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

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

John Doe
DFKI Bremen
http://dfki.de/jdoe

July 20, 2010

Contents

[*] Version v0.3 (last revised 2010/06/25)

1 Introduction

Author(s): Michael Kohlhase

The dcm packge allows mark up Dublin Core Metadata [DUB03] 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]

EdNote(1)
EdNote(2)

2 The User Interface

Author(s): Michael Kohlhase John Doe

2.1 Persons

The dcm package recognizes that from a metadata perspective, persons are complex entities. In particular, specifying metadata is a tedious and repetitive task that leads to embarrassing errors. Therefore the dcm package takes a hint from bibTeX and allows to specify personal metadata in a database and use it by a database key. The \DCMperson macro allows to specify personal metadata[3] with

```
\DCMperson[id=jdoe,affiliation={DFKI Bremen},
           url=http://dfki.de/jdoe]{John Doe}
\DCMperson[id=miko,affiliation={Jacobs University, Bremen},
           url=http://kwarc.info/kohlhase]{Michael Kohlhase}
```

Example 1: A small database of Persons

\DCMperson

EdNote(3)

the following keys:

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

[2] EdNote: continue

[3] EdNote: This should be synchronized with the FOAF specification [BM07]

key	comment
id	identifier of this person
birthdate	birthdate
email	the primary e-mail address
url	the URI of the primary home page
affiliation	the primary professional affiliation
personaltitle	the personal title e.g. `King`
academictitle	the academic title e.g. `Prof. Dr.`
department	the department specified in the work address
workaddress	the work address
privaddress	the private address
worktel	work telephone number
privtel	private telephone number
workfax	work fax number
privfax	private fax number
worktelfax	if the phone and fax share a prefix, give this as well
privtelfax	dito

In Figure 1 we have specified (minimal) metadata for the authors of the `dcm` package. The metadata can be accessed by specifying the identifiers (given by the `id` key) in the `dcm` macros defined below, see for instance the `\DCMcreators` macro in Figure 2, which leads to the title block of this note.

Like in bibTeX [Pat], it is a good idea to collect the metadata in a separate file that is input in the document. In practice it may be possible to generate these files from conventional address databases.

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 2 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 `id`s of persons specified via `\DCMperson` 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

`\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 2: The DC Metadata block for the `dcm` package documentation

2.3 DCM Metadata Block Styles

Author(s): John Doe

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.

`maketitle`
`titlepage`

`\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 Subsection 2.2 there is a token register `\dcm@`⟨*md*⟩ that contains the value specified in the key.

2.4 Augmented Sectioning Macros

`\DCMchapter`
`\DCMsection`
`\DCMsubsection`
`\DCMsubsubsection`

The `dcm` package supplies sectioning commands customized for metadata handling: The `\DCMchatper` `\DCMsection` `\DCMsubsection`, and `\DCMsubsubsection` macros behave like their regular LaTeX counterpart, except that the optional argument, which is used to specify a short title there, is now a KeyVal argument. The `short` key takes over the role of specifying a short title in `\DCM`⟨*sect*⟩. The `id` key allows to specify an identifier, which can be used for LaTeX-style referencing (use `\ref{`⟨*sect*⟩`.`⟨*id*⟩`}`) or `sref`-style referencing [Koh] (use `\sref{`⟨*id*⟩`}`). Finally, the keys `creators` and `contributors` can be used to specify the authors (creators in Dublin Core speak [DUB03]) and contributors. For instance the following line was used to specify the heading of Section 2.

```
\DCMsection[id=user,creators={miko,jdoe}]{The User Interface}
```

Finally, the `style` key can be used to specify a style.

2.5 Configuration

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

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

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 The Implementation

Author(s): John Doe

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{sref}
⟨/package⟩
```

```
⟨*ltxml⟩
# -*- CPERL -*-
package LaTeXML::Package::Pool;
use strict;
use LaTeXML::Global;
use LaTeXML::Package;
RequirePackage('sref');
⟨/ltxml⟩
```

3.1 Persons

To implement the \DCMperson macro, we need to implement its keywords first and also the \dcm@person@clear@keys macro that clears them. [4]

EdNote(4)

```
⟨*package⟩
\omdaddkey{dcm@person}{id}
\omdaddkey{dcm@person}{birthdate}
\omdaddkey{dcm@person}{email}
\omdaddkey{dcm@person}{url}
\omdaddkey{dcm@person}{affiliation}
\omdaddkey{dcm@person}{personaltitle}
\omdaddkey{dcm@person}{academictitle}
\omdaddkey{dcm@person}{department}
\omdaddkey{dcm@person}{workaddress}
\omdaddkey{dcm@person}{privaddress}
\omdaddkey{dcm@person}{worktel}
\omdaddkey{dcm@person}{privtel}
\omdaddkey{dcm@person}{workfax}
\omdaddkey{dcm@person}{privfax}
\omdaddkey{dcm@person}{worktelfax}
\omdaddkey{dcm@person}{privtelfax}
⟨/package⟩
```

The next macro is an auxiliary one that puts the value into an appropriate token register. At the LaTeXML side we have a function ExportMetadata that does a similar job, fishing out the metadata keys from the keyval arguments and storing them in a safe place so they can be accessed later.

```
⟨*package⟩
\def\dcm@pers@def#1#2{\expandafter\xdef\csname dcm@person@\dcm@person@id @#1\endcsname{#2}}
\def\dcm@pers@ref#1#2{\csname dcm@person@#1@#2\endcsname}
\let\dcm@persons=\relax
⟨/package⟩
⟨*ltxml⟩
 sub getKeyValue_noDelim {
 my ($keyval,$key)=@_;
 my $valuelist = ToString($keyval->getValue($key));
 $valuelist =~ s/^{(.*)}$/$1/g if $valuelist;
```

[4] EdNote: need some DCM stuff here

```
 return $valuelist;
}
sub ExportMetadata {
  my $keys = shift;
  my($id, $email,$affill,$address,$url,$name)=$keys
    && map(getKeyValue_noDelim($keys,$_),qw(id email affiliation address url name));
  if ($id) {
   AssignValue('DCM_'.$id.'_email',$email,'global') if $email;
   AssignValue('DCM_'.$id.'_affiliation',$affill,'global') if $affill;
   AssignValue('DCM_'.$id.'_address',$email,'global') if $email;
   AssignValue('DCM_'.$id.'_url',$url,'global') if $url;
   AssignValue('DCM_'.$id.'_name',$name,'global') if $name;
  } else {print STDERR "Warning: key 'id' undefined in \\DCMperson\n"};
  return;}
⟨/ltxml⟩
```

With this we can define the `\DCMperson` macro, it just clears the keys, sets them again, and stores them in token registers. If course only if a `id` attribute is given, else we raise an error.

DCMPerson

```
⟨*package⟩
\newcommand{\DCMperson}[2][]{\omdsetkeys{dcm@person}{#1}
\ifx\dcm@person@id\@empty\@latex@warning{key 'id' undefined in DCMperson}\else
\dcm@pers@def{name}{#2}
\dcm@pers@def{email}{\dcm@person@email}
\dcm@pers@def{birthdate}{\dcm@person@birthdate}
\dcm@pers@def{url}{\dcm@person@url}
\dcm@pers@def{affiliation}{\dcm@person@affiliation}
\dcm@pers@def{workaddress}{\dcm@person@workaddress}
\dcm@pers@def{privaddress}{\dcm@person@privaddress}
\dcm@pers@def{personaltitle}{\dcm@person@personaltitle}
\dcm@pers@def{academictitle}{\dcm@person@academictitle}
\dcm@pers@def{department}{\dcm@person@department}
\dcm@pers@def{workaddress}{\dcm@person@workaddress}
\dcm@pers@def{privaddress}{\dcm@person@privaddress}
\dcm@pers@def{worktel}{\dcm@person@worktel}
\dcm@pers@def{privtel}{\dcm@person@privtel}
\dcm@pers@def{workfax}{\dcm@person@workfax}
\dcm@pers@def{privfax}{\dcm@person@privfax}
\dcm@pers@def{worktelfax}{\dcm@person@worktelfax}
\dcm@pers@def{privtelfax}{\dcm@person@privtelfax}
\@ifundefined{dcm@persons}{\xdef\dcm@persons{\dcm@person@id}}{\xdef\dcm@persons{\dcm@persons,\d
\fi}
⟨/package⟩
⟨*ltxml⟩
DefKeyVal('dcm@person','id','Semiverbatim');
DefKeyVal('dcm@person','birthdate','Semiverbatim');
DefKeyVal('dcm@person','email','Semiverbatim');
DefKeyVal('dcm@person','url','Semiverbatim');
DefKeyVal('dcm@person','affiliation','Semiverbatim');
DefKeyVal('dcm@person','personaltitle','Semiverbatim');
DefKeyVal('dcm@person','academictitle','Semiverbatim');
DefKeyVal('dcm@person','department','Semiverbatim');
DefKeyVal('dcm@person','workaddress','Semiverbatim');
DefKeyVal('dcm@person','privaddress','Semiverbatim');
DefKeyVal('dcm@person','worktel','Semiverbatim');
DefKeyVal('dcm@person','privtel','Semiverbatim');
DefKeyVal('dcm@person','workfax','Semiverbatim');
DefKeyVal('dcm@person','privfax','Semiverbatim');
DefKeyVal('dcm@person','worktelfax','Semiverbatim');
DefKeyVal('dcm@person','privtelfax','Semiverbatim');

DefConstructor('\DCMperson OptionalKeyVals:dcm@person {}','',
beforeDigest=> sub { AssignValue(inPreamble=>0); },
afterDigest=>sub {
  my ($stomach,$whatsit)=@_;
  my $keys=$whatsit->getArg(1);
  my $name=ToString($whatsit->getArg(2));
  $keys->setValue('name',$name);
  ExportMetadata($keys);
  return;
});#$
⟨/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⟩
```

3.2 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 @
⟨/package⟩
⟨*ltxml⟩
DefEnvironment('{DCmetadata}[]', "<omdoc:metadata>#body</omdoc:metadata>");
⟨/ltxml⟩
```

EdNote(5)

Now some auxiliary macros to make author blocks. [5]

```
⟨*package⟩
\def\dc@mtabline#1#2{\xdef\tab@line{}%
\@for\@p:={#1}\do{\xdef\tab@line{\tab@line&\dcm@pers@ref\@p{#2}}}
\tab@line}
\def\dcm@atabline#1#2#3{\xdef\tab@line{}%
\@for\@p:={#1}\do{\xdef\tab@line{\tab@line&#2: \dcm@pers@ref\@p{#3}}}
\tab@line}
\def\dcm@bitabline#1#2#3#4{\xdef\tab@line{}%
\@for\@p:={#1}\do{\xdef\tab@line{\tab@line&\dcm@pers@ref\@p{#2} #3 \dcm@pers@ref\@p{#4}}}
\tab@line}
⟨/package⟩
```

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{%
\ifx\csname dcm@person@\@I @id\endcsname\@empty% undefined \dcm@person@id
\PackageError{dcm}{reference to undefined DCMperson \@I}%
{you must define a person with id=\@I\MessageBreak%
via the macro \protect\DCMperson, before you can use it in \protect\DCMcreators}\fi}%
\def\dcm@creators{#1}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMcreators{}',sub{
  my ($document,$args,%properties) = @_;
  my $keyval = LaTeXML::KeyVals->new('dcm@person',T_BEGIN,T_END,('creators'=>$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{%
\ifx\csname dcm@person@\@I @id\endcsname\@empty% undefined \dcm@person@id
```

[5] EdNote: use and document them!

```
\PackageError{dcm}{reference to undefined DCMperson \@I}%
{you must define a person with id=\@I\MessageBreak%
via the macro \protect\DCMperson, before you can use it in \protect\DCMcontributors}}%
\else% all \dcm@person@id in the list are defined
\def\dcm@contributors{#1}\fi}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMcontributors{}',sub{
  my ($document,$args,%properties) = @_;
  my $keyval = LaTeXML::KeyVals->new('dcm@person',T_BEGIN,T_END,('contributors'=>$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⟩
```

EdNote(6)

\DCMdate the \DCMdate uses \today as a default[6]

```
⟨*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⟩
```

\DCMlanguage

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

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

\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⟩
```

3.3 DCM Block Styles

Author(s): John Doe

We now define various commonly used styles.

\dcm@authorblock This internal macro builds an author block from a list of \DCMperson labels in \dcm@creators.

```
⟨*package⟩
\def\dcm@authorblock{\newcounter{authors}\stepcounter{authors}
{\let\tabularnewline\relax
  \@for\@I:=\dcm@creators\do{\stepcounter{authors}}
  \def\@authors{}\def\@affs{}\def\@urls{}
  \@for\@I:=\dcm@creators\do
    {\xdef\@authors{\@authors&\csname dcm@person@\@I @name\endcsname}
     \xdef\@affs{\@affs&\csname dcm@person@\@I @affiliation\endcsname}
     \xdef\@urls{\@urls&\csname dcm@person@\@I @url\endcsname}}
  \message{authors: \@authors}}
  \begin{tabular}[t]{l*{\theauthors}{c}}\@authors\\\@affs\\\@urls\end{tabular}}
⟨/package⟩
```

\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\dcm@authorblock\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{\dcm@authorblock}\def\@date{\dcm@date}\maketitle}
⟨/package⟩
```

3.4 Augmented Sectioning Macros

\dcm@sect@clear@keys We first define the sectioning keys with the infrastructure from the `omd` package [Koh10]

```
⟨*package⟩
\srefaddidkey{dcm@sect}
\omdaddkey{dcm@sect}{short}
\omdaddkey{dcm@sect}{creators}
\omdaddkey{dcm@sect}{contributors}
⟨/package⟩
```

\dcm@section The next step is to define an auxiliary macro that does all the work. `\dcm@section{`⟨*type*⟩`}{`⟨*title*⟩`}{`⟨*keys*⟩`}` delegates the presentation of the ⟨*title*⟩ and the metadata in ⟨*keys*⟩ to the `\dcm@section@*@style` macro specified ⟨*style*⟩.

```
⟨*package⟩
\def\dcm@section#1#2#3{\def\dcm@sect@type{#1}\omdsetkeys{dcm@sect}{#3}\sref@target%
\ifx\dcm@sect@short\@empty\csname #1\endcsname{#2}%
\else\csname #1\endcsname[\dcm@sect@short]{#2}\fi~%
\@ifundefined{sect@style} %
{\dcm@section@default@style{#1}}%
{\csname dcm@section@\sect@style @style\endcsname{#1}}}
⟨/package⟩
```

\dcm@section@default@style This is the default style for sectional metadata

```
⟨*package⟩
\def\dcm@section@default@style#1{%
\ifx\dcm@sect@creators\@empty\else%
\textbf{\dcm@creators@heading}:%
\@for\@I:=\dcm@sect@creators\do{~\csname dcm@person@\@I @name\endcsname}\fi%
\ifx\dcm@sect@contributors\@empty\else% there are contributors
\dcm@contributors@connector%
\@for\@I:=\dcm@sect@contributors\do{~\csname dcm@person@\@I @name\endcsname}\\\fi}
⟨/package⟩
```

Armed with these, the rest is very simple

\DCMchapter

```
⟨*package⟩
\newcommand{\DCMchapter}[2][]%
{\dcm@section{chapter}{#2}{#1}\sref@label@id{\dcm@chapter@heading\ \thechapter}}
⟨/package⟩
```

For the LaTeXML bindings we have to be a bit more creative. We only open the `omgroup` element (and declare it to autoclose)[7] EdNote(7)

```
⟨*ltxml⟩
Tag('omdoc:omgroup', autoClose=>1);
DefConstructor('\DCMchapter OptionalKeyVals:omgroup {}',sub {
  my ($document,$keyvals,$title,%properties) = @_;
  my $id=$keyvals->getValue('id')||"";
  $document->openElement('omdoc:chapter',('xml:id'=>$id));
  $document->openElement('omdoc:metadata');
  $document->insertElement('dc:title',$title);
  FishOutMetadata($document,$keyvals);
  return;});
⟨/ltxml⟩
```

EdNote(8) [8]

\DCMsection

```
⟨*package⟩
\newcommand{\DCMsection}[2][]%
{\dcm@section{section}{#2}{#1}\sref@label@id{\dcm@section@heading\ \thesection}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMsection OptionalKeyVals:omgroup {}',sub{
  my ($document,$keyvals,$title,%properties) = @_;
  my $id=$keyvals->getValue('id')||"";
  $document->openElement('omdoc:section',('xml:id'=>$id));
  $document->openElement('omdoc:metadata');
  $document->insertElement('dc:title',$title);
  FishOutMetadata($document,$keyvals);
  return;});
⟨/ltxml⟩
```

\DCMsubsection

```
⟨*package⟩
\newcommand{\DCMsubsection}[2][]%
{\dcm@section{subsection}{#2}{#1}\sref@label@id{\dcm@subsection@heading\ \thesubsection}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMsubsection OptionalKeyVals:omgroup {}',sub{
  my ($document,$keyvals,$title,%properties) = @_;
```

[7] EdNote: @DEYAN, we need to document more here; and we should also close the metadata element just to be more tidy.

[8] EdNote: @DEYAN: I guess since they all do the same as DCMchapter, we may actually get away with Let(DCMsection,DCMchapter); here.

```
  my $id=$keyvals->getValue('id')||"";
  $document->openElement('omdoc:subsection',('xml:id'=>$id));
  $document->openElement('omdoc:metadata');
  $document->insertElement('dc:title',$title);
  FishOutMetadata($document,$keyvals);
  return;});
⟨/ltxml⟩
```

\DCMsubsubsection

```
⟨*package⟩
\newcommand{\DCMsubsubsection}[2][]%
{\dcm@section{subsubsection}{#2}{#1}\sref@label@id{\dcm@subsubsection@heading\ \thesubsubsectio
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMsubsubsection OptionalKeyVals:omgroup {}',sub{
  my ($document,$keyvals,$title,%properties) = @_;
  my $id=$keyvals->getValue('id')||"";
  $document->openElement('omdoc:subsubsection',('xml:id'=>$id));
  $document->openElement('omdoc:metadata');
  $document->insertElement('dc:title',$title);
  FishOutMetadata($document,$keyvals);
  return;});
⟨/ltxml⟩
```

\DCMparagraph

```
⟨*package⟩
\newcommand{\DCMparagraph}[2][]%
{\dcm@section{paragraph}{#2}{#1}\sref@label@id{this \dcm@paragraph@heading}}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\DCMparagraph OptionalKeyVals:omgroup {}',sub{
  my ($document,$keyvals,$title,%properties) = @_;
  my $id=$keyvals->getValue('id')||"";
  $document->openElement('omdoc:paragraph',('xml:id'=>$id));
  $document->openElement('omdoc:metadata');
  $document->insertElement('dc:title',$title);
  FishOutMetadata($document,$keyvals);
  return;});
⟨/ltxml⟩
```

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

```
⟨*ltxml⟩
Tag('omdoc:chapter',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
Tag('omdoc:section',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
Tag('omdoc:subsection',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
Tag('omdoc:subsubsection',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
Tag('omdoc:paragraph',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
Tag('omdoc:subparagraph',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
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⟩
```

3.5 Dealing with ISO Dates

EdNote(9)

The first step is to build a macro for making ISO dates.[9]

```
⟨*package⟩
\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⟩
RawTeX('\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 -\the\month-\the\day T\the\count1:\the\count3:00Z}');
⟨/ltxml⟩
```

3.6 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}
⟨/package⟩
```

3.7 Providing IDs for OMDoc Elements

[9] EdNote: make better ltxml

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

```
⟨*ltxml⟩
Tag('dc:title',afterOpen=>\&numberIt,afterClose=>\&locateIt);
⟨/ltxml⟩
```

3.8 Finale

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

```
⟨ltxml⟩1;
```

References

[BM07] Dan Brickley and Libby Miller. *FOAF Vocabulary Specification 0.91.* Tech. rep. ILRT Bristol, Nov. 2007. URL: `http://xmlns.com/foaf/spec/20071002.html`.

[DUB03] The DCMI Usage Board. *DCMI Metadata Terms.* DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: `http://dublincore.org/documents/dcmi-terms/`.

[Koh] Tech. rep. Comprehensive TeX Archive Network (CTAN), URL: `http://www.ctan.org/tex-archive/macros/latex/contrib/stex/sref/sref.pdf`.

[Koh10] Michael Kohlhase. *`omd.sty`: A generic framework for extensible Metadata in LaTeX.* Self-documenting LaTeX package. Comprehensive TeX Archive Network (CTAN), 2010. URL: `http://www.ctan.org/tex-archive/macros/latex/contrib/stex/omd/omd.pdf`.

[Pat] Oren Patashnik. *bibTeXing.* URL: `http://www.ctan.org/get/biblio/bibtex/contrib/doc/btxdoc.pdf` (visited on 12/14/2009).