

dcm.sty: An Infrastructure for marking up Dublin Core Metadata in L^AT_EX documents*

Michael Kohlhase John Doe

April 27, 2014

Contents

*Version v0.3 (last revised 2012/09/23)

EdN:1
EdN:2

1 Introduction

The `dcm` package allows mark up Dublin Core Metadata [**DCMI:dmt03**] in \LaTeX documents so that it can be harvested by automated tools or exported to PDF¹. This package allows to attribute authorship to arbitrary text fragments.²

2 The User Interface

2.1 Package Options

`showmeta` The `dcm` package takes a single option: `showmeta`. If this is set, then the metadata keys are shown (see [**Kohlhase:metakeys:ctan**] for details and customization options).

2.2 The DC Metadata Block

`DCmetadata` The `dcm` provides the environment `DCmetadata` for Dublin Core Metadata Blocks. `DCmetadata` defines local macros for the specifying the relevant Dublin Core metadata fields and takes an optional argument that specifies the presentation of the metadata block, see Figure ?? for an example which would generate the title block for the `dcm` package. Let us now come to the macros themselves

`\DCMcreators` The `\DCMcreators` and `\DCMcontributors` macros are used to specify the authors and contributors to a text fragments. These macros take one argument, the authorship of a document specified in terms of `ids` of persons specified via `\WAperson` before. They can occur multiply in a metadata block.

`\DCMtitle` The `\DCMtitle` macro takes one argument, the

`\DCMshorttitle` The `\DCMshorttitle` macro takes one argument, the

`\DCMsubject` The `\DCMsubject` macro takes one argument, the

`\DCMdescription` The `\DCMdescription` macro takes one argument, the

`\DCMpublisher` The `\DCMpublisher` macro takes one argument, the

`\DCMdate` The `\DCMdate` macro takes one argument, the

`\DCMtype` The `\DCMtype` macro takes one argument, the

`\DCMidentifier` The `\DCMidentifier` macro takes two arguments, the first one is the identification system, and the second one the identifier string itself.

`\DCMsource` The `\DCMsource` macro takes one argument, the

`\DCMlanguage` The `\DCMlanguage` macro takes one argument, the

`\DCMrelation` The `\DCMrelation` macro takes one argument, the

`\DCMrights` The `\DCMrights` macro takes one argument, the

`\DCMlicense` The `\DCMlicense` macro takes one argument, the

`\DCMabstract` The `\DCMabstract` macro takes one argument, the

`\DCMlicensenotice` The `\DCMlicensenotice` macro takes one argument, the

`\DCMcopyrightnotice` The `\DCMcopyrightnotice` macro takes one argument, the

`\DCMcclicense` The `\DCMcclicense` macro

¹EDNOTE: This still needs to be implemented, see <http://www.wlug.org.nz/PdfLatexNotes> for details

²EDNOTE: continue

```

\attribution
\noncommercial
\sharealike
\noderivativeworks

```

```

\begin{DCmetadata}[maketitle]
  \DCMtitle{An Infrastructure for marking up Dublin Core Metadata in
    {\LaTeX} documents\thanks{Version {\fileversion}
      (last revised {\filedate})}}
  \DCMcreators{miko,jdoe}
  \DCMdate{\today}
  \DCMcopyrightnotice{2008}{Michael Kohlhase}
  \DCMlicense{Copyright (c) 2008 Michael Kohlhase, all rights
    reserved. This file is released under the LaTeX Project Public
    License (LPPL)}
  \DCMabstract{The {\texttt{dcm}} package allows mark up Dublin
    Core Metadata in {\LaTeX} documents that can be harvested by
    automated tools or exported to PDF, while at the same time
    generating conventional title information.}
\end{DCmetadata}

```

Example 1: The DC Metadata block for the dcm package documentation

2.3 DCM Metadata Block Styles

The `DCmetadata` environment takes an optional argument that specifies the style the metadata block is rendered in. The `dcm` package supplies two styles: `maketitle` and `titlepage`. The former uses the `\maketitle` macro from the calling class to assemble a title, whereas the latter builds a title page from scratch. The title block of this documentation has been created by the `maketitle` style.

To add a further metadata block style $\langle sty \rangle$, we simply have to supply a `\dcm@ $\langle sty \rangle$ @block` macro that expands to the intended presentation. This macro does not take any arguments, but can use the internal token registers defined by the `DCmetadata` environment. Generally, for any of the metadata commands `\DCM $\langle md \rangle$` defined in `?user.dcm.mdblock?` there is a token register `\dcm@ $\langle md \rangle$` that contains the value specified in the key.

2.4 Configuration

The `dcm` package provides a set of macros that customize (e.g. for multiple languages) the generated content.

```

\dcm@abstract@heading
\dcm@creators@heading
\dcm@contributors@connector
\dcm@chapter@heading
\dcm@section@heading
\dcm@subsection@heading
\dcm@subsubsection@heading

```

Macro	Default
<code>\dcm@abstract@heading</code>	Abstract
<code>\dcm@creators@heading</code>	Author(s)
<code>\dcm@contributors@connector</code>	with contributions from
<code>\dcm@chapter@heading</code>	Chapter
<code>\dcm@section@heading</code>	Section
<code>\dcm@subsection@heading</code>	Subsection
<code>\dcm@subsubsection@heading</code>	Subsubsection

3 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:online`].

1. none reported yet

4 The Implementation

The `dcm` package generates two files: the `LATEX` package (all the code between `<*package>` and `</package>`) and the `LATEXML` bindings (between `<*ltxml>` and `</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.

4.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. They all come with their own conditionals that are set by the options.

```

1 <*package>
2 \DeclareOption{showmeta}{\PassOptionsToPackage{\CurrentOption}{metakeys}}
3 \ProcessOptions
4 </package>
```

The first measure is to ensure that the `KeyVal` package is loaded (in the right version). For `LATEXML` we also initialize the package inclusions.

```

5 <*package>
6 \RequirePackage{workaddress}
7 \RequirePackage[sectioning]{rdfmeta}
8 </package>
9 <*ltxml>
10 # -*- CPERL -*-
11 package LaTeXML::Package::Pool;
12 use strict;
13 use LaTeXML::Global;
14 use LaTeXML::Package;
15 RequirePackage('rdfmeta');
```

```

16 RequirePackage('workaddress');
17 </ltxml>

    Furthermore, we need a couple of helper functions for the

18 <*ltxml>
19 sub FishOutMetadata {
20   my ($document,$keyvals)=@_;
21   foreach my $role(qw(creators contributors)) {
22     my $idlist_string=getKeyValue_noDelim($keyvals,$role);
23     my @ids = split(/,\s*/, $idlist_string);
24     foreach my $id(@ids) {
25       my $name = LookupValue('DCM_'. $id.'_name');
26       if ($name) {
27         my $prop_role = $role;
28         chop $prop_role if $prop_role;
29         $document->insertElement("dc:$prop_role",$name) if $role;
30       } else {print STDERR "Warning: no $role with 'id' $id !\n";}
31     }
32   }
33   return;}#$
34 </ltxml>

```

4.2 The DC Metadata Block

Then we make an environment for defining the metadata. Note that since we have defined the `omdoc:metadata` element to auto-open and auto-close, we do not have to (and should not for that matter) supply it in the `DCmetadata` element.

`DCmetadata`

```

35 <*package>
36 \newenvironment{DCmetadata}[1][]{%
37 {\def\@style{#1}}% to set the way things are presented.
38 {\@ifundefined{dcm@\@style @block}%
39 {\message{style {\@style} not defined}}%
40 {\csname dcm@\@style @block\endcsname}}
41 </package>
42 <*ltxml>
43 DefEnvironment('{DCmetadata}[1]', "<omdoc:metadata>#body</omdoc:metadata>");
44 </ltxml>

```

Here come the constructors, most of them are relatively straightforward

`\DCMcreators` the `\DCMcreators` macro checks whether all ids are defined.

```

45 <*package>
46 \def\DCMcreators#1{\@for\@I:=#1\do{\wa@ref@test{person}\@I{id}}
47 \gdef\dcm@creators{#1}}
48 </package>
49 <*ltxml>
50 DefConstructor('{\DCMcreators}',sub{

```

```

51 my ($document,$args,%properties) = @_;
52 my $keyval = LaTeXML::KeyVals->new('wa@person',T_BEGIN,T_END,('creators'=>$args));
53 FishOutMetadata($document,$keyval);
54 return;});
55 </ltxml>

```

`\DCMcontributors` the `\DCMcontributors` macro also checks whether all ids are defined.

```

56 <*package>
57 \def\DCMcontributors#1{\@for\@I:=#1\do{\wa@ref@test{person}\@I{id}}}%
58 \def\dcm@contributors{#1}}
59 </package>
60 <*ltxml>
61 DefConstructor('DCMcontributors{',sub{
62 my ($document,$args,%properties) = @_;
63 my $keyval = LaTeXML::KeyVals->new('wa@person',T_BEGIN,T_END,('contributors'=>$args));
64 FishOutMetadata($document,$keyval);
65 return;});
66 </ltxml>

```

`\DCMtitle`

```

67 <*package>
68 \def\DCMtitle#1{\def\dcm@title{#1}\providecommand{\dcm@shorttitle}{#1}}
69 </package>
70 <*ltxml>
71 DefConstructor('DCMtitle{', "<dc:title>#1</dc:title>");
72 </ltxml>

```

`\DCMsubtitle`

```

73 <*package>
74 \def\dcm@subtitle{}
75 \def\DCMsubtitle#1{\def\dcm@subtitle{#1}}
76 </package>

```

`\DCMshorttitle`

```

77 <*package>
78 \def\dcm@shorttitle{}
79 \def\DCMshorttitle#1{\def\dcm@shorttitle{#1}}
80 </package>

```

`\DCMsubject`

```

81 <*package>
82 \def\DCMsubject#1{\def\dcm@subject{#1}}
83 </package>
84 <*ltxml>
85 DefConstructor('DCMsubject{', "<dc:subject>#1</dc:subject>");
86 </ltxml>

```

`\DCMdescription`

```

87 <*package>

```

```

88 \long\def\DCMdescription#1{\long\def\dcm@description{#1}}
89 \end{package}
90 \end{xml}
91 DefConstructor('\DCMdescription{#1}', "<dc:description>#1</dc:description>");
92 \end{xml}

```

\DCMpublisher

```

93 \begin{package}
94 \def\DCMpublisher#1{\def\dcm@publisher{#1}}
95 \end{package}
96 \end{xml}
97 DefConstructor('\DCMpublisher{#1}', "<dc:publisher>#1</dc:publisher>");
98 \end{xml}

```

EdN:3

\DCMdate the \DCMdate uses \today as a default³

```

99 \begin{package}
100 \def\dcm@date{\today}
101 \def\DCMdate#1{\def\dcm@date{#1}}
102 \end{package}
103 \end{xml}
104 DefConstructor('\DCMdate{#1}', "<dc:date>#1</dc:date>");
105 \end{xml}

```

\DCMtype

```

106 \begin{package}
107 \def\DCMtype#1{\def\dcm@type{#1}}
108 \end{package}
109 \end{xml}
110 DefConstructor('\DCMtype{#1}', "<dc:type>#1</dc:type>");
111 \end{xml}

```

\DCMidentifier

```

112 \begin{package}
113 \def\DCMidentifier#1#2{\def\dcm@scheme{#1}\def\dcm@identifier{#2}}
114 \end{package}
115 \end{xml}
116 DefConstructor('\DCMidentifier{#1}{#2}', "<dc:identifier scheme='#{#1}'>#{#2}</dc:identifier>");
117 \end{xml}

```

\DCMsource

```

118 \begin{package}
119 \def\DCMsource#1{\def\dcm@source{#1}}
120 \end{package}
121 \end{xml}
122 DefConstructor('\DCMsource{#1}', "<dc:source>#1</dc:source>");
123 \end{xml}

```

³EdNOTE: @DEYAN: do that in latexml

\DCMLanguage

```
124 <*package>
125 \def\DCMLanguage#1{\def\dcm@language{#1}}
126 </package>
127 <*ltxml>
128 DefConstructor('\DCMLanguage{','<dc:language>#1</dc:language>');
129 </ltxml>
```

\DCMrelation

```
130 <*package>
131 \def\DCMrelation#1{\def\dcm@relation{#1}}
132 </package>
133 <*ltxml>
134 DefConstructor('\DCMrelation{','<dc:relation>#1</dc:relation>');
135 </ltxml>
```

\DCMrights

```
136 <*package>
137 \def\DCMrights#1{\long\def\dcm@rights{#1}}
138 </package>
139 <*ltxml>
140 DefConstructor('\DCMrights{','<dc:rights>#1</dc:rights>');
141 </ltxml>
```

\DCMlicense

```
142 <*package>
143 \def\DCMlicense#1{\def\dcm@license{#1}}
144 </package>
```

\DCMlicensenotice here we have a default

```
145 <*package>
146 \def\dcm@license{All rights reserved}
147 \def\DCMlicensenotice#1{\long\def\dcm@license{\[1ex]License: #1}}
148 </package>
149 <*ltxml>
150 DefMacro('\DCMlicensenotice{','\DCMrights{#1}');
151 </ltxml>
```

\DCMcopyrightnotice

```
152 <*package>
153 \def\DCMcopyrightnotice#1#2{\DCMrights{Copyright {\copyright} #1: #2}}
154 </package>
155 <*ltxml>
156 DefMacro('\DCMcopyrightnotice{','\DCMrights{Copyright {\copyright} #1: #2}');
157 </ltxml>
```

\cclicense

```
158 <*package>
159 \def\cclicense#1{\def\attribution{\def\dcm@by{yes}}
```



```

160 \def\noncommercial{\def\dcm@nc{yes}}
161 \def\sharealike{\def\dcm@sharealike{yes}}
162 \def\noderivativeworks{\def\dcm@derivatives{no}}
163 \end{package}
164 \end{*xml}
165 DefConstructor('cc:license', "<cc:license>#1</cc:license>");
166 DefConstructor('attribution', "<cc:attribution/>");
167 DefConstructor('noncommercial', "<cc:noncommercial/>");
168 DefConstructor('sharealike', "<cc:sharealike/>");
169 DefConstructor('noderivativeworks', "<cc:noderivativeworks>");
170 \end{*xml}

```

\DCMabstract

```

171 \end{package}
172 \long\def\DCMabstract#1{\long\def\dcm@abstract{#1}}
173 \end{package}
174 \end{*xml}
175 DefConstructor('DCMabstract', "<dc:description>#1</dc:description>");
176 \end{*xml}

```

4.3 DCM Block Styles

We now define various commonly used styles.

\dcm@titlepage@block This style builds up a title page from scratch

```

177 \end{package}
178 \def\dcm@titlepage@block{\begin{titlepage}
179   \null\vfil\vskip 60\p@
180   \begin{center}
181     \ifx\dcm@title\@empty
182       \PackageWarning{dcm}{No title specified}{\LARGE Add title here\par}
183     \else\LARGE \dcm@title \par\fi
184     \ifx\dcm@subtitle\@empty
185       \vskip 3em\LARGE \dcm@subtitle \par\vskip 3em
186     \else\large\lineskip .75em\WAAuthorblock\dcm@creators\vskip 1.5em\fi
187     \ifx\dcm@date\@empty
188       \PackageWarning{dcm}{No date specified}{\large\today\par}
189     \else\large\dcm@date\par\vskip 2em\fi
190   \end{center}\vskip2em
191   \ifx\dcm@abstract\@empty
192     \PackageWarning{dcm}{No Abstract specified}\else
193     \begin{quote}\textbf{\dcm@abstract@heading:\dcm@abstract}\end{quote}\fi
194     \vskip 2em\par\vfil\noindent
195     {\small\noindent\dcm@rights\dcm@license}
196   \end{titlepage}}
197 \end{package}

```

\dcm@maketitle@block This style makes use of the title facility of the document class.

```

198 \end{package}

```

```

199 \def\dcm@maketitle@block{\def\@title{\dcm@title\ifx\dcm@subtitle\empty\else\newline\dcm@subtitl
200 \def\@author{\WAauthorblock\dcm@creators}%
201 \def\@date{\dcm@date}\maketitle}
202 \end{package}

```

We have to make sure that the DCM metadata commands have IDs, so that we do not get duplicates.

```

203 \begin{xml}
204 \tag{dc:description,afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1};
205 \tag{dc:date,afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1};
206 \tag{dc:creator,afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1};
207 \tag{dc:contributor,afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1};
208 \tag{dc:title,afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1};
209 \tag{dc:subject,afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1};
210 \tag{dc:publisher,afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1};
211 \tag{dc:type,afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1};
212 \tag{dc:identifier,afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1};
213 \tag{dc:language,afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1};
214 \tag{dc:rights,afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1};
215 \end{xml}

```

4.4 Dealing with ISO Dates

The first step is to build a macro for making ISO dates.⁴

```

216 \begin{xml}RawTeX{
217 \begin{package} | \end{package}
218 \def\ISOtimestamp{\count1=\time\divide\count1 by 60 % hours
219 \count2=\count1\multiply\count2 by 60% minutes in \count1 hours
220 \count3=\time\advance\count3 by -\count2 % minutes
221 \the\year -\ifnum\month>9\else0\fi\the\month-\ifnum\day>9\else0\fi\the\day
222 T\ifnum\count1>9\else0\fi\the\count1:\ifnum\count3>9\else0\fi\the\count3:00Z}
223 \end{package} | \end{xml}
224 \end{xml}';

```

4.5 Configuration

```

225 \begin{package}
226 \def\dcm@abstract@heading{Abstract}
227 \def\dcm@creators@heading{Author(s)}
228 \def\dcm@contributors@connector{with contributions from}
229 \def\dcm@chapter@heading{Chapter}
230 \def\dcm@section@heading{Section}
231 \def\dcm@subsection@heading{Subsection}
232 \def\dcm@subsubsection@heading{Subsubsection}
233 \def\dcm@paragraph@heading{Paragraph}
234 \end{package}

```

⁴EDNOTE: make better ltxml

4.6 Providing IDs for Elements

To provide default identifiers, we tag all elements that allow `xml:id` attributes by executing the `numberIt` procedure below.

```
235 <|txml>  
236 Tag('dc:title',afterOpen=>\&numberIt,afterClose=>\&locateIt);  
237 </txml>
```

4.7 Finale

Finally, we need to terminate the file with a success mark for perl.

```
238 <txml>1;
```