

||||| HEAD ||||| HEAD

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

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

October 5, 2014

## **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

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 [Koh13c] for details.

`extrefs` `omdoc.cls` accepts all options of the `omdoc.sty` (see ) 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 XMRref 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 <code>id</code> key is used to give an identifier to the <code>omdoc</code> element resulting from the $\text{\LaTeX}$ transformation.
<code>id</code>	
<code>omgroup</code>	The structure of the document is given by the <code>omgroup</code> environment just like in OMDoc. In the $\text{\LaTeX}$ route, the <code>omgroup</code> environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of <code>omgroup</code> environments. Correspondingly, the <code>omgroup</code> environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the <code>omgroup</code> . The optional metadata argument has the keys <code>id</code> for an identifier, <code>creators</code> and <code>contributors</code> for the Dublin Core metadata [DCM03]; see [Koh13a] for details of the format. The <code>short</code> allows to give a short title for the generated section.
<code>id</code>	
<code>creators</code>	
<code>contributors</code>	
<code>short</code>	$\text{\LaTeX}$ automatically computes the sectioning level, from the nesting of <code>omgroup</code> environments. But sometimes, we want to skip levels (e.g. to use a subsection* as an introduction for a chapter). Therefore the <code>omdoc</code> package provides a variant <code>blindomgroup</code> 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 <code>blindomgroup</code> environment is useful e.g. for creating frontmatter at the correct level. Example ?? shows a typical setup for the outer document structure of a book with parts and chapters. We use two levels of <code>blindomgroup</code> :
<code>blindomgroup</code>	<ul style="list-style-type: none"> <li>• The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This <code>blindomgroup</code> makes sure that the introductory remarks become a “chapter” instead of a “part”.</li> <li>• The inner one groups the frontmatter<sup>1</sup> and makes the preface of the book a section-level construct. Note that here the <code>display=flow</code> on the <code>omgroup</code> environment prevents numbering as is traditional for prefaces.</li> </ul>
<code>\currentsectionlevel</code>	
<code>\CurrentSectionLevel</code>	The <code>\currentsectionlevel</code> macro supplies the name of the current sectioning level, e.g. “chapter”, or “subsection”. <code>\CurrentSectionLevel</code> is the capitalized variant. They are useful to write something like “In this <code>\currentsectionlevel</code> , we will...” in an <code>omgroup</code> environment, where we do not know which sectioning level we will end up.

## 2.3 Ignoring Inputs

<code>ignore</code>	The <code>ignore</code> 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
<code>showignores</code>	<code>showignores</code> option is given to the <code>omdoc</code> class or <code>package</code> . But in the generated OMDoc result, the body is marked up with a <code>ignore</code> element. This is useful in two situations. For

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

---

<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

**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 [Koh13d] 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 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  $\text{\LaTeXML}$  generate the correct reference.

`\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{⟨something⟩}` writes `⟨something⟩` in blue. The macros `\red` `\green`, `\cyan`,  
...

---

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

`\black` `\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

## 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*{\PassOptionsToClass{\CurrentOption}{article}}
10 \ProcessOptions
11 </cls>
12 <*ltxml.cls>
13 # -*- CPERL -*-
14 package LaTeXML::Package::Pool;
15 use strict;
16 use LaTeXML::Package;
17 use LaTeXML::Util::Pathname;
18 use Cwd qw(abs_path);
19 DeclareOption('report',sub {PassOptions('omdoc','sty',ToString(Digest(T_CS('\CurrentOption'))))})
20 DeclareOption('book',sub {PassOptions('omdoc','sty',ToString(Digest(T_CS('\CurrentOption'))))});
21 DeclareOption('showignores',sub {PassOptions('omdoc','sty',ToString(Digest(T_CS('\CurrentOption'))))});
22 DeclareOption('extrefs',sub {PassOptions('sref','sty',ToString(Digest(T_CS('\CurrentOption'))))});
23 DeclareOption(undef,sub {PassOptions('article','cls',ToString(Digest(T_CS('\CurrentOption'))))});
24 ProcessOptions();
25 </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.

```
26 <*cls>
27 \LoadClass{\omdoc@class}
28 \RequirePackage{etoolbox}
29 \RequirePackage{omdoc}
30 </cls>
31 <*ltxml.cls>
```



```

32 LoadClass('article');
33 RequirePackage('sref');
34 \ltxml.cls

```

## 5.2 Setting up Namespaces and Schemata for LaTeXXML

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

```

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

```

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

```

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

```

54 \ltxml.cls
55 RequirePackage('omdoc');
56 \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.

```

57 \cls
58 \let\orig@document=\document
59 \srefaddidkey{document}
60 \renewcommand{\document}[1][\metasetkeys{document}{#1}\orig@document]

```

---

<sup>6</sup>EdNOTE: reword

```

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

```

## 6 Implementation: OMDoc Package

### 6.1 Package Options

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

```

93 <*ltxml.sty>
94 package LaTeXML::Package::Pool;
95 use strict;
96 use LaTeXML::Package;
97 use Cwd qw(abs_path);
98 </ltxml.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

EdN:7

to true (otherwise they stay false).<sup>7</sup>

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

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

```
118 <*package>
119 \RequirePackage{sref}
120 \RequirePackage{xspace}
121 \RequirePackage{comment}
122 </package>
123 <*ltxml.sty>
124 RequirePackage('sref');
125 RequirePackage('xspace');
126 RequirePackage('omtext');
127 </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`

```
128 <*package>
129 \def\currentsectionlevel{document\xspace}%
130 \def\Currentsectionlevel{Document\xspace}%
131 </package>
132 <*ltxml.sty>
133 DefMacro('currentsectionlevel','\@currentsectionlevel\xspace');
```

---

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

```

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

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

\omgroup@c1 Convenience macro: defines the \currentsectionlevel macro from the keywords
in the arguments
154 <*package>
155 \newcommand\omgroup@c1[2]{%
156 \def\currentsectionlevel{#1\xspace}%
157 \def\Currentsectionlevel{#2\xspace}}

\omgroup@nonum convenience macro: \omgroup@nonum{<level>}{<title>} makes an unnumbered sectioning
with title <title> at level <level>.
158 \newcommand\omgroup@nonum[2]{%
159 \ifx\hyper@anchor\@undefined\else\phantomsection\fi%
160 \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.
161 \newcommand\omgroup@num[2]{\sref@label@id\omdoc@sect@Name \@nameuse{the#1}}%
162 \ifx\omgroup@short\@empty\@nameuse{#1}{#2}%
163 \else\@ifundefined{rdfmeta@sectioning}{\@nameuse{#1}[\omgroup@short]{#2}}%
164 {\@nameuse{rdfmeta@#2@old}[\omgroup@short]{#2}}\fi}
165 </package>

omgroup
166 <*package>
167 \srefaddidkey{omgroup}

```

```

168 \addmetakey{omgroup}{creators}
169 \addmetakey{omgroup}{date}
170 \addmetakey{omgroup}{contributors}
171 \addmetakey{omgroup}{type}
172 \addmetakey*{omgroup}{short}
173 \addmetakey*{omgroup}{display}

\at@begin@omgroup we define a switch for numbering lines and a hook for the beginning of groups:
The \at@begin@omgroup macro allows customization. It is run at the beginning
of the omgroup, i.e. after the section heading.

174 \newif\if@num@@numtrue
175 \newif\if@frontmatter\@frontmatterfalse
176 \newif\if@backmatter\@backmatterfalse
177 \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.

```

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

```

now the environment itself.

```

191 \newenvironment{omgroup}[2] []% keys, title
192 {\metasetkeys{omgroup}{#1}\sref@target%
193 \ifx\omgroup@display\st@flow\@numfalse\fi
194 \if@frontmatter\@numfalse\fi

```

now we construct the entries for the table of contents. They depend on whether modules.sty and hyperref.sty are loaded.

```

195 \ifx\imported@modules\@undefined% modules.sty loaded?
196 \ifx\hyper@anchor\@undefined% hyperref.sty loaded?
197 \def\addcontentsline##1##2##3{\addtocontents{##1}{\protect\contentsline{##2}{##3}{\thepage}}}
198 \else\def\addcontentsline##1##2##3{%
199 \addtocontents{##1}{\protect\contentsline{##2}{##3}{\thepage}{\@currentHref}}}
200 \fi% hyperref.sty loaded
201 \else% modules.sty loaded?
202 \ifx\hyper@anchor\@undefined% hyperref.sty loaded?
203 \def\addcontentsline##1##2##3{%
204 \addtocontents{##1}{\protect\contentsline{##2}{\string\importmodules{\imported@modules}##3}{\th
205 \else\def\addcontentsline##1##2##3{%
206 \addtocontents{##1}{\protect\contentsline{##2}{\string\importmodules{\imported@modules}##3}{\th

```

```

207 \fi% hypreref.sty loaded
208 \fi% modules.sty loaded

now we only need to construct the right sectioning depending on the value of
\section@level.

209 \advance\section@level by 1
210 \ifcase\section@level%
211 \or\omdoc@sectioning[name=part,Name=Part,clear,num]{part}{#2}%
212 \or\omdoc@sectioning[name=chapter,Name=Chapter,clear,num]{chapter}{#2}%
213 \or\omdoc@sectioning[name=section,Name=Section,num]{section}{#2}%
214 \or\omdoc@sectioning[name=subsection,Name=Subsection,num]{subsection}{#2}%
215 \or\omdoc@sectioning[name=subsubsection,Name=Subsubsection,num]{subsubsection}{#2}%
216 \or\omdoc@sectioning[name=paragraph,Name=Paragraph,ref=this paragraph]{paragraph}{#2}%
217 \or\omdoc@sectioning[name=subparagraph,Name=Subparagraph,ref=this subparagraph]{subparagraph}{#2}%
218 \fi% \ifcase
219 \at@begin@omgroup[#1]\section@level{#2}}% for customization
220 {\advance\section@level by -1}
221 \</package>
222 \<!xml.sty>
223 DefEnvironment('{omgroup} OptionalKeyVals:omgroup {}',
224               "<omdoc:omgroup layout='sectioning'"
225               . " ?&GetKeyVal(#1,'id')(xml:id='&GetKeyVal(#1,'id')')()"
226               . " ?&GetKeyVal(#1,'type')(type='&GetKeyVal(#1,'type')')()>\n"
227               . " <dc:title>#2</dc:title>\n"
228               . " #body\n"
229               . " </omdoc:omgroup>");
230 \</!xml.sty>

```

### 6.3 Front and Backmatter

Index markup is provided by the `omtext` package [Koh13b], so in the `omdoc` package we only need to supply the corresponding `\printindex` command, if it is not already defined

```

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

```

`\tableofcontents` The table of contents already exists in  $\text{\LaTeX}$ , so we only need to provide a  $\text{\LaTeX}$ XML binding for it.

```

237 \<!xml.sty>
238 DefConstructor('\tableofcontents',
239               "<omdoc:tableofcontents level='&ToString(&CounterValue('tocdepth'))'/>");
240 \</!xml.sty>

```

The case of the `\bibliography` command is similar

`\bibliography`

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

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

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

## 6.4 Ignoring Inputs

`ignore`

```
277 <*package>
278 \ifshow@ignores
```

```

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

```

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

```

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

```

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

```

309 \<*package>
310 \newcommand\STRsemantics[3] [] {#2\def\@test{#1}\ifx\@test\@empty\STRlabeldef{#1}{#2}\fi}
311 \</package>
312 \<*lxml.sty>
313 DefConstructor('\STRsemantics[]{}{}', sub {

```

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



```

314 my($document,$label,$ignore,$object)=@_;
315 $document->absorb($object);
316 $document->addAttribute('xml:id'=>ToString($label)) if $label; });
317 </ltxml.sty>#

```

`\STRlabeldef` This is the macro that does the actual labeling. Is it called inside `\STRlabel`

```

318 <*package>
319 \def\STRlabeldef#1{\expandafter\gdef\csname STR@#1\endcsname}
320 </package>
321 <*ltxml.sty>
322 DefMacro(' \STRlabeldef{}{}', "");
323 </ltxml.sty>

```

## 6.6 Colors

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

```

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

```

For the  $\text{\LaTeX}$ ML bindings, we go a generic route, we replace `\blue{#1}` by `{\@omdoc@color{blue}\@omdoc@color@content{#1}}`.

```

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

```

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

```

```
350 </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.

```
351 <*ltxml.sty>
352 DefConstructor('newpage', '');
353 </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>X<sub>M</sub>L.

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

## 6.9 Leftovers

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

EdN:9 and finally, we need to terminate the file with a success mark for perl.

```
378 <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>.
- [Koh13a] 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), 2013. URL: <http://www.ctan.org/tex-archive/macros/latex/contrib/stex/dcm/dcm.pdf>.
- [Koh13b] 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), 2013. URL: <http://www.ctan.org/tex-archive/macros/latex/contrib/stex/omtext/omtext.pdf>.
- [Koh13c] 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), 2013. URL: <http://www.ctan.org/tex-archive/macros/latex/contrib/stex/sref/sref.pdf>.
- [Koh13d] Michael Kohlhase. *statements.sty: Structural Markup for Mathematical Statements*. Tech. rep. Comprehensive T<sub>E</sub>X Archive Network (CTAN), 2013. 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).