`dcm.sty`: An Infrastructure for marking up Dublin Core Metadata in LaTeX documents*

Michael Kohlhase John Doe

December 12, 2013

Contents

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

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 PDF[1]. This package allows to attribute authorship to arbitrary text fragments.[2]

EdN:1
EdN: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
\DCMcontributors

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

[1] EdNote: This still needs to be implemented, see http://www.wlug.org.nz/PdfLatexNotes for details

[2] EdNote: 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}
  \DCMlicensenotice{Copyright (c) 2008 Michael Kohlhase, all rights
     reserved. This file is released under the LaTeX Project Public
     License (LPPL)}
  \DCMabstract{The {\texttt{dcm}} packge 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

maketitle
titlepage

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.

\dcm@*@block

To add a further metadata block style ⟨*sty*⟩, we simply have to supply a `\dcm@`⟨*sty*⟩`@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`⟨*md*⟩ defined in ?user.dcm.mdblock? there is a token register `\dcm@`⟨*md*⟩ 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
\dcm@abstract@heading	Abstract
\dcm@creators@heading	Author(s)
\dcm@contributors@connector	with contributions from
\dcm@chapter@heading	Chapter
\dcm@section@heading	Section
\dcm@subsection@heading	Subsection
\dcm@subsubsection@heading	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.

```
⟨*package⟩
\DeclareOption{showmeta}{\PassOptionsToPackage{\CurrentOption}{metakeys}}
\ProcessOptions
⟨/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.

```
⟨*package⟩
\RequirePackage{workaddress}
\RequirePackage[sectioning]{rdfmeta}
⟨/package⟩
⟨*ltxml⟩
# -*- CPERL -*-
package LaTeXML::Package::Pool;
use strict;
use LaTeXML::Global;
use LaTeXML::Package;
RequirePackage('rdfmeta');
```

```
RequirePackage('workaddress');
⟨/ltxml⟩
```

Furthermore, we need a couple of helper functions for the

```
⟨*ltxml⟩
sub FishOutMetadata {
  my ($document,$keyvals)=@_;
  foreach my $role(qw(creators contributors)) {
    my $idlist_string=getKeyValue_noDelim($keyvals,$role);
    my @ids = split(/,\s*/, $idlist_string);
    foreach my $id(@ids) {
      my $name = LookupValue('DCM_'.$id.'_name');
      if ($name) {
        my $prop_role = $role;
chop $prop_role if $prop_role;
        $document->insertElement("dc:$prop_role",$name) if $role;
      } else {print STDERR "Warning: no $role with 'id' $id !\n";}
    }
  }
  return;}#$
⟨/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`

```
⟨*package⟩
\newenvironment{DCmetadata}[1][]%
{\def\@style{#1}}% to set the way things are presented.
{\@ifundefined{dcm@\@style @block}%
{\message{style {\@style} not defined}}%
{\csname dcm@\@style @block\endcsname}}
⟨/package⟩
⟨*ltxml⟩
DefEnvironment('{DCmetadata}[]', "<omdoc:metadata>#body</omdoc:metadata>");
⟨/ltxml⟩
```

Here come the constructors, most of them are relatively straightforward

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

```
⟨*package⟩
\def\DCMcreators#1{\@for\@I:=#1\do{\wa@ref@test{person}\@I{id}}
\gdef\dcm@creators{#1}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMcreators{}',sub{
```

```
  my ($document,$args,%properties) = @_;
  my $keyval = LaTeXML::KeyVals->new('wa@person',T_BEGIN,T_END,('creators'=>$args));
  FishOutMetadata($document,$keyval);
  return;});
⟨/ltxml⟩
```

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

```
⟨*package⟩
\def\DCMcontributors#1{\@for\@I:=#1\do{\wa@ref@test{person}\@I{id}}%
\def\dcm@contributors{#1}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMcontributors{}',sub{
  my ($document,$args,%properties) = @_;
  my $keyval = LaTeXML::KeyVals->new('wa@person',T_BEGIN,T_END,('contributors'=>$args));
  FishOutMetadata($document,$keyval);
  return;});
⟨/ltxml⟩
```

\DCMtitle

```
⟨*package⟩
\def\DCMtitle#1{\def\dcm@title{#1}\providecommand{\dcm@shorttitle}{#1}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMtitle{}',"<dc:title>#1</dc:title>");
⟨/ltxml⟩
```

\DCMsubtitle

```
⟨*package⟩
\def\dcm@subtitle{}
\def\DCMsubtitle#1{\def\dcm@subtitle{#1}}
⟨/package⟩
```

\DCMshorttitle

```
⟨*package⟩
\def\dcm@shorttitle{}
\def\DCMshorttitle#1{\def\dcm@shorttitle{#1}}
⟨/package⟩
```

\DCMsubject

```
⟨*package⟩
\def\DCMsubject#1{\def\dcm@subject{#1}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMsubject{}',"<dc:subject>#1</dc:subject>");
⟨/ltxml⟩
```

\DCMdescription

```
⟨*package⟩
```

```
\long\def\DCMdescription#1{\long\def\dcm@description{#1}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMdescription{}',"<dc:description>#1</dc:description>");
⟨/ltxml⟩
```

\DCMpublisher

```
⟨*package⟩
\def\DCMpublisher#1{\def\dcm@publisher{#1}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMpublisher{}',"<dc:publisher>#1</dc:publisher>");
⟨/ltxml⟩
```

EdN:3 \DCMdate the \DCMdate uses \today as a default[3]

```
⟨*package⟩
\def\dcm@date{\today}
\def\DCMdate#1{\def\dcm@date{#1}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMdate{}',"<dc:date>#1</dc:date>");
⟨/ltxml⟩
```

\DCMtype

```
⟨*package⟩
\def\DCMtype#1{\def\dcm@type{#1}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMtype{}',"<dc:type>#1</dc:type>");
⟨/ltxml⟩
```

\DCMidentifier

```
⟨*package⟩
\def\DCMidentifier#1#2{\def\dcm@scheme{#1}\def\dcm@identifier{#2}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMidentifier{}{}',"<dc:identifier scheme='#1'>#2</dc:identifier>");
⟨/ltxml⟩
```

\DCMsource

```
⟨*package⟩
\def\DCMsource#1{\def\dcm@source{#1}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMsource{}',"<dc:source>#1</dc:source>");
⟨/ltxml⟩
```

[3] EdNote: @DEYAN: do that in latexml

\DCMlanguage

```
⟨*package⟩
\def\DCMlanguage#1{\def\dcm@language{#1}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMlanguage{}',"<dc:language>#1</dc:language>");
⟨/ltxml⟩
```

\DCMrelation

```
⟨*package⟩
\def\DCMrelation#1{\def\dcm@relation{#1}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMrelation{}',"<dc:relation>#1</dc:relation>");
⟨/ltxml⟩
```

\DCMrights

```
⟨*package⟩
\def\DCMrights#1{\long\def\dcm@rights{#1}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMrights{}',"<dc:rights>#1</dc:rights>");
⟨/ltxml⟩
```

\DCMlicense

```
⟨*package⟩
\def\DCMlicense#1{\def\dcm@license{#1}}
⟨/package⟩
```

\DCMlicensenotice here we have a default

```
⟨*package⟩
\def\dcm@license{All rights reserved}
\def\DCMlicensenotice#1{\long\def\dcm@license{\\[1ex]License: #1}}
⟨/package⟩
⟨*ltxml⟩
DefMacro('\DCMlicensenotice{}','\DCMrights{#1}');
⟨/ltxml⟩
```

\DCMcopyrightnotice

```
⟨*package⟩
\def\DCMcopyrightnotice#1#2{\DCMrights{Copyright {\copyright} #1: #2}}
⟨/package⟩
⟨*ltxml⟩
DefMacro('\DCMcopyrightnotice{}{}','\DCMrights{Copyright {\copyright} #1: #2}');
⟨/ltxml⟩
```

\cclicense

```
⟨*package⟩
\def\cclicense#1{\def\attribution{\def\dcm@by{yes}}
```

```
  \def\noncommercial{\def\dcm@nc{yes}}
  \def\sharealike{\defdcm@sharealike{yes}}
  \def\noderivativeworks{\def\dcm@derivatives{no}}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\cclicense{}',"<cc:license>#1</cc:license>");
DefConstructor('\attribution',"<cc:attribution/>");
DefConstructor('\noncommercial',"<cc:noncommercial/>");
DefConstructor('\sharealike',"<cc:sharealike/>");
DefConstructor('\noderivativeworks',"<cc:noderivativeworks>");
⟨/ltxml⟩
```

`\DCMabstract`

```
⟨*package⟩
\long\def\DCMabstract#1{\long\def\dcm@abstract{#1}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMabstract{}',"<dc:description>#1</dc:description>");
⟨/ltxml⟩
```

4.3 DCM Block Styles

We now define various commonly used styles.

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

```
⟨*package⟩
\def\dcm@titlepage@block{\begin{titlepage}
    \null\vfil\vskip 60\p@
    \begin{center}
      \ifx\dcm@title\@empty
      \PackageWarning{dcm}{No title specified}{\LARGE Add title here\par}
      \else\LARGE \dcm@title \par\fi
      \ifx\dcm@subtitle\@empty
      \vskip 3em\Large \dcm@subtitle \par\vskip 3em
      \else\large\lineskip .75em\WAauthorblock\dcm@creators\vskip 1.5em\fi
      \ifx\dcm@date\@empty
      \PackageWarning{dcm}{No date specified}{\large\today\par}
      \else{\large\dcm@date\par}\vskip 2em\fi
  \end{center}\vskip2em
  \ifx\dcm@abstract\@empty
  \PackageWarning{dcm}{No Abstract specified}\else
  \begin{quote}\textbf\dcm@abstract@heading:\dcm@abstract\end{quote}\fi
  \vskip 2em\par\vfil\noindent
  {\small\noindent\dcm@rights\dcm@license}
  \end{titlepage}}
⟨/package⟩
```

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

```
⟨*package⟩
```

```
\def\dcm@maketitle@block{\def\@title{\dcm@title\ifx\dcm@subtitle\empty\else\newline\dcm@subtitl
\def\@author{\WAauthorblock\dcm@creators}%
\def\@date{\dcm@date}\maketitle}
⟨/package⟩
```

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

```
⟨*ltxml⟩
Tag('dc:description',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
Tag('dc:date',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
Tag('dc:creator',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
Tag('dc:contributor',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
Tag('dc:title',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
Tag('dc:subject',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
Tag('dc:publisher',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
Tag('dc:type',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
Tag('dc:identifier',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
Tag('dc:language',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
Tag('dc:rights',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
⟨/ltxml⟩
```

4.4 Dealing with ISO Dates

EdN:4

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

```
⟨ltxml⟩RawTeX('
⟨*package | ltxml⟩
\def\ISOtimestamp{\count1=\time\divide\count1 by 60 % hours
\count2=\count1\multiply\count2 by 60% minutes in \count1 hours
\count3=\time\advance\count3 by -\count2 % minutes
\the\year -\ifnum\month>9\else0\fi\the\month-\ifnum\day>9\else0\fi\the\day
T\ifnum\count1>9\else0\fi\the\count1:\ifnum\count3>9\else0\fi\the\count3:00Z}
⟨/package | ltxml⟩
⟨ltxml⟩');
```

4.5 Configuration

```
⟨*package⟩
\def\dcm@abstract@heading{Abstract}
\def\dcm@creators@heading{Author(s)}
\def\dcm@contributors@connector{with contributions from}
\def\dcm@chapter@heading{Chapter}
\def\dcm@section@heading{Section}
\def\dcm@subsection@heading{Subsection}
\def\dcm@subsubsection@heading{Subsubsection}
\def\dcm@paragraph@heading{Paragraph}
⟨/package⟩
```

[4] EdNote: 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 ⟨∗ltxml⟩
```
236 Tag('dc:title',afterOpen=>\&numberIt,afterClose=>\&locateIt);
```
237 ⟨/ltxml⟩

4.7 Finale

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

238 ⟨ltxml⟩`1;`