTextfraction</code>	This macro determines the part of the page width devoted to the text area. It is a fraction in the range from 0 to 1. <code>\MapTitlefraction</code> and <code>\MapTextfraction</code> should add up to something less or equal to 1. Otherwise you will get some “overfull hbox” messages.
<code>\MapNewpage</code>	The macro <code>\MapNewpage</code> is expanded whenever a new page is required between maps. Thus it can be used to suppress the newpages by <code>\leting</code> it to <code>\relax</code> . Note that this is not in the spirit of the Information Mapping method.
<code>\MapTOC</code>	The macro <code>\MapTOC</code> is expanded to generate the entry in the table of contents. It can be redefined to allow another behaviour.

## 2.4 Changing or Adding Language Specific Settings

Several strings are used automatically by the current class or package. Default values for several languages are hardwired in the implementation. Nevertheless it is possible to change those language specific settings.

If you create settings for a new language it is highly recommended to contact the author to integrate them into the default distribution.

The following macros can be redefined in the preamble after the package or class has been loaded to reset the language specific text.

<code>\MapContinued</code>	The macro <code>\MapContinued</code> contains the text appearing at the end of map which are continued on the next page.
<code>\MapContinuing</code>	The macro <code>\MapContinuing</code> contains the text appearing at the beginning of map which are continued from the previous page. It is typeset after the map title.
<code>\MapTOCname</code>	The macro <code>\MapTOCname</code> contains the text of the heading in table of contents of maps for the column of map titles.
<code>\MapTOCpage</code>	The macro <code>\MapTOCpage</code> contains the text of the heading in table of contents of maps for the column of page numbers.

If you want to provide a new language *lang* you can define the macro `\IMAP@SelectLanguage@lang` which redefines the macros given above. This definition has to be present before the package is loaded.

Note that the macro name contains a @ character. Thus the definition should be made in a package of its own.

## 2.5 The Configuration File

When the class or package is loaded as a last action a configuration file is loaded if it can be found. The name of the configuration file is `limap.cfg`. This file can contain redefinitions of the several macros to adjust the behaviour of `limap` on a per directory, per user or per installation base.

Note that some settings are activated before the configuration file is loaded. Thus some settings may not have any effect at all.

---

*To be completed*

---

### 3 The Documentation Driver

The documentation driver is necessary to provide a self documenting dtx file. With this construction the dtx file can be run through L<sup>A</sup>T<sub>E</sub>X to produce the documentation.

#### 3.1 The Version Information

`\LIMAP@RCS` The macro `\LIMAP@RCS` is used to parse rcs information. The second word enclosed in spaces is preserved. The other parts are ignored.

```
1 \def\LIMAP@RCS#1: #2 #3#{#2}
```

Now the usual macros are filled. Some of the informations are taken from strings automatically managed by rcs.

`\filename` `\filename` is the name of the dtx file.

```
2 \def\filename{limap.dtx}
```

`\fileversion` `\fileversion` is the version number of the dtx file.

```
3 \xdef\fileversion{\LIMAP@RCS$Revision: 1.2 $}
```

`\filedate` `\filedate` is the change date of the dtx file.

```
4 \xdef\filedate{\LIMAP@RCS$Date: 2000/03/01 20:11:42 $}
```

`\docversion` `\docversion` is version number of the documentation. It is identical to the version number of the dtx file.

```
5 \let\docversion=\fileversion
```

`\docdate` `\docdate` is change date of the documentation. It is identical to the change date of the dtx file.

```
6 \let\docdate=\filedate
```

#### 3.2 Producing the Documentation

The driver section contains a complete L<sup>A</sup>T<sub>E</sub>X document which loads the dtx file. The special class `ltxdoc` is used and some arrangements are made for this purpose.

```
7 ⟨*driver⟩
8 \documentclass{ltxdoc}
9 \RequirePackage{textcomp}
10 \InputIfFileExists{limap.dcf}{-}{}
11 \RecordChanges
12 \EnableCrossrefs
13 \CodelineIndex
14 \begin{document}
15 \DeleteShortVerb{|}
16 \DocInput{\filename}
```

```

17 \newpage
18 \PrintChanges
19 \newpage
20 \setcounter{IndexColumns}{2}
21 \PrintIndex
22 \end{document}
23 </driver>

```

## 4 The Implementation

The rest of the document describes the implementation. Usually it is not meant for the casual user. Nevertheless it might be fruitful for those searching for inspiration or for tricks when using this class or style.

### 4.1 Preliminaries and Option Processing

First of all we request a descent version of L<sup>A</sup>T<sub>E</sub>X to be used.

```
24 \NeedsTeXFormat{LaTeX2e}
```

#### 4.1.1 The Package/Class Declarations

When the package is generated, the package identification is included.

```

25 <*package>
26 \ProvidesPackage{limap}[\filedate\space gene]
27 </package>

```

When the class is generated, the class identification is included.

```

28 <*class>
29 \ProvidesClass{limap}[\filedate\space gene]
30 </class>

```

#### 4.1.2 Language Specific Declarations

`\LIMAP@Language` The macro `\IMAP@Language` determines the language to be used for several small text fragments to be inserted at certain places. It is redefined by package/class options and evaluated at the end to activate the selected settings.

```

31 \providecommand\LIMAP@Language{english}

32 \DeclareOption{austrian}{\renewcommand\LIMAP@Language{austrian}}
33 \DeclareOption{german}{\renewcommand\LIMAP@Language{german}}
34 \DeclareOption{french}{\renewcommand\LIMAP@Language{french}}
35 \DeclareOption{english}{\renewcommand\LIMAP@Language{english}}
36 \DeclareOption{USenglish}{\renewcommand\LIMAP@Language{USenglish}}

```

`\ifLIMAP@strict` The boolean `\ifLIMAP@strict` determines if the lower sectioning macros should be disabled in the class.

```

37 \newif\ifLIMAP@strict \LIMAP@stricttrue
38 \DeclareOption{nonstrict}{\LIMAP@strictfalse}

```

### 4.1.3 Determining the Appropriate Base Class

39  $\langle$ \*class $\rangle$

$\backslash$ LIMAP@ClassType The macro  $\backslash$ LIMAP@ClassType determines the type of the class to be used. Usually it can take the values `book`, `report`, `article`, and `letter` (for completeness). This macro is redefined when the options of the class are evaluated. Finally this macro helps to select the appropriate base class.

```
40 \providecommand\LIMAP@ClassType{report}

41 \DeclareOption{book}{\renewcommand\LIMAP@ClassType{book}}
42 \DeclareOption{report}{\renewcommand\LIMAP@ClassType{report}}
43 \DeclareOption{article}{\renewcommand\LIMAP@ClassType{article}}
44 \DeclareOption{letter}{\renewcommand\LIMAP@ClassType{letter}}
```

$\backslash$ LIMAP@Variant The macro  $\backslash$ LIMAP@Variant determines the variant of the class to be used. Usually it can take the values `base` and `koma`. This macro is redefined when the options of the class are evaluated. Finally this macro helps to select the appropriate base class.

```
45 \providecommand\LIMAP@Variant{base}

46 \DeclareOption{koma}{\renewcommand\LIMAP@Variant{koma}}
47 \DeclareOption{base}{\renewcommand\LIMAP@Variant{base}}
```

Define a mapping between the variant and class type to the class name to be used.

```
48 \newcommand\LIMAP@Class@base@article{article}
49 \newcommand\LIMAP@Class@base@report{report}
50 \newcommand\LIMAP@Class@base@book{book}
51 \newcommand\LIMAP@Class@base@letter{letter}
52 \newcommand\LIMAP@Class@koma@article{scrartcl}
53 \newcommand\LIMAP@Class@koma@report{scrreprt}
54 \newcommand\LIMAP@Class@koma@book{scrbook}
55 \newcommand\LIMAP@Class@koma@letter{scrletter}

56 \DeclareOption*{\PassOptionsToClass{\CurrentOption}{%
57   \csname LIMAP@Class@\LIMAP@Variant @\LIMAP@ClassType\endcsname}%
58 }
```

Thus the class specific options are completed.

59  $\langle$ /class $\rangle$

Now we can process all options.

```
60 \ProcessOptions
```

61  $\langle$ \*class $\rangle$

The requested class is loaded and the options remaining are processed.

```
62 \LoadClass{\csname LIMAP@Class@\LIMAP@Variant @\LIMAP@ClassType\endcsname}
63  $\langle$ /class $\rangle$ 
```

#### 4.1.4 Loading Required Packages

The package `longtable` is used internally to implement a part of the required functionality. Thus we need to ensure that it is loaded.

```
64 \RequirePackage{longtable}
```

The package `booktabs` is used internally to implement a part of the required functionality. Thus we need to ensure that it is loaded.

```
65 \RequirePackage{booktabs}
```

```
66 \class
```

```
67 \RequirePackage{fancyhdr}
```

```
68 \addtolength{\headheight}{2ex}%
```

```
69 \pagestyle{fancy}%
```

```
70 \cfoot{}
```

```
71 \rhead{\small\thepage}
```

```
72 \lhead{\textit{\footnotesize\@title}}
```

```
73 \def\@title{}
```

```
74 \endclass
```

Since the blocks are not supposed to line up at the end of the page we declare `\raggedbottom`.

```
75 \raggedbottom
```

## 4.2 Layout Parameters

The layout can be influenced by a large number of parameters. Thus the design decisions have been made transparent (to a certain degree at least). These options are not meant to be changed except when a new layout is being designed and implemented.

`\MapRuleWidth` The macro `\MapRuleWidth` determines the width of the rules drawn between blocks.

```
76 \newcommand\MapRuleWidth{.25pt}
```

`\MapContinued` This macro determines the text to be used in the title of continued maps. This macro is reset when the language specific initializations are performed.

```
77 \newcommand\MapContinued{}
```

`\MapContinuing` The macro `\MapContinuing` determines the text to be used at the bottom of the map which is continued. This macro is reset when the language specific initializations are performed.

```
78 \newcommand\MapContinuing{}
```

`\MapContinuingFormat` This macro determines the format of the bottom line on continued maps. I.e. it includes the text as well as font changing commands. The text is passed to this command as argument 1.

```
79 \newcommand\MapContinuingFormat[1]{\textit{\footnotesize #1}}
```

<code>\MapContinuedFormat</code>	This macro determines the format of the bottom line on continued maps. I.e. it includes the text passed to it as argument 1 as well as font changing commands. 80 <code>\newcommand\MapContinuedFormat[1]{, {\MapTitleContinuedSize #1}}</code>
<code>\MapFont</code>	The macro <code>\MapFont</code> determines the font changing command to be used when starting a new map. 81 <code>\let\MapFont\textsf</code>
<code>\MapTitleSize</code>	The macro <code>\MapTitleSize</code> determines the size changing command to be used when typesetting the title of a map. 82 <code>\let\MapTitleSize\Large</code>
<code>\MapTitleContinuedSize</code>	This macro determines the font changing command to be used for typesetting the additional text after titles on followup pages of multipage maps. 83 <code>\let\MapTitleContinuedSize\small</code>
<code>\MapParskip</code>	The macro <code>\MapParskip</code> determines the distance of the text from the separating rules. 84 <code>\newcommand\MapParskip{2ex}</code>
<code>\MapTitlefraction</code>	The macro <code>\MapTitlefraction</code> determines the part of the page width devoted to the title area. It is a fraction in the range from 0 to 1. 85 <code>\newcommand\MapTitlefraction{.2}</code>
<code>\MapTextfraction</code>	This macro determines the part of the page width devoted to the text area. It is a fraction in the range from 0 to 1. <code>\MapTitlefraction</code> and <code>\MapTextfraction</code> should add up to something less or equal to 1. Otherwise you will get some “overfull hbox” messages. 86 <code>\newcommand\MapTextfraction{.75}</code>

### 4.3 Adaptable Macros

<code>\MapNewpage</code>	The macro <code>\MapNewpage</code> is expanded whenever a new page is required between maps. Thus it can be used to suppress the newpages by <code>\leting</code> it to <code>\relax</code> . 87 <code>\let\MapNewpage\newpage</code>
<code>\MapTOC</code>	The macro <code>\MapTOC</code> is expanded to generate the entry in the table of contents. It can be redefined to allow another behaviour. 88 <code>\newcommand\MapTOC[1]{%</code> 89 <code>  \refstepcounter{\@nameuse{Map@TOC@name}\the\Map@level}}%</code> 90 <code>  \addcontentsline{toc}{\@nameuse{Map@TOC@name}\the\Map@level}}{#1}%</code> 91 <code>}</code>
<code>\MapTOCname</code>	The macro <code>\MapTOCname</code> contains the heading for the section title in contents blocks. This macro is reset when the language specific initializations are performed. 92 <code>\newcommand\MapTOCname{}</code>

`\MapTOCpage` The macro `\MapTOCpage` contains the heading for the page number in contents blocks. This macro is reset when the language specific initializations are performed.

```
93 \newcommand\MapTOCpage{}
```

`\MapTOCemph` The macro `\MapTOCemph`

```
94 \let\MapTOCemph=\emph
```

## 4.4 Language Specific Macros

This section contains interanl macros used to implement the functionality. New languages can be easily be added. For this puropose only a new macro has to be defined and a package/class option for the convenience of the user.

Consider you want to add a new language “latin” then you have to provide the command `\LIMAP@SelectLanguage@latin`. This macro should simply redefine the macros containing strings of the language specific texts. Examples for other languages are provided in this section.

To enable the language settings for “latin” the macro `\LIMAP@Language` has to be defined to contain the value “latin”. Usually this is accomplished by providing a convenient option to the package or class.

```
\LIMAP@SelectLanguage@austrian Provide the definition for the langauge “austrian”.
95 \providecommand\LIMAP@SelectLanguage@austrian{%
96 \renewcommand\MapContinued{ Fortsetzung}%
97 \renewcommand\MapContinuing{Fortsetzung\dots}
98 \renewcommand\MapTOCname{Titel}
99 \renewcommand\MapTOCpage{Seite}
100 }

\LIMAP@SelectLanguage@german Provide the definitions for the laguage “german”.
101 \providecommand\LIMAP@SelectLanguage@german{%
102 \renewcommand\MapContinued{ Fortsetzung}%
103 \renewcommand\MapContinuing{Fortsetzung\dots}
104 \renewcommand\MapTOCname{Titel}
105 \renewcommand\MapTOCpage{Seite}
106 }

\LIMAP@SelectLanguage@english Provide the definitions for the language “english”.
107 \providecommand\LIMAP@SelectLanguage@english{%
108 \renewcommand\MapContinued{ Continued}%
109 \renewcommand\MapContinuing{Continuing\dots}
110 \renewcommand\MapTOCname{Title}
111 \renewcommand\MapTOCpage{Page}
112 }

\LIMAP@SelectLanguage@USenglish Provide the definitions for the language “USenglish”.
113 \providecommand\LIMAP@SelectLanguage@USenglish{%
114 \renewcommand\MapContinued{ Continued}%
```

```

115 \renewcommand\MapContinuing{Continuing\dots}
116 \renewcommand\MapTOCname{Title}
117 \renewcommand\MapTOCpage{Page}
118 }

```

## 4.5 Internal Macros, Lengths, and Counters

This section contains internal macros used to implement the functionality.

- \Map@length** The length register `\Map@length` is allocated to store the width of the space between the columns of a block.
- ```

119 \newlength{\Map@length}

```
- \Map@level** The macro `\Map@level` determines the level of inclusion of maps. It is used to determine the appearance in the table of contents.
- ```

120 \newcount\Map@level
121 \Map@level=0

```
- \Map@blockcount** The macro `\Map@blockcount` is used to count the blocks per map to issue a style warning if required.
- ```

122 \newcount\Map@blockcount

```
- \ifMap@open@** The conditional `\ifMap@open@` is used to record the opening and closing of the `longtable` environment, since can not be used inside itself. Thus it can be closed before a new instance is opened.
- ```

123 \newif\ifMap@open@
124 \Map@open@false

```
- \Map@TOC@name** The macros `\Map@TOC@name...` provide a mapping between a number and a sectioning unit. This mapping is used when the entry in the table of contents is generated.
- ```

125 \@namedef{Map@TOC@name0}{chapter}
126 \@namedef{Map@TOC@name1}{section}
127 \@namedef{Map@TOC@name2}{subsection}
128 \@namedef{Map@TOC@name3}{subsubsection}
129 \@namedef{Map@TOC@name4}{paragraph}
130 \@namedef{Map@TOC@name5}{subparagraph}
131 \@namedef{Map@TOC@name6}{subsubparagraph}
132 \@namedef{Map@TOC@name7}{subsubparagraph}
133 \@namedef{Map@TOC@name8}{subsubparagraph}
134 \@namedef{Map@TOC@name9}{subsubparagraph}
135 \@namedef{Map@TOC@name10}{subsubparagraph}
136 \@namedef{Map@TOC@name11}{subsubparagraph}
137 \@namedef{Map@TOC@name12}{subsubparagraph}

```
- \Map@start** The macro `\Map@start` is used to initiate the use of a map. It uses the `longtable` environment to perform the page breaking and marking of continued pages.
- ```

138 \newcommand\Map@start{%

```

```

139 % \typeout{--- MAP START}%
140 \setlength{\Map@length}{\textwidth}%
141 \addtolength{\Map@length}{-\MapTitlefraction\textwidth}%
142 \addtolength{\Map@length}{-\MapTextfraction\textwidth}%
143 \MapTOC{\Map@TITLE}%
144 \longtable
145   {@}p{\MapTitlefraction\textwidth}@{\hspace{\Map@length}}
146   p{\MapTextfraction\textwidth}@{}%
147   \multicolumn{2}{@}p{\textwidth}@{}-%
148   \MapFont{\MapTitleSize\rule{0pt}{3ex}}%
149   \Map@TITLE}}
150 \endfirsthead
151 \multicolumn{2}{@}p{\textwidth}@{}-%
152 \MapFont{\MapTitleSize\rule{0pt}{3ex}}%
153 \Map@TITLE\MapContinuedFormat{\MapContinued}}}%
154 \endhead
155 \par\vspace*{-\parskip}\vspace*{-2ex}\\
156 &\rule{\MapTextfraction\textwidth}{\MapRuleWidth}\newline
157 \mbox{} \hfill\raisebox{3pt}{\MapContinuingFormat{\MapContinuing}}
158 \endfoot
159 &\rule{\MapTextfraction\textwidth}{\MapRuleWidth}%
160 \vspace{\MapParskip}
161 \endlastfoot
162 \xdef\@currentlabel{\Map@TITLE}%
163 \global\Map@open@true
164 }

```

`\Map@end` The macro `\Map@end` is expanded when the end of the end of the `longtable` environment might be needed. The boolean `\ifMap@open@` determines whether such an environment is really open.

```

165 \newcommand\Map@end{%
166 % \typeout{--- MAP END}%
167 \ifMap@open@
168   \global\Map@open@false
169   \endlongtable
170   \MapNewpage
171 \fi
172 \iftrue
173 \ifnum\Map@blockcount>9
174 \PackageWarning{limap}%
175 {*** The current map contains too much blocks: \the\Map@blockcount}%
176 \else\ifnum\Map@blockcount>7
177 \PackageWarning{limap}%
178 {--- The current map contains \the\Map@blockcount blocks.}%
179 \fi\fi
180 \fi
181 }

```

`\Map@UP` The macro `\Map@UP` contains the number of the parent map or the empty string.

```

182 \newcommand\Map@UP{}

```

`\Map@no` The counter `\Map@no` contains the sequence number for all maps. This value is used internally to reference single maps.

```
183 \newcount\Map@no
```

`\Map@parts@` The macro `\Map@parts@` is used to store the parts of the toplevel maps. This is the initialization of a feature otherwise used in the aux file.

```
184 \@namedef{Map@parts@}{}
```

## 4.6 Typesetting a Map

`Map` This environment determines the appearance of a Map. It is implemented as a longtable environment which takes care for the page breaks and inserts material at the end of the page and the beginning of the new page upon page break.

```
185 \newenvironment{Map}[1]{%
```

First the messages of `longtable` are modified to show this package name instead.

```
186 \def\LT@err{\PackageError{limap}}%
```

```
187 \def\LT@warn{\PackageWarning{limap}}%
```

The map local macro `\Block` and the environment `Block` is activated. The counter for blocks is reset.

```
188 \let\Block\Map@Block
```

```
189 \let\endBlock\Map@endBlock
```

```
190 \Map@blockcount=0
```

The number of the map in the internal counting is set by incrementing the old value.

```
191 \global\advance\Map@no1
```

```
192 % \typeout{--- \the\Map@no: \Map@UP}%
```

```
193 \ifx\Map@UP\empty\else
```

```
194 \immediate\write\@auxout
```

```
195 {\string\expandafter\string\xdef\string\csname\space
```

```
196 Map@parts@\Map@UP\string\endcsname{\string\csname\space
```

```
197 Map@parts@\Map@UP\string\endcsname\the\Map@no:}}%
```

```
198 \fi
```

```
199 \edef\Map@UP{\the\Map@no}%
```

```
200 \ifnum\Map@level>0
```

```
201 \xdef\Map@@up{\Map@UP}% Just to save the value across blocks.
```

```
202 \endgroup
```

```
203 % \typeout{--- Closing Map \the\Map@level}%
```

```
204 \Map@end
```

```
205 \begingroup
```

```
206 \edef\Map@UP{\Map@@up}%
```

```
207 \def\@currenvir{Map}%
```

```
208 \fi
```

```
209 \edef\Map@this{\the\Map@no}%
```

The entries for future use of submaps are written to the aux file.

```
210 \immediate\write\@auxout
```

```

211   {\string\global\string\@namedef{Map@title@\the\Map@no}{#1}}%
212 \immediate\write\@auxout
213   {\string\global\string\@namedef{Map@page@\the\Map@no}{\the\c@page}}%
214 \immediate\write\@auxout
215   {\string\global\string\@namedef{Map@parts@\the\Map@no}{}}%
216 % \typeout{--- Opening Map \the\Map@level}%
217 \global\advance\Map@level1
218 \def\Map@TITLE{#1}%
219 \Map@start
220 }{%
221 \Map@end
222 \global\advance\Map@level-1
223 % \typeout{--- At end of Map \Map@this level \the\Map@level}%
224 }

```

## 4.7 Typesetting a Block

**Map@Block** This macro is used to typeset a block inside a Map. To avoid abuse outside of a map it is activated within a Map only.

```

225 \newenvironment{Map@Block}[1]{\par\vspace*{-\parskip}\vspace*{-2ex}%
226   \\\null\par
227   \vspace*{\MapParskip}}%
228 \raggedright\hspace{Opt}\MapFont{#1}%
229 \gdef\@currentlabel{#1}%
230 % \global\advance\Map@blockcount1
231 &\parskip=\MapParskip
232 \rule{\MapTextfraction\textwidth}{\MapRuleWidth}\par

```

The final action is empty. Thus the block can be used as a simple macro as well.

```

233 }{%
234 }

```

**\WideBlock** The macro `\WideBlock` takes one argument which is added to the current block where the whole width of the table is used.

```

235 \newcommand\Wide@Block{\multicolumn2{@{1@{}}}{}}

```

## 4.8 Typesetting a Table of Contents

**\MapTableOfContents** The macro `\MapTableOfContents` produces the table of contents for the current map. It produces a tabular containing the titles and pages of all maps directly contained in the current map.

```

236 \newcommand\MapTableOfContents{%
237   \medskip\par
238   \xdef\Map@@{\csname Map@parts@\the\Map@no\endcsname}%
239   \gdef\Map@@@{}%
240   \centering

```

```

241 \begin{tabular}{p{.6\textwidth}r}\toprule
242 \MapTOCemph{\MapTOCname}&\MapTOCemph{\MapTOCpage}\\
243 \midrule
244 \expandafter\Map@toc@loop \Map@:@%
245 \\ \bottomrule
246 \end{tabular}
247 }

```

`\Map@toc@loop` The macro `\Map@toc@loop` is a recursive solution to loop through all elements of a list of children.

```

248 \def\Map@toc@loop#1:{%
249 \def\Map@@{#1}%
250 \ifx\Map@@\empty
251 \global\let\Map@@=\relax
252 \else
253 \gdef\Map@@{\Map@@@\nameuse{Map@title@#1}&\nameuse{Map@page@#1}%
254 \global\let\Map@@@=\%
255 \Map@toc@loop}%
256 \fi
257 \Map@@
258 }

```

## 4.9 Typesetting a Title Page

`\MakeTitle` The macro `\MakeTitle` can be used as a replacement for the `\maketitle` macro.

```

259 \newcommand\MakeTitle{\thispagestyle{empty}
260 \rule{0pt}{.25\textheight}\par
261 \mbox{}\hfill
262 \begin{minipage}{\MapTextfraction\textwidth}
263 \raggedright
264 \rule{\textwidth}{1pt}\par
265 \vspace*{5ex}%
266 \sf{\huge \@title}\par%
267 \vspace*{5ex}%
268 \rule{\textwidth}{1pt}\par
269 \vspace*{5ex}%
270 \MapFont{\large \@author} \par
271 \vspace*{10ex}%
272 \MapFont{\footnotesize \@date}
273 \vspace*{10ex}%
274 \end{minipage}%
275 \par
276 }

```

The new `\maketitle` macro is activated for the class.

```

277 <*class>
278 \let\maketitle\MakeTitle
279 </class>

```

## 4.10 Final Actions

Load the configuration file at the end if it can be found.

```
280 \InputIfFileExists{limap.cfg}{-}{-}
```

Finally we have to activate the proper settings for the choosen language.

```
281 \csname LIMAP@SelectLanguage@LIMAP@Language\endcsname
```

```
282 \ifLIMAP@strict
```

```
283 % \def\section{\PackageWarning{limap}{The sectioning command
```

```
284 %     'section' is not available.}}
```

```
285 \def\subsection{\PackageWarning{limap}{The sectioning command
```

```
286     'subsection' is not available.}}
```

```
287 \def\subsubsection{\PackageWarning{limap}{The sectioning command
```

```
288     'subsubsection' is not available.}}
```

```
289 \def\paragraph{\PackageWarning{limap}{The sectioning command
```

```
290     'paragraph' is not available.}}
```

```
291 \def\subparagraph{\PackageWarning{limap}{The sectioning command
```

```
292     'subparagraph' is not available.}}
```

```
293 \def\subsubparagraph{\PackageWarning{limap}{The sectioning command
```

```
294     'subsubparagraph' is not available.}}
```

```
295 \fi
```

That's all.

## Index

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

<b>B</b>		<code>\LIMAP@ClassType</code> . . . <u>40</u> , 41–44, 57, 62	
<code>\Block</code> . . . . .	4, 188	<code>\LIMAP@Language</code> . . . . . <u>31</u> , 32–36, 281	
<code>Block</code> (environment) . . . . .	4	<code>\LIMAP@RCS</code> . . . . . <u>1</u> , 3, 4	
<code>\bottomrule</code> . . . . .	245	<code>\LIMAP@SelectLanguage@austrian</code> . . . . . <u>95</u>	
<b>C</b>		<code>\LIMAP@SelectLanguage@english</code> . . . . . <u>107</u>	
<code>\c@page</code> . . . . .	213	<code>\LIMAP@SelectLanguage@german</code> . . . . . <u>101</u>	
<code>\cfoot</code> . . . . .	70	<code>\LIMAP@SelectLanguage@USenglish</code> . . . . . <u>113</u>	
<b>D</b>		<code>\LIMAP@strictfalse</code> . . . . . 38	
<code>\docdate</code> . . . . .	<u>6</u>	<code>\LIMAP@stricttrue</code> . . . . . 37	
<code>\docversion</code> . . . . .	<u>5</u>	<code>\LIMAP@Variant</code> . . . . . <u>45</u> , 46, 47, 57, 62	
<b>E</b>		<code>\longtable</code> . . . . . 144	
<code>\endBlock</code> . . . . .	189	<code>\LT@err</code> . . . . . 186	
<code>\endfirsthead</code> . . . . .	150	<code>\LT@warn</code> . . . . . 187	
<code>\endfoot</code> . . . . .	158	<b>M</b>	
<code>\endhead</code> . . . . .	154	<code>\MakeTitle</code> . . . . . <u>259</u> , 278	
<code>\endlastfoot</code> . . . . .	161	<code>\maketitle</code> . . . . . 4, 278	
<code>\endlongtable</code> . . . . .	169	<code>Map</code> (environment) . . . . . 4, <u>185</u>	
environments:		<code>\Map@@</code> . . . . . 238, 244, 249–251, 253, 257	
<code>Block</code> . . . . .	4	<code>\Map@@@</code> . . . . . 239, 253, 254	
<code>Map</code> . . . . .	4, <u>185</u>	<code>\Map@@@up</code> . . . . . 201, 206	
<code>Map@Block</code> . . . . .	<u>225</u>	<code>\Map@Block</code> . . . . . 188	
<b>F</b>		<code>Map@Block</code> (environment) . . . . . <u>225</u>	
<code>\filedate</code> . . . . .	<u>4</u> , 6, 26, 29	<code>\Map@blockcount</code> . . . . .	
<code>\filename</code> . . . . .	<u>2</u> , 16	. . . . . <u>122</u> , 173, 175, 176, 178, 190, 230	
<code>\fileversion</code> . . . . .	<u>3</u> , 5	<code>\Map@end</code> . . . . . <u>165</u> , 204, 221	
<b>I</b>		<code>\Map@endBlock</code> . . . . . 189	
<code>\ifLIMAP@strict</code> . . . . .	<u>37</u> , 282	<code>\Map@length</code> . . . . . <u>119</u> , 140–142, 145	
<code>\ifMap@open@</code> . . . . .	<u>123</u> , 167	<code>\Map@level</code> . . . . . 89,	
<b>L</b>		90, <u>120</u> , 200, 203, 216, 217, 222, 223	
<code>\lhead</code> . . . . .	72	<code>\Map@no</code> . . . . . <u>183</u> , 191, 192,	
<code>\LIMAP@Class@base@article</code> . . . . .	48	197, 199, 209, 211, 213, 215, 238	
<code>\LIMAP@Class@base@book</code> . . . . .	50	<code>\Map@open@false</code> . . . . . 124, 168	
<code>\LIMAP@Class@base@letter</code> . . . . .	51	<code>\Map@open@true</code> . . . . . 163	
<code>\LIMAP@Class@base@report</code> . . . . .	49	<code>\Map@parts@</code> . . . . . <u>184</u>	
<code>\LIMAP@Class@koma@article</code> . . . . .	52	<code>\Map@start</code> . . . . . <u>138</u> , 219	
<code>\LIMAP@Class@koma@book</code> . . . . .	54	<code>\Map@this</code> . . . . . 209, 223	
<code>\LIMAP@Class@koma@letter</code> . . . . .	55	<code>\Map@TITLE</code> . . . . . 143, 149, 153, 162, 218	
<code>\LIMAP@Class@koma@report</code> . . . . .	53	<code>\Map@toc@loop</code> . . . . . 244, <u>248</u>	
		<code>\Map@TOC@name</code> . . . . . <u>125</u>	
		<code>\Map@UP</code> . . . . . <u>182</u> ,	
		192, 193, 196, 197, 199, 201, 206	

