

# `omdoc.sty/cls`: Semantic Markup for Open Mathematical Documents in $\text{\LaTeX}$

Michael Kohlhase  
Jacobs University, Bremen  
<http://kwarc.info/kohlhase>

April 2, 2015

## **Abstract**

The `omdoc` package is part of the  $\text{\LaTeX}$  collection, a version of  $\text{\TeX}/\text{\LaTeX}$  that allows to markup  $\text{\TeX}/\text{\LaTeX}$  documents semantically without leaving the document format, essentially turning  $\text{\TeX}/\text{\LaTeX}$  into a document format for mathematical knowledge management (MKM).

This package supplies an infrastructure for writing OMDoc documents in  $\text{\LaTeX}$ . This includes a simple structure sharing mechanism for  $\text{\LaTeX}$  that allows to move from a copy-and-paste document development model to a copy-and-reference model, which conserves space and simplifies document management. The augmented structure can be used by MKM systems for added-value services, either directly from the  $\text{\LaTeX}$  sources, or after translation.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>The User Interface</b>	<b>3</b>
2.1	Package and Class Options . . . . .	3
2.2	Document Structure . . . . .	3
2.3	Ignoring Inputs . . . . .	5
2.4	Structure Sharing . . . . .	5
2.5	Colors . . . . .	6
<b>3</b>	<b>Miscellaneous</b>	<b>6</b>
<b>4</b>	<b>Limitations</b>	<b>6</b>
<b>5</b>	<b>Implementation: The OMDoc Class</b>	<b>7</b>
5.1	Class Options . . . . .	7
5.2	Setting up Namespaces and Schemata for LaTeXXML . . . . .	8
5.3	Beefing up the <code>document</code> environment . . . . .	8
<b>6</b>	<b>Implementation: OMDoc Package</b>	<b>9</b>
6.1	Package Options . . . . .	9
6.2	Document Structure . . . . .	10
6.3	Front and Backmatter . . . . .	13
6.4	Ignoring Inputs . . . . .	15
6.5	Structure Sharing . . . . .	15
6.6	Colors . . . . .	16
6.7	L <sup>A</sup> T <sub>E</sub> X Commands we interpret differently . . . . .	17
6.8	Miscellaneous . . . . .	17
6.9	Leftovers . . . . .	17

EdN:1  
EdN:2  
EdN:3

EdN:4

# 1 Introduction

The `omdoc` package supplies macros and environment that allow to label document fragments and to reference them later in the same document or in other documents. In essence, this enhances the document-as-trees model to documents-as-directed-acyclic-graphs (DAG) model. This structure can be used by MKM systems for added-value services, either directly from the  $\text{\LaTeX}$  sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the  $\text{\LaTeX}$  collection.

$\text{\LaTeX}$  is a version of  $\text{\TeX}$ / $\text{\LaTeX}$  that allows to markup  $\text{\TeX}$ / $\text{\LaTeX}$  documents semantically without leaving the document format, essentially turning  $\text{\TeX}$ / $\text{\LaTeX}$  into a document format for mathematical knowledge management (MKM). The package supports direct translation to the OMDoc format [Koh06]

DAG models of documents allow to replace the “Copy and Paste” in the source document with a label-and-reference model where document are shared in the document source and the formatter does the copying during document formatting/presentation.<sup>123</sup>

# 2 The User Interface

The `omdoc` package generates four files: `omdoc.cls`, `omdoc.sty` and their  $\text{\LaTeX}$ ML bindings `omdoc.cls.ltxml` and `omdoc.sty.ltxml`. We keep the corresponding code fragments together, since the documentation applies to both of them and to prevent them from getting out of sync. The OMDoc class is a minimally changed variant of the standard `article` class that includes the functionality provided by `omdoc.sty`. Most importantly, `omdoc.cls` sets up the  $\text{\LaTeX}$ ML infrastructure and thus should be used if OMDoc is to be generated from the  $\text{\LaTeX}$  sources. The rest of the documentation pertains to the functionality introduced by `omdoc.sty`.

## 2.1 Package and Class Options

`noindex` `omdoc.sty` has the `noindex` package option, which allows to suppress the creation of index entries. The option can be set to activate multi-file support, see [Koh15c] for details.

`extrefs` `omdoc.cls` accepts all options of the `omdoc.sty` (see Subsection 2.0) and `article.cls` and just passes them on to these.<sup>4</sup>

## 2.2 Document Structure

`document` The top-level `document` environment is augmented with an optional key/value

<sup>1</sup>EDNOTE: talk about the advantages and give an example.

<sup>2</sup>EDNOTE: is there a way to load documents at URIs in  $\text{\LaTeX}$ ?

<sup>3</sup>EDNOTE: integrate with  $\text{\LaTeX}$ ML’s `XMRef` in the Math mode.

<sup>4</sup>EDNOTE: describe them

argument that can be used to give metadata about the document. For the moment only the `id` key is used to give an identifier to the `omdoc` element resulting from the  $\text{\LaTeX}$  transformation.

`omgroup` The structure of the document is given by the `omgroup` environment just like in OMDoc. In the  $\text{\LaTeX}$  route, the `omgroup` environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of `omgroup` environments. Correspondingly, the `omgroup` environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the `omgroup`. The optional metadata argument has the keys `id` for an identifier, `creators` and `contributors` for the Dublin Core metadata [DCM03]; see [Koh15a] for details of the format. The `short` allows to give a short title for the generated section. If the title contains semantic macros, they need to be protected by `\protect`, and we need to give the `loadmodules` key it needs no value. For instance we would have

```
\begin{module}{foo}
\symdef{bar}{B^a_r}
...
\begin{omgroup}{Introducing $\protect\bar$ Derivations}
```

$\text{\LaTeX}$  automatically computes the sectioning level, from the nesting of `omgroup` environments. But sometimes, we want to skip levels (e.g. to use a subsection\* as an introduction for a chapter). Therefore the `omdoc` package provides a variant `blindomgroup` that does not produce markup, but increments the sectioning level and logically groups document parts that belong together, but where traditional document markup relies on convention rather than explicit markup. The `blindomgroup` environment is useful e.g. for creating frontmatter at the correct level. Example 1 shows a typical setup for the outer document structure of a book with parts and chapters. We use two levels of `blindomgroup`:

- The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This `blindomgroup` makes sure that the introductory remarks become a “chapter” instead of a “part”.
- The inner one groups the frontmatter<sup>1</sup> and makes the preface of the book a section-level construct. Note that here the `display=flow` on the `omgroup` environment prevents numbering as is traditional for prefaces.

`\currentsectionlevel` The `\currentsectionlevel` macro supplies the name of the current sectioning level, e.g. “chapter”, or “subsection”. `\CurrentSectionLevel` is the capitalized variant. They are useful to write something like “In this `\currentsectionlevel`, we will...” in an `omgroup` environment, where we do not know which sectioning level we will end up.

---

<sup>1</sup>We shied away from redefining the `frontmatter` to induce a `blindomgroup`, but this may be the “right” way to go in the future.

```

\begin{document}
\begin{blindomgroup}
\begin{blindomgroup}
\begin{frontmatter}
\maketitle\newpage
\begin{omgroup}[display=flow]{Preface}
... <<preface>> ...
\end{omgroup}
\clearpage\setcounter{tocdepth}{4}\tableofcontents\clearpage
\end{frontmatter}
\end{blindomgroup}
... <<introductory remarks>> ...
\end{blindomgroup}
\begin{omgroup}{Introduction}
... <<intro>> ...
\end{omgroup}
... <<more chapters>> ...
\bibliographystyle{alpha}\bibliography{kwarc}
\end{document}

```

**Example 1:** A typical Document Structure of a Book

## 2.3 Ignoring Inputs

**ignore** The `ignore` environment can be used for hiding text parts from the document structure. The body of the environment is not PDF or DVI output unless the `showignores` option is given to the `omdoc` class or `package`. But in the generated OMDoc result, the body is marked up with a `ignore` element. This is useful in two situations. For

**editing** One may want to hide unfinished or obsolete parts of a document

**narrative/content markup** In  $\text{\LaTeX}$  we mark up narrative-structured documents. In the generated OMDoc documents we want to be able to cache content objects that are not directly visible. For instance in the `statements` package [Koh15d] we use the `\inlinedef` macro to mark up phrase-level definitions, which verbalize more formal definitions. The latter can be hidden by an `ignore` and referenced by the `verbalizes` key in `\inlinedef`.

## 2.4 Structure Sharing

`\STRlabel` The `\STRlabel` macro takes two arguments: a label and the content and stores the content for later use by `\STRcopy[⟨URL⟩]{⟨label⟩}`, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL `⟨URL⟩` that lets  $\text{\LaTeX}$ ML generate the correct reference.

`\STRcopy`

`\STRsemantics` The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in  $\text{\LaTeX}$ . This allows to specify the meaning of the content (whatever that may mean) in cases,

where the source document is not formatted for presentation, but is transformed into some content markup format.<sup>5</sup>

## 2.5 Colors

For convenience, the `omdoc` package defines a couple of color macros for the `color` package: For instance `\blue` abbreviates `\textcolor{blue}`, so that `\blue{\textit{something}}` writes *something* in blue. The macros `\red` `\green`, `\cyan`, `\magenta`, `\brown`, `\yellow`, `\orange`, `\gray`, and finally `\black` are analogous.

## 3 Miscellaneous

## 4 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the `sTeX` TRAC [sTeX].

1. none reported yet

---

<sup>5</sup>EdNOTE: make an example

## 5 Implementation: The OMDoc Class

The functionality is spread over the `omdoc` class and package. The class provides the `document` environment and the `omdoc` element corresponds to it, whereas the package provides the concrete functionality.

`omdoc.dtx` generates four files: `omdoc.cls` (all the code between `<*cls>` and `</cls>`), `omdoc.sty` (between `<*package>` and `</package>`) and their L<sup>A</sup>T<sub>E</sub>XML bindings (between `<*ltxml.cls>` and `</ltxml.cls>` and `<*ltxml.sty>` and `</ltxml.sty>` respectively). We keep the corresponding code fragments together, since the documentation applies to both of them and to prevent them from getting out of sync.

### 5.1 Class Options

To initialize the `omdoc` class, we declare and process the necessary options.

```
1 <*cls>
2 \DeclareOption{showmeta}{\PassOptionsToPackage{\CurrentOption}{metakeys}}
3 \def\omdoc@class{article}
4 \DeclareOption{report}{\def\omdoc@class{report}\PassOptionsToPackage{\CurrentOption}{omdoc}}
5 \DeclareOption{book}{\def\omdoc@class{book}\PassOptionsToPackage{\CurrentOption}{omdoc}}
6 \DeclareOption{showignores}{\PassOptionsToPackage{\CurrentOption}{omdoc}}
7 \DeclareOption{showmods}{\PassOptionsToPackage{\CurrentOption}{modules}}
8 \DeclareOption{extrefs}{\PassOptionsToPackage{\CurrentOption}{sref}}
9 \DeclareOption{noauxreq}{\PassOptionsToPackage{\CurrentOption}{modules}}
10 \DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
11 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{omdoc}}
12 \ProcessOptions
13 </cls>
14 <*ltxml.cls>
15 # -*- CPERL -*-
16 package LaTeXML::Package::Pool;
17 use strict;
18 use LaTeXML::Package;
19 use LaTeXML::Util::Pathname;
20 use Cwd qw(abs_path);
21 DeclareOption('report',sub {PassOptions('omdoc','sty',ToString(Digest(T_CS('\CurrentOption'))))})
22 DeclareOption('book',sub {PassOptions('omdoc','sty',ToString(Digest(T_CS('\CurrentOption'))))})
23 DeclareOption('showignores',sub {PassOptions('omdoc','sty',ToString(Digest(T_CS('\CurrentOption'))))})
24 DeclareOption('extrefs',sub {PassOptions('sref','sty',ToString(Digest(T_CS('\CurrentOption'))))})
25 DeclareOption(undef,sub {PassOptions('article','cls',ToString(Digest(T_CS('\CurrentOption'))))})
26 ProcessOptions();
27 </ltxml.cls>
```

We load `article.cls`, and the desired packages. For the L<sup>A</sup>T<sub>E</sub>XML bindings, we make sure the right packages are loaded.

```
28 <*cls>
29 \LoadClass{\omdoc@class}
30 \RequirePackage{etoolbox}
31 \RequirePackage{omdoc}
```

```

32 </cls>
33 <*ltxml.cls>
34 LoadClass('article');
35 RequirePackage('sref');
36 </ltxml.cls>

```

## 5.2 Setting up Namespaces and Schemata for LaTeXXML

Now, we also need to register the namespace prefixes for LaTeXXML to use.

```

37 <*ltxml.cls>
38 RegisterNamespace('omdoc'=>"http://omdoc.org/ns");
39 RegisterNamespace('om'=>"http://www.openmath.org/OpenMath");
40 RegisterNamespace('m'=>"http://www.w3.org/1998/Math/MathML");
41 RegisterNamespace('dc'=>"http://purl.org/dc/elements/1.1/");
42 RegisterNamespace('cc'=>"http://creativecommons.org/ns");
43 RegisterNamespace('stex'=>"http://kwarc.info/ns/sTeX");
44 RegisterNamespace('ltx'=>"http://dlmf.nist.gov/LaTeXML");
45 </ltxml.cls>

```

Since we are dealing with a class, we need to set up the document type in the LaTeXXML bindings.

```

46 <*ltxml.cls>
47 RelaxNGSchema('omdoc+ltxml',
48     '#default'=>"http://omdoc.org/ns",
49     'om'=>"http://www.openmath.org/OpenMath",
50     'm'=>"http://www.w3.org/1998/Math/MathML",
51     'dc'=>"http://purl.org/dc/elements/1.1/",
52     'cc'=>"http://creativecommons.org/ns",
53     'stex'=>"http://kwarc.info/ns/sTeX",
54     'ltx'=>"http://dlmf.nist.gov/LaTeXML");
55 </ltxml.cls>

```

Then we load the omdoc package, which we define separately in the next section so that it can be loaded separately<sup>6</sup>

```

56 <*ltxml.cls>
57 RequirePackage('omdoc');
58 </ltxml.cls>

```

## 5.3 Beefing up the document environment

Now, we will define the environments we need. The top-level one is the `document` environment, which we redefined so that we can provide keyval arguments.

`document` For the moment we do not use them on the LaTeX level, but the document identifier is picked up by LaTeXXML.

```

59 <*cls>
60 \let\orig@document=\document

```

---

<sup>6</sup>EdNOTE: reword



```

61 \srefaddidkey{document}
62 \renewcommand{\document}[1][\metasetkeys{document}{#1}\orig@document}
63 \</cls>
64 \<*!xml.cls>
65 sub xmlBase {
66   my $baseuri = LookupValue('URLBASE');
67   $baseuri =~ s/\$//g; # No trailing slashes
68   Tokenize($baseuri); }
69 DefEnvironment('{document} OptionalKeyVals:omdoc',
70   "<omdoc:omdoc "
71   .   "?&GetKeyVal(#1,'id')(xml:id='&GetKeyVal(#1,'id'))"
72   .   "(?&Tokenize(&LookupValue('SOURCEBASE'))"
73   .   "(xml:id='&Tokenize(&LookupValue('SOURCEBASE')).omdoc'))()"
74   .   "?&Tokenize(&LookupValue('URLBASE'))"
75   .   "(xml:base='&xmlBase()')()>"
76   .   "#body"
77   . "</omdoc:omdoc>",
78   beforeDigest=> sub { AssignValue(inPreamble=>0); },
79   afterDigest=> sub { $_[0]->getGullet->flush; return; },
80   afterDigestBegin => sub {
81     $_[1]->setProperty(id => Expand(T_CS('\thedocument@ID')));
82     if (my $ops = LookupValue('@at@begin@document')) {
83       Digest(Tokens(@$ops)); }
84     else {
85       return; } },
86   beforeDigestEnd => sub {
87     $_[0]->getGullet->flush;
88     if (my $ops = LookupValue('@at@end@document')) {
89       Digest(Tokens(@$ops)); }
90     else {
91       return; } },
92   mode => 'text');
93 Tag('omdoc:omdoc', 'afterOpen:late'=>\&insertFrontMatter);
94 \</!xml.cls>%$

```

## 6 Implementation: OMDoc Package

### 6.1 Package Options

The initial setup for L<sup>A</sup>T<sub>E</sub>XML:

```

95 \<*!xml.sty>
96 package LaTeXML::Package::Pool;
97 use strict;
98 use LaTeXML::Package;
99 use Cwd qw(abs_path);
100 \</!xml.sty>

```

We declare some switches which will modify the behavior according to the package options. Generally, an option xxx will just set the appropriate switches to true (otherwise they stay false).<sup>7</sup>

```

101 <*package>
102 \DeclareOption{showmeta}{\PassOptionsToPackage{\CurrentOption}{metakeys}}
103 \DeclareOption{showmods}{\PassOptionsToPackage{\CurrentOption}{modules}}
104 \newcount\section@level
105 \newif\ifshow@ignores\show@ignorefalse
106 \def\omdoc@class{article}\section@level=2
107 \DeclareOption{report}{\def\omdoc@class{report}\section@level=1}
108 \newif\ifclass@book\class@bookfalse
109 \DeclareOption{book}{\def\omdoc@class{book}\section@level=0\class@booktrue}
110 \DeclareOption{showignores}{\show@ignoretrue}
111 \DeclareOption{extrefs}{\PassOptionsToPackage{\CurrentOption}{sref}}
112 \DeclareOption*{}% accept all other options
113 \ProcessOptions
114 </package>
115 <*ltxml.sty>
116 DeclareOption('report','');
117 DeclareOption('book','');
118 DeclareOption('showignores','');
119 DeclareOption('extrefs','');
120 </ltxml.sty>

```

Then we need to set up the packages by requiring the `sref` package to be loaded.

```

121 <*package>
122 \RequirePackage{sref}
123 \RequirePackage{xspace}
124 \RequirePackage{comment}
125 </package>
126 <*ltxml.sty>
127 RequirePackage('sref');
128 RequirePackage('xspace');
129 RequirePackage('omtext');
130 </ltxml.sty>

```

## 6.2 Document Structure

The structure of the document is given by the `omgroup` environment just like in OMDoc. The hierarchy is adjusted automatically according to the  $\text{\LaTeX}$  class in effect.

`\currentsectionlevel`

```

131 <*package>
132 \def\currentsectionlevel{document\xspace}%
133 \def\Currentsectionlevel{Document\xspace}%

```

---

<sup>7</sup>EdNOTE: need an implementation for  $\text{\LaTeX}$ ML

```

134 </package>
135 <*ltxml.sty>
136 DefMacro('\'currentsectionlevel',\'\'@currentsectionlevel\xspace');
137 DefMacro('\'Currentsectionlevel',\'\'@Currentsectionlevel\xspace');
138 DefConstructor('\'@currentsectionlevel',
139               "<ltx:text class='omdoc-currentsectionlevel'>section</ltx:text>");
140 DefConstructor('\'@CurrentSectionLevel',
141               "<ltx:text class='omdoc-Currentsectionlevel'>Section</ltx:text>");
142 </ltxml.sty>

blindomgroup

143 <*package>
144 \newcommand\at@begin@blindomgroup[1]{%
145 \newenvironment{blindomgroup}
146 {\advance\section@level by 1\at@begin@blindomgroup\section@level}
147 {\advance\section@level by -1}
148 </package>
149 <*ltxml.sty>
150 DefEnvironment('\'{blindomgroup} OptionalKeyVals:omgroup',
151               "<omdoc:omgroup layout='invisible'"
152               . "  "?&GetKeyVal{#1,'id'}(xml:id='&GetKeyVal{#1,'id'}')()>"
153               . "  "?&GetKeyVal{#1,'type'}(type='&GetKeyVal{#1,'type'}')()>\n"
154               . "#body\n"
155               . "</omdoc:omgroup>");
156 </ltxml.sty>

\omgroup@c1 Convenience macro: defines the \currentsectionlevel macro from the keywords
in the arguments

157 <*package>
158 \newcommand\omgroup@c1[2]{%
159 \def\currentsectionlevel{#1\xspace}%
160 \def\Currentsectionlevel{#2\xspace}}

\omgroup@nonum convenience macro: \omgroup@nonum{<level>}{<title>} makes an unnumbered sectioning
with title <title> at level <level>.

161 \newcommand\omgroup@nonum[2]{%
162 \ifx\hyper@anchor\undefined\else\phantomsection\fi%
163 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}}

\omgroup@num convenience macro: \omgroup@num{<level>}{<title>} makes numbered sectioning
with title <title> at level <level>. We have to check the short key was given in the
omgroup environment and – if it is use it. But how to do that depends on whether
the rdfmeta package has been loaded.

164 \newcommand\omgroup@num[2]{\sref@label@id{\omdoc@sect@Name~\@nameuse{the#1}}}%
165 \ifx\omgroup@short\@empty\@nameuse{#1}{#2}%
166 \else\@ifundefined{rdfmeta@sectioning}{\@nameuse{#1}[\omgroup@short]{#2}}%
167 {\@nameuse{rdfmeta@#2@old}[\omgroup@short]{#2}}\fi
168 </package>

```

omgroup

```

169 <*package>
170 \def\@true{true}
171 \def\@false{false}
172 \srefaddidkey{omgroup}
173 \addmetakey{omgroup}{creators}
174 \addmetakey{omgroup}{date}
175 \addmetakey{omgroup}{contributors}
176 \addmetakey{omgroup}{type}
177 \addmetakey*{omgroup}{short}
178 \addmetakey*{omgroup}{display}
179 \addmetakey[false]{omgroup}{loadmodules}[true]

```

we define a switch for numbering lines and a hook for the beginning of groups:

`\at@begin@omgroup` The `\at@begin@omgroup` macro allows customization. It is run at the beginning of the `omgroup`, i.e. after the section heading.

```

180 \newif\if@num\@numtrue
181 \newif\if@frontmatter\@frontmatterfalse
182 \newif\if@backmatter\@backmatterfalse
183 \newcommand\at@begin@omgroup[3] [] {}

```

Then we define a helper macro that takes care of the sectioning magic. It comes with its own key/value interface for customization.

```

184 \addmetakey{omdoc@sect}{name}
185 \addmetakey{omdoc@sect}{Name}
186 \addmetakey[false]{omdoc@sect}{clear}[true]
187 \addmetakey{omdoc@sect}{ref}
188 \addmetakey[false]{omdoc@sect}{num}[true]
189 \newcommand\omdoc@sectioning[3] [] {\metasetkeys{omdoc@sect}{#1}%
190 \ifx\omdoc@sect@clear\@true\cleardoublepage\fi%
191 \if@num% numbering not overridden by frontmatter, etc.
192 \ifx\omdoc@sect@num\@true\omgroup@num{#2}{#3}\else\omgroup@nonum{#2}{#3}\fi
193 \omgroup@c1\omdoc@sect@name\omdoc@sect@Name
194 \else\omgroup@nonum{#2}{#3}\fi}

```

and another one, if redefines the `\addtocontentsline` macro of L<sup>A</sup>T<sub>E</sub>X to import the respective macros. It takes as an argument a list of module names.

```

195 \newcommand\omgroup@redefine@addtocontents[1]{\edef\@import{#1}%
196 \@for\@I:=\@import\do{\edef\@path{\csname\@I @cd@file@base\endcsname}%
197 \@ifundefined{tf@toc}\relax{\protected@write\tf@toc}{\string\@requiremodules{\@path}{tex}}}}
198 \ifx\hyper@anchor\@undefined% hyperref.sty loaded?
199 \def\addcontentsline##1##2##3{%
200 \addtocontents{##1}{\protect\contentsline{##2}{\string\importmodules{#1}##3}{\thepage}}}
201 \else\def\addcontentsline##1##2##3{%
202 \addtocontents{##1}{\protect\contentsline{##2}{\string\importmodules{#1}##3}{\thepage}{\@current}}
203 \fi}% hyperref.sty loaded?

```

now the `omgroup` environment itself. This takes care of the table of contents via the helper macro above and then selects the appropriate sectioning command from `article.cls`.

```

204 \newenvironment{omgroup}[2][]% keys, title
205 {\metasetkeys{omgroup}{#1}\sref{target}%
206 \ifx\omgroup@display\st@flow\@numfalse\fi
207 \if@frontmatter\@numfalse\fi

```

If the `loadmodules` key is set on `\begin{omgroup}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```

208 \ifx\omgroup@loadmodules\@true%
209 \omgroup@redefine@addtocontents{\@ifundefined{mod@id}\imported@modules%
210 {\@ifundefined{mod@id @cd@file@base}\imported@modules\mod@id}}\fi%
    now we only need to construct the right sectioning depending on the value of
    \section@level.
211 \advance\section@level by 1
212 \ifcase\section@level%
213 \or\omdoc@sectioning[name=part,Name=Part,clear,num]{part}{#2}%
214 \or\omdoc@sectioning[name=chapter,Name=Chapter,clear,num]{chapter}{#2}%
215 \or\omdoc@sectioning[name=section,Name=Section,num]{section}{#2}%
216 \or\omdoc@sectioning[name=subsection,Name=Subsection,num]{subsection}{#2}%
217 \or\omdoc@sectioning[name=subsubsection,Name=Subsubsection,num]{subsubsection}{#2}%
218 \or\omdoc@sectioning[name=paragraph,Name=Paragraph,ref=this paragraph]{paragraph}{#2}%
219 \or\omdoc@sectioning[name=subparagraph,Name=Subparagraph,ref=this subparagraph]{paragraph}{#2}%
220 \fi% \ifcase
221 \at@begin@omgroup[#1]\section@level{#2}}}% for customization
222 {\advance\section@level by -1}
223 \end{package}
224 \end{ltxml.sty}
225 DefEnvironment('omgroup' OptionalKeyVals:omgroup {}',
226               "<omdoc:omgroup layout='sectioning'"
227               . "&GetKeyVal{#1,'id'}(xml:id='&GetKeyVal{#1,'id'}')()"
228               . "&GetKeyVal{#1,'type'}(type='&GetKeyVal{#1,'type'}')()>\n"
229               . "<dc:title>#2</dc:title>\n"
230               . "#body\n"
231               . "</omdoc:omgroup>");
232 \end{ltxml.sty}

```

## 6.3 Front and Backmatter

Index markup is provided by the `omtext` package [Koh15b], so in the `omdoc` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

233 \providecommand\printindex{\IfFileExists{\jobname.ind}{\input{\jobname.ind}}{}}
234 \end{package}
235 \end{ltxml.sty}
236 DefConstructor('\printindex', '<omdoc:index/>');
237 \end{ltxml.sty}

```

`\tableofcontents` The table of contents already exists in L<sup>A</sup>T<sub>E</sub>X, so we only need to provide a L<sup>A</sup>T<sub>E</sub>X<sub>ML</sub> binding for it.

```
239 <*ltxml.sty>
240 DefConstructor(' \tableofcontents',
241               "<omdoc:tableofcontents level='&ToString(&CounterValue('tocdepth'))' />");
242 </ltxml.sty>
```

The case of the `\bibliography` command is similar

`\bibliography`

```
243 <*ltxml.sty>
244 DefConstructor(' \bibliography{ }', "<omdoc:bibliography files='#1' />");
245 </ltxml.sty>
```

`frontmatter` `book.cls` already has a `\frontmatter` macro, so we have to redefine the front matter environment in this case.

```
246 <*cls>
247 \ifclass@book
248 \renewenvironment{frontmatter}
249 {\@frontmattertrue\cleardoublepage\@mainmatterfalse\pagenumbering{roman}}
250 {\@frontmatterfalse\setcounter{page}{1}\pagenumbering{arabic}}
251 \else
252 \newenvironment{frontmatter}
253 {\@frontmattertrue\pagenumbering{roman}}
254 {\@frontmatterfalse\setcounter{page}{1}\pagenumbering{arabic}}
255 \fi
256 </cls>
257 <*ltxml.cls>
258 DefEnvironment('{frontmatter}', '#body');
259 </ltxml.cls>
260 % \End{macrocode}
261 % \end{environment}
262 %
263 % \begin{environment}{backmatter}
264 % |book.cls| already has a |\backmatter| macro, so we have to redefine the back
265 % matter environment in this case.
266 % \begin{macrocode}
267 <*cls>
268 \ifclass@book
269 \renewenvironment{backmatter}
270 {\cleardoublepage\@mainmatterfalse\@backmattertrue}
271 {\@backmatterfalse}
272 \else
273 \newenvironment{backmatter}{\@backmattertrue}{\@backmatterfalse}
274 \fi
275 </cls>
276 <*ltxml.cls>
277 DefEnvironment('{backmatter}', '#body');
278 </ltxml.cls>
```

## 6.4 Ignoring Inputs

```
ignore
279 <*package>
280 \ifshow@ignores
281 \addmetakey{ignore}{type}
282 \addmetakey{ignore}{comment}
283 \newenvironment{ignore}[1] []
284 {\metasetkeys{ignore}{#1}\textless\ignore@type\textgreater\bgroup\itshape}
285 {\egroup\textless/\ignore@type\textgreater}
286 \renewenvironment{ignore}{}{}\else\excludacomment{ignore}\fi
287 </package>
288 <*ltxml.sty>
289 DefKeyVal('ignore','type','Semiverbatim');
290 DefKeyVal('ignore','comment','Semiverbatim');
291 DefEnvironment('{ignore} OptionalKeyVals:ignore',
292               "<omdoc:ignore  %&GetKeyVals(#1)>#body</omdoc:ignore>");
293 </ltxml.sty>
```

## 6.5 Structure Sharing

**\STRlabel** The main macro, it is used to attach a label to some text expansion. Later on, using the **\STRcopy** macro, the author can use this label to get the expansion originally assigned.

```
294 <*package>
295 \long\def\STRlabel#1#2{\STRlabeldef{#1}{#2}{#2}}
296 </package>
297 <*ltxml.sty>
298 DefConstructor('\STRlabel{}{}', sub {
299   my($document,$label,$object)=@_;
300   $document->absorb($object);
301   $document->addAttribute('xml:id'=>ToString($label)) if $label; });
302 </ltxml.sty>
```

**\STRcopy** The **\STRcopy** macro is used to call the expansion of a given label. In case the label is not defined it will issue a warning.<sup>8</sup>

```
303 <*package>
304 \newcommand\STRcopy[2] [] {\expandafter\ifx\csname STR@#2\endcsname\relax
305 \message{STR warning: reference #2 undefined!}
306 \else\csname STR@#2\endcsname\fi}
307 </package>
308 <*ltxml.sty>
309 DefConstructor('\STRcopy[]{}', "<omdoc:ref xref='#1#2'/>");
310 </ltxml.sty>
```

**\STRsemantics** if we have a presentation form and a semantic form, then we can use

---

<sup>8</sup>EdNOTE: MK: we need to do something about the ref!

```

311 <*package>
312 \newcommand\STRsemantics[3][\def\test{#1}\ifx\test\empty\STRlabeldef{#1}{#2}\fi}
313 </package>
314 <*ltxml.sty>
315 DefConstructor('\STRsemantics[]{}', sub {
316   my($document,$label,$ignore,$object)=@_;
317   $document->absorb($object);
318   $document->addAttribute('xml:id'=>ToString($label)) if $label; });
319 </ltxml.sty>#

\STRlabeldef This is the macro that does the actual labeling. Is it called inside \STRlabel
320 <*package>
321 \def\STRlabeldef#1{\expandafter\gdef\csname STR@#1\endcsname}
322 </package>
323 <*ltxml.sty>
324 DefMacro('\STRlabeldef{}{}', "");
325 </ltxml.sty>

```

## 6.6 Colors

blue, red, green, magenta We will use the following abbreviations for colors from color.sty

```

326 <*package>
327 \def\black#1{\textcolor{black}{#1}}
328 \def\gray#1{\textcolor{gray}{#1}}
329 \def\blue#1{\textcolor{blue}{#1}}
330 \def\red#1{\textcolor{red}{#1}}
331 \def\green#1{\textcolor{green}{#1}}
332 \def\cyan#1{\textcolor{cyan}{#1}}
333 \def\magenta#1{\textcolor{magenta}{#1}}
334 \def\brown#1{\textcolor{brown}{#1}}
335 \def\yellow#1{\textcolor{yellow}{#1}}
336 \def\orange#1{\textcolor{orange}{#1}}
337 </package>

For the LATEXML bindings, we go a generic route, we replace \blue{#1} by
{\@omdoc@color{blue}\@omdoc@color@content{#1}}.

338 <*ltxml.sty>
339 sub omdocColorMacro {
340   my ($color, @args) = @_;
341   my $tok_color = TokenizeInternal($color);
342   (T_BEGIN, T_CS('\@omdoc@color'), T_BEGIN, $tok_color->unlist,
343    T_END, T_CS('\@omdoc@color@content'), T_OTHER(''), $tok_color->unlist, T_OTHER('')),
344   T_BEGIN, $args[1]->unlist, T_END, T_END); }
345 DefMacro('\@omdoc@color{}', sub { MergeFont(color=>$_[1]->toString); return; });#
346 </ltxml.sty>

```

Ideally, here we will remove the optional argument and have a conversion module add the attribute at the end (or maybe add it just for math?) or, we can take the attributes for style from the current font ?



```

347 <*ltxml.sty>
348 DefConstructor(' \@omdoc@color@content[] {}',
349   "?#isMath(#2)(<ltx:text ?#1(style='color:#1')()>#2</ltx:text>");
350 foreach my $color(qw(black gray blue red green cyan magenta brown yellow orange)) {
351   DefMacro("\\".$color.'{}', sub { omdocColorMacro($color, @_); }); }#$
352 </ltxml.sty>

```

## 6.7 L<sup>A</sup>T<sub>E</sub>X Commands we interpret differently

The reinterpretations are quite simple, we either disregard presentational markup or we re-interpret it in terms of OMDoc.

```

353 <*ltxml.sty>
354 DefConstructor('\newpage', '');
355 </ltxml.sty>

```

## 6.8 Miscellaneous

Some shortcuts that use math symbols but are not mathematical at all; in particular, they should not be translated by L<sup>A</sup>T<sub>E</sub>XML.

```

356 <*package>
357 \newcommand\hateq{\ensuremath{\hat{=}}\xspace}
358 \newcommand\hatequiv{\ensuremath{\hat{=}}\equiv\xspace}
359 \newcommand\textleadsto{\ensuremath{\leadsto}\xspace}
360 </package>
361 <*ltxml.sty>
362 DefMacro('\hateq', '\@hateq\xspace');
363 DefConstructor('\@hateq', "\x{2259}");
364 DefMacro('\hatequiv', '\@hatequiv\xspace');
365 DefConstructor('\@hatequiv', "\x{2A6F}");
366 DefMacro('\textleadsto', '\@textleadsto\xspace');
367 DefConstructor('\@textleadsto', "\x{219D}");
368 </ltxml.sty>

```

## 6.9 Leftovers

```

369 <*package>
370 \newcommand\baseURI[2] [] {}
371 </package>
372 <*ltxml.sty>
373 DefMacro('\baseURI []Semiverbatim', sub {
374   my $baselocal = ToString(Digest($_[1]));
375   $baselocal = abs_path($baselocal) unless $baselocal =~ /^(w+):\/\//;
376   AssignValue('BASELOCAL'=>$baselocal, 'global');
377   AssignValue('URLBASE'=>ToString(Digest($_[2])), 'global');
378 });
379 </ltxml.sty>%$

```

EdN:9                   <sup>9</sup> and finally, we need to terminate the file with a success mark for perl.  
380 `\ltxml.sty | ltxml.cls`1;

---

<sup>9</sup>EDNOTE: this should be handled differently, omdoc.sty should include url and give a new macro for it, which we then use in omdoc

## References

- [DCM03] The DCMI Usage Board. *DCMI Metadata Terms*. DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: <http://dublincore.org/documents/dcmi-terms/>.
- [Koh06] Michael Kohlhase. *OMDOC – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: <http://omdoc.org/pubs/omdoc1.2.pdf>.
- [Koh15a] Michael Kohlhase. *dcm.sty: An Infrastructure for marking up Dublin Core Metadata in L<sup>A</sup>T<sub>E</sub>X documents*. Tech. rep. Comprehensive T<sub>E</sub>X Archive Network (CTAN), 2015. URL: <http://www.ctan.org/tex-archive/macros/latex/contrib/stex/dcm/dcm.pdf>.
- [Koh15b] Michael Kohlhase. *omtext: Semantic Markup for Mathematical Text Fragments in L<sup>A</sup>T<sub>E</sub>X*. Tech. rep. Comprehensive T<sub>E</sub>X Archive Network (CTAN), 2015. URL: <http://www.ctan.org/tex-archive/macros/latex/contrib/stex/omtext/omtext.pdf>.
- [Koh15c] Michael Kohlhase. *sref.sty: Semantic Crossreferencing in L<sup>A</sup>T<sub>E</sub>X*. Tech. rep. Comprehensive T<sub>E</sub>X Archive Network (CTAN), 2015. URL: <http://www.ctan.org/tex-archive/macros/latex/contrib/stex/sref/sref.pdf>.
- [Koh15d] Michael Kohlhase. *statements.sty: Structural Markup for Mathematical Statements*. Tech. rep. Comprehensive T<sub>E</sub>X Archive Network (CTAN), 2015. URL: <http://www.ctan.org/tex-archive/macros/latex/contrib/stex/statements/statements.pdf>.
- [sTeX] *Semantic Markup for L<sup>A</sup>T<sub>E</sub>X*. Project Homepage. URL: <http://trac.kwarc.info/sTeX/> (visited on 02/22/2011).