

CNX \LaTeX : A \LaTeX -based Syntax for Connexions Modules*

Michael Kohlhasse
Jacobs University, Bremen
<http://kwarc.info/kohlhasse>

July 20, 2010

Abstract

We present CNX \LaTeX , a collection of \LaTeX macros that allow to write CONNEXIONS modules without leaving the \LaTeX workflow. Modules are authored in CNX \LaTeX using only a text editor, transformed to PDF and proofread as usual. In particular, the \LaTeX workflow is independent of having access to the CONNEXIONS system, which makes CNX \LaTeX attractive for the initial version of single-author modules.

For publication, CNX \LaTeX modules are transformed to CNXML via the \LaTeX XML translator and can be uploaded to the CONNEXIONS system.

*Version ? (last revised ?)

Contents

1	Introduction	3
2	The User Interface	3
2.1	Document Structure	3
2.2	Mathematics	4
2.3	Statements	4
2.4	Connexions: Links and Cross-References	5
2.5	Metadata	6
2.6	Exercises	7
2.7	Graphics, etc.	7
3	The Implementation	8
3.1	Document Structure	9
3.2	Mathematics	12
3.3	Rich Text	12
3.4	Statements	15
3.5	Connexions	18
3.6	Metadata	19

1 Introduction

EdNote(1)

The Connexions project is a¹

The CNXML format — in particular the embedded content MATHML — is hard to write by hand, so we provide a set of environments that allow to embed the CNXML document model into L^AT_EX.

2 The User Interface

EdNote(2)

This document is not a manual for the Connexions XML encoding, or a practical guide how to write Connexions modules. We only document the L^AT_EX bindings for CNXML and will presuppose experience with the format or familiarity with². Note that formatting CNXL^AT_EX documents with the L^AT_EX formatter does little to enforce the restrictions imposed by the CNXML document model. You will need to run the L^AT_EXML converter for that (it includes DTD validation) and any CNX-specific quality assurance tools after that.³

EdNote(3)

The CNXL^AT_EX class makes heavy use of the KeyVal package, which is part of your L^AT_EX distribution. This allows to add optional information to L^AT_EX macros in the form of key-value pairs: A macro `\foo` that takes a KeyVal argument and a regular one, so a call might look like `\foo{bar}` (no KeyVal information given) or `\foo[key1=val1,...,keyn=valn]{bar}`, where `key1,...,keyn` are predefined keywords and values are L^AT_EX token sequences that do not contain comma characters (though they may contain blank characters). If a value needs to contain commas, then it must be enclosed in curly braces, as in `\foo[args={a,comma,separated,list}]`. Note that the order the key/value pairs appear in a KeyVal Argument is immaterial.

2.1 Document Structure

```
\documentclass{cnx}
\begin{document}
  \begin{cnxmodule}[name=Hello World,id=m4711]
    \begin{ccontent}
      \begin{cpara}[id=p01] Hello World\end{cpara}
    \end{ccontent}
  \end{cnxmodule}
\end{document}
```

Example 1: A Minimal CNXL^AT_EX Document

The first set of CNXL^AT_EX environments concern the top-level structure of the modules. The minimal Connexions document in L^AT_EX can be seen in Figure 1:

¹EDNOTE: continue; copy from somewhere...

²EDNOTE: cite the relevant stuff here

³EDNOTE: talk about Content MATHML and cmathml.sty somewhere

<code>cnxmodule</code>	we still need the \LaTeX document environment, then the <code>cnxmodule</code> environment contains the module-specific information as a KeyVal argument with the two keys: <code>id</code> for the module identifier supplied by the CONNEXIONS system) and <code>name</code> for the title of the module.
<code>ccontent</code>	The <code>content</code> environment delineates the module content from the metadata (see Section 2.5). It is needed to make the conversion to CNXML simpler.
<code>c*section</code>	CNXML knows three levels of sectioning, so the $\text{CNX}\LaTeX$ class supplies three as well: <code>csection</code> , <code>csubsection</code> and <code>csubsubsection</code> . In contrast to regular \LaTeX , these are environments to keep the tight connection between the formats. These environments take an optional KeyVal argument with key <code>id</code> for the identifier and a regular argument for the title of the section (to be transformed into the CNXML <code>name</code> element).
<code>cpara</code> , <code>cnote</code>	The lowest levels of the document structure are given by paragraphs and notes. The <code>cpara</code> and <code>cnote</code> environment take a KeyVal argument with the <code>id</code> key for identification, the latter also allows a <code>type</code> key for the note type (an unspecified string ⁴).

EdNote(4)

2.2 Mathematics

Mathematical formulae are integrated into text via the \LaTeX math mode, i.e. wrapped in `$` characters or between `\(` and `\)` for inline mathematics and wrapped in `$$` or between `\[` and `\]` for display-style math. Note that CNXML expects Content MATHML as the representation format for mathematical formulae, while run-of-the-mill \LaTeX only specifies the presentation (i.e. the two-dimensional layout of formulae). The \LaTeX ML converter can usually figure out some of the content MATHML from regular \LaTeX , in other cases, the author has to specify it e.g. using the infrastructure supplied by the `cmathml` package.

<code>cequation</code>	For numbered equations, CNXML supplies the <code>equation</code> element, for which $\text{CNX}\LaTeX$ provides the <code>cequation</code> environment. This environment takes a KeyVal argument with the <code>id</code> key for the (required) identifier.
------------------------	--

2.3 Statements

CNXML provides special elements that make various types of claims; we collectively call them statements.

<code>cexample</code>	The <code>cexample</code> environment and <code>definition</code> elements take a KeyVal argument with key <code>id</code> for identification.
-----------------------	--

<code>crule</code> , <code>statement</code> , <code>proof</code>	In CNXML, the <code>rule</code> element is used to represent a general assertion about the state of the world. The $\text{CNX}\LaTeX$ <code>rule</code> ⁵ environment is its $\text{CNX}\LaTeX$ counterpart. It takes a KeyVal attribute with the keys <code>id</code> for identification, <code>type</code> to specify the type of the assertion (e.g. “Theorem”, “Lemma” or “Conjecture”), and <code>name</code> , if the assertion has a title. The body of the <code>crule</code> environment contains the statement of assertion in the <code>statement</code> environment and (optionally)
--	---

EdNote(5)

⁴EDNOTE: what are good values?

⁵EDNOTE: we have called this “`crule`”, since “`rule`” is already used by TEX .

a proof in the `proof` environment. Both take a KeyVal argument with an `id` key for identification.

```
\begin{crule}[id=prop1,type=Proposition]
  \begin{statement}[id=prop1s]
    Sample statement
  \end{statement}
  \begin{proof}[id=prop1p]
    Your favourite proof
  \end{proof}
\end{crule}
```

Example 2: A Basic crule Example

definition, cmeaning A definition defines a new technical term or concept for later use. The `definition` environment takes a KeyVal argument with the keys `id` for identification and `term` for the concept (definiendum) defined in this form. The definition text is given in the `cmeaning` environment¹, which takes a KeyVal argument with key `id` for identification. After the `cmeaning` environment, a `definition` can contain arbitrarily many `cexamples`.

```
\begin{definition}{term=term-to-be-defined, id=termi-def]
  \begin{cmeaning}[id=termi-meaning]
    {\term{Term-to-be-defined}} is defined as: Sample meaning
  \end{cmeaning}
\end{definition}
```

Example 3: A Basic definition and cmeaning Example

2.4 Connexions: Links and Cross-References

As the name CONNEXIONS already suggests, links and cross-references are very important for CONNEXIONS modules. CNXML provides three kinds of them. Module links, hyperlinks, and concept references.

cnxn Module links are specified by the `\cnxn` macro, which takes a keyval argument with the keys `document`, `target`, and `strength`. The `document` key allows to specify the module identifier of the desired module in the repository, if it is empty, then the current module is intended. The `target` key allows to specify the document fragment. Its value is the respective identifier (given by its `id` attribute in CNXML or the `id` key of the corresponding environment in CNXI^ATeX). Finally, the `strength` key allows to specify the relevance of the link.

The regular argument of the `\cnxn` macro is used to supply the link text.

link Hyperlinks can be specified by the `\link` macro in CNXI^ATeX. It takes a

¹we have called this `cmeaning`, since `meaning` is already taken by TeX

KeyVal argument with the key `src` to specify the URL of the link. The regular argument of the `\link` macro is used to supply the link text.

`\term` The `\term` marco can be used to specify the⁶

2.5 Metadata

Metadata is mostly managed by the system in CONNEXIONS, so we often do not need to care about it. On the other hand, it influences the system, so if we have work on the module extensively before converting it to CNXML, it may be worth-wile specify some of the data in advance.

```
\begin{metadata}[version=2.19,
                  created=2000/07/21,revised=2004/08/17 22:07:27.213 GMT-5]
\begin{authorlist}
  \cnxauthor[id=miko,firstname=Michael,surname=Kohlhase,
             email=m.kohlhase@iu-bremen.de]
\end{authorlist}
\begin{keywordlist}\keyword{Hello}\end{keywordlist}
\begin{cnxabstract}
  A Minimal CNXLaTeX Document
\end{cnxabstract}
\end{metadata}
```

Example 4: Typical CNXLaTeX Metadata

`metadata` The `metadata` environment takes a KeyVal argument with the keys `version`, `created`, and `revised` with the obvious meanings. The latter keys take ISO 8601 norm representations for dates and times. Concretely, the format is `CCYY-MM-DDThh:mm:ss` where “CC” represents the century, “YY” the year, “MM” the month, and “DD” the day, preceded by an optional leading “-” sign to indicate a negative number. If the sign is omitted, “+” is assumed. The letter “T” is the date/time separator and “hh”, “mm”, “ss” represent hour, minutes, and seconds respectively.

`authorlist, maintainerlist` The lists of authors and maintainers can be specified in the `authorlist` and `maintainerlist` environments, which take no arguments.

`cnxauthor, maintainer` The entries on this lists are specified by the `\cnxauthor` and `\maintainer` macros. Which take a KeyVal argument specifying the individual. The `id` key is the identifier for the person, the `honorific`, `firstname`, `other`, `surname`, and `lineage` keys are used to specify the various name parts, and the `email` key is used to specify the e-mail address of the person.

`keywordlist, keyword` The keywords are specified with a list of `keyword` macros, which take the respective keyword in their only argument, inside a `keyword` environment. Neither take any KeyVal arguments.

`cnxabstract` The abstract of a CONNEXIONS module is considered to be part of the meta-

⁶EDNOTE: continue, pending Chuck’s investigation.

data. It is specified using the `cnxabstract` environment. It does not take any arguments.

2.6 Exercises

`cexercise`, `cproblem`,
`csolution` An exercise or problem in CONNEXIONS is specified by the `cexercise` environment, which takes an optional keyval argument with the keys `id` and `name`. It must contain a `cproblem` environment for the problem statement and a (possibly) empty set of `csolution` environments. Both of these take an optional keyval argument with the key `id`.

2.7 Graphics, etc.

EdNote(7) `cfigure` For graphics we will use the `cfigure`⁷ macro, which provides a non-floating environment for including graphics into CNXML files. `cfigure` takes three arguments first an optional CNXML keys, then the keys of the `graphicx` package in a regular argument (leave that empty if you don't have any) and finally a path. So

```
\cfigure[id=foo,type=image/jpeg,caption=The first F00]{width=7cm,height=2cm}{../images/f
```

EdNote(8) Would include a graphic from the file at the path `../images/foo`, equip this image with a caption, and tell L^AT_EXML that⁸ the original of the images has the MIME type `image/jpeg`.

⁷EDNOTE: probably better call it `cgraphics`

⁸EDNOTE: err, exactly what does it tell latexml?

3 The Implementation

The `cnx` package generates to files: the \LaTeX package (all the code between $\langle *package \rangle$ and $\langle /package \rangle$) and the \LaTeXML bindings (between $\langle *ltxml \rangle$ and $\langle /ltxml \rangle$). We keep the corresponding code fragments together, since the documentation applies to both of them and to prevent them from getting out of sync.

We first make sure that the `sref` [Koh] and `graphicx` packages are loaded.

```
1  $\langle *cls \rangle$ 
2  $\backslash$ RequirePackage{sref}
3  $\backslash$ RequirePackage{graphicx}
```

The next step is to declare (a few) class options that handle the paper size; this is useful for printing.

```
4  $\backslash$ DeclareOption{letterpaper}
5   { $\backslash$ setlength\paperheight {11in}%
6      $\backslash$ setlength\paperwidth {8.5in}}
7  $\backslash$ DeclareOption{a4paper}
8   { $\backslash$ setlength\paperheight {297mm}%
9      $\backslash$ setlength\paperwidth {210mm}}
10  $\backslash$ ExecuteOptions{letterpaper}
11  $\backslash$ ProcessOptions
```

Finally, we input all the usual size settings. There is no sense to use something else, and we initialize the page numbering counter and tell it to output the numbers in arabic numerals (otherwise label and reference do not work).

```
12  $\backslash$ input{size10.clo}
13  $\backslash$ pagenumbering{roman}
14  $\langle /cls \rangle$ 
```

Now comes the equivalent for \LaTeXML : this is something that we will have throughout this document. Every part of the $\text{\TeX}/\text{\LaTeX}$ implementation has a \LaTeXML equivalent. We keep them together to ensure that they do not get out of sync.

```
15  $\langle *ltxml \rangle$ 
16 # -*- CPERL -*-
17 package LaTeXML::Package::Pool;
18 use strict;
19 use LaTeXML::Package;
20 RequirePackage('omd');
```

We set up the necessary namespaces, the first one is the default one for CNXML

```
21 RegisterNamespace('cnx'=>"http://cnx.rice.edu/cnxml");
22 RegisterNamespace('md'=>"http://cnx.rice.edu/mdml/0.4");
23 RegisterNamespace('bib'=>"http://bibtexml.sf.net/");
24 RegisterNamespace('m'=>"http://www.w3.org/1998/Math/MathML");
```

For \LaTeXML we also have to set up the correct document type information. The first line gives the root element. The second gives the public identifier for the CNX DTD, then we have its URL, and finally the CNX namespace.

```
25 DocType("cnx:document",
```



```

26 "-//CNX//DTD CNXML 0.5 plus LaTeXML//EN",
27 "../dtd/cnxml+ltxml.dtd",
28 '#default'=>"http://cnx.rice.edu/cnxml",
29   'md'=>"http://cnx.rice.edu/mdml/0.4",
30   'bib'=>"http://bibtexml.sf.net/",
31   'm'=>"http://www.w3.org/1998/Math/MathML",
32   'ltx'=>"http://dmlf.nist.gov/LaTeXML");

```

And finally, we need to set up the counters for itemization, since we are defining a class file from scratch.⁹

```

33 NewCounter('@itemizei', 'document', idprefix=>'I');
34 NewCounter('@itemizeii', '@itemizei', idprefix=>'I');
35 NewCounter('@itemizeiii', '@itemizeii', idprefix=>'I');
36 NewCounter('@itemizeiv', '@itemizeiii', idprefix=>'I');
37 NewCounter('@itemizev', '@itemizeiv', idprefix=>'I');
38 NewCounter('@itemizevi', '@itemizev', idprefix=>'I');
39
40 NewCounter('enumi', '@itemizei', idprefix=>'i');
41 NewCounter('enumii', '@itemizeii', idprefix=>'i');
42 NewCounter('enumiii', '@itemizeiii', idprefix=>'i');
43 NewCounter('enumiv', '@itemizeiv', idprefix=>'i');
44 # A couple of more levels, since we use these for ID's!
45 NewCounter('enumv', '@itemizev', idprefix=>'i');
46 NewCounter('enumvi', '@itemizevi', idprefix=>'i');
47
48 DefMacro('\theenumi', '\arabic{enumi}');
49 DefMacro('\theenumii', '\alph{enumii}');
50 DefMacro('\theenumiii', '\roman{enumiii}');
51 DefMacro('\theenumiv', '\Alph{enumiv}');
52
53 NewCounter('equation', 'document', idprefix=>'E');
54 DefMacro('\theequation', '\arabic{equation}');
55 DefMacro('\textwidth', '16cm');

```

And another thing that is now needed:

```

56 Let('\thedocument@ID', '\@empty');
57 </ltxml>

```

3.1 Document Structure

Now, we start with the document structure markup. The `cnxmodule` environment does not add anything to the \LaTeX output, it's attributes only show up in the XML. There we have a slight complication: we have to put an `id` attribute on the `document` element in CNXML, but we cannot redefine the `document` environment in \LaTeX . Therefore we specify the information in the `cnxmodule` environment. This means however that we have to put in on the `document` element when we are already past this. The solution here is that when we parse the `cnxmodule`

⁹EDNOTE: this will have to change, when Bruce updates to the next version (0.6?)

environment, we store the value and put it on the `document` element when we leave the `document` environment (thanks for Ioan Sucan for the code).

cnxmodule

```

58 <*cls>
59 \omdaddkey{cnxmodule}{name}
60 \srefaddidkey{cnxmodule}{id}
61 \newenvironment{cnxmodule}[1][\omdsetkeys{cnxmodule}{#1}]{
62 </cls>
63 <*ltxml>
64 DefKeyVal('cnxmodule','name','Semiverbatim');
65 DefKeyVal('cnxmodule','id','Semiverbatim');
66 DefEnvironment('{document}','<cnx:document>#body</cnx:document>',
67     beforeDigest=> sub { AssignValue(inPreamble=>0); },
68     afterDigest=> sub { $_[0]->getGullet->flush; return; });
69 DefEnvironment('{cnxmodule} OptionalKeyVals:cnxmodule',
70     "<cnx:name>&KeyVal('#1','name')</cnx:name>\n#body\n",
71     afterDigestBegin => sub {
72 AssignValue('cnxmodule_id',
73     KeyVal($_[1]->getArg(1), 'id')->toString,
74     'global');
75     });#$
76 Tag('cnx:document', afterClose => sub {
77     $_[1]->setAttribute('id', LookupValue('cnxmodule_id'));
78     });
79 </ltxml>

```

ccontent The `ccontent` environment is only used for transformation. Its optional `id` attribute is not taken up in the \LaTeX bindings.

```

80 <*cls>
81 \newenvironment{ccontent}{}{}
82 </cls>
83 <*ltxml>
84 DefEnvironment('{ccontent}', "<cnx:content>#body</cnx:content>");
85 </ltxml>

```

c*section The sectioning environments employ the obvious nested set of counters.

```

86 <*cls>
87 \newcounter{section}
88 \srefaddidkey{sectioning}{id}
89 \newenvironment{csection}[2][\%
90 {\stepcounter{section}\strut\{1.5ex\}\noindent%
91 {\Large\bfseries\arabic{section}.~{#2}}\{1.5ex\}
92 \omdsetkeys{sectioning}{#1}}
93 {}
94 \newcounter{subsection}[section]
95 \newenvironment{csubsection}[2][\%
96 {\refstepcounter{subsection}\strut\{1ex\}\noindent%
97 {\large\bfseries\arabic{section}.\arabic{subsection}.~#2\{1ex\}}\%

```

```

98 \omdsetkeys{sectioning}{#1}}%
99 {}
100 \newcounter{subsubsection}[subsection]
101 \newenvironment{csubsubsection}[2] []
102 {\refstepcounter{subsubsection}\strut\[\.5ex]\noindent
103 {\bfseries\arabic{section}.\arabic{subsection}.\arabic{subsubsecction}~#2\[\.5ex}}%
104 \omdsetkeys{sectioning}{#1}}{}
105 \</cls>
106 \<*txml>
107 DefKeyVal('sectioning','id','Semiverbatim');
108 DefEnvironment('{csection}OptionalKeyVals:sectioning{','
109     "<cnx:section %&KeyVals(#1)>\n"
110     . "?#2(<cnx:name>#2</cnx:name>\n)()"
111     . "#body\n</cnx:section>\n");
112 DefEnvironment('{csubsubsection}OptionalKeyVals:sectioning{','
113     "<cnx:section %&KeyVals(#1)>\n"
114     . "?#2(<cnx:name>#2</cnx:name>\n)()"
115     . "#body\n</cnx:section>\n");
116 DefEnvironment('{csubsubsection}OptionalKeyVals:sectioning{','
117     "<cnx:section %&KeyVals(#1)>\n"
118     . "?#2(<cnx:name>#2</cnx:name>\n)()"
119     . "#body\n</cnx:section>\n");
120 \</txml>

```

cpara For the `<cnx:para>` element we have to do some work, since we want them to be numbered. This handling is adapted from Bruce Miller's `LaTeX.ltxml` numbered.

```

121 \<cls>
122 \srefaddidkey{para}{id}
123 \newenvironment{cpara}[1] [] {\omdsetkeys{para}{#1}}{\par}
124 \</cls>
125 \<*txml>
126 DefKeyVal('para','id','Semiverbatim');
127 DefEnvironment('{cpara}OptionalKeyVals:para','<cnx:para %&KeyVals(#1)>#body</cnx:para>');
128 sub number_para {
129     my($document,$node,$whatsit)=@_;
130     # Get prefix from first parent with an id.
131     my(@parents)=$document->findnodes('ancestor::*[@id]', $node); # find 1st id'd parent.
132     my $prefix= (@parents ? $parents[$#parents]->getAttribute('id')." " : '');
133     # Get the previous number within parent; Worried about intervening elements around para's, bu
134     my(@siblings)=$document->findnodes("preceding-sibling::cnx:para", $node);
135     my $n=1;
136     $n = $1+1 if (@siblings && $siblings[$#siblings]->getAttribute('id')=~/(\\d+)$/);
137     $node->setAttribute(id=>$prefix."p$n"); }
138 Tag('cnx:para', afterOpen=>\&number_para);
139 DefConstructor('\par', sub { $_[0]->maybeCloseElement('cnx:para'); }, alias=>"\par\n");
140 Tag('cnx:para', autoClose=>1, autoOpen=>1);
141 \</txml>

```

cnote

```

142 <*cls>
143 \srefaddidkey{note}
144 \omdaddkey{note}{type}
145 \newenvironment{cnote}[1][]{%
146 {\omdsetkeys{note}{#1}\par\noindent\strut\hfill\begin{minipage}{10cm}{\bfseries\note@type:~}%
147 {\end{minipage}\hfill\strut\par}
148 </cls>
149 <*ltxml>
150 DefKeyVal('note','id','Semiverbatim');
151 DefKeyVal('note','type','Semiverbatim');
152 DefEnvironment('{cnote}OptionalKeyVals:note','<cnx:note %&KeyVals(#1)>#body</cnx:note>');
153 </ltxml>

```

3.2 Mathematics

cequation

```

154 <*cls>
155 \srefaddidkey{equation}{id}
156 \newenvironment{cequation}[1][]{%
157 {\omdsetkeys{equation}{#1}\begin{displaymath}}
158 {\end{displaymath}}
159 </cls>
160 <*ltxml>
161 DefKeyVal('equation','id','Semiverbatim');
162 DefEnvironment('{cequation}OptionalKeyVals:equation',
163     "<cnx:equation %&KeyVals(#1)>"
164     . "<ltx:Math mode='display'>"
165     . "<ltx:XMath>#body</ltx:XMath>"
166     . "</ltx:Math></cnx:equation>",
167     mode=>'display_math');
168 </ltxml>

```

3.3 Rich Text

In this section, we redefine some of L^AT_EX commands that have their counterparts in CNXML.

quote

```

169 <*cls>
170 \srefaddidkey{cquote}
171 \omdaddkey{cquote}{type}
172 \omdaddkey{cquote}{src}
173 \newenvironment{cquote}[1][]{%
174 \omdsetkeys{cquote}{#1}\begin{center}\begin{minipage}{.8\textwidth}}{\end{minipage}\end{center}}
175 </cls>
176 <*ltxml>
177 DefKeyVal('cquote','id','Semiverbatim');
178 DefKeyVal('cquote','type','Semiverbatim');
179 DefKeyVal('cquote','src','Semiverbatim');

```

```

180 DefEnvironment('{cquote} OptionalKeyVals:cquote',
181                 "<cnx:quote %&KeyVals(#1)>#body</cnx:quote>");
182 </ltxml>

```

footnote

```

183 <*ltxml>
184 DefConstructor('\footnote[]{}', "<cnx:note type='foot'>#2</cnx:note>");
185 </ltxml>

```

emph

```

186 <*ltxml>
187 DefConstructor('\emph{}', "<cnx:emphasis>#1</cnx:emphasis>");
188 </ltxml>

```

displaymath, eqnarray We redefine the abbreviate display math environment and the eqnarray and eqnarray* environments to use the CNXML equation tags, everything else stays the same.

```

189 <*ltxml>
190 DefConstructor('\[',
191               "<cnx:equation id='#id'>"
192               . "<ltx:Math mode='display'>"
193               . "<ltx:XMATH>"
194               . "#body"
195               . "</ltx:XMATH>"
196               . "</ltx:Math>"
197               . "</cnx:equation>",
198               beforeDigest=> sub{ $_[0]->beginMode('display_math'); },
199               captureBody=>1,
200               properties=> sub { RefStepID('equation') });
201 DefConstructor('\]', "", beforeDigest=> sub{ $_[0]->endMode('display_math'); });
202 </ltxml>

```

displaymath We redefine the abbreviate display math environment to use the CNXML equation tags, everything else stays the same.¹⁰

EdNote(10)

```

203 <*ltxml>
204 DefConstructor('\[',
205               "<cnx:equation id='#id'>"
206               . "<ltx:Math mode='display'>"
207               . "<ltx:XMATH>"
208               . "#body"
209               . "</ltx:XMATH>"
210               . "</ltx:Math>"
211               . "</cnx:equation>",
212               beforeDigest=> sub{ $_[0]->beginMode('display_math'); },
213               captureBody=>1,

```

¹⁰EDNOTE: check LaTeX.ltxml frequently and try to keep in sync, it would be good, if the code in LaTeXML.ltxml could be modularized, so that the cnx/ltx namespace differences could be relegated to config options

```

214         properties=> sub { RefStepID('equation') });
215 DefConstructor('\[' , "",beforeDigest=> sub{ $_[0]->endMode('display_math'); });
216
217 DefMacro('\eqnarray', '\@@eqnarray\@start@alignment');
218 DefMacro('\endeqnarray', '\@finish@alignment\end@eqnarray');
219 DefMacro('\csname eqnarray*\endcsname', '\@@eqnarray*\@start@alignment');
220 DefMacro('\csname endeqnarray*\endcsname', '\@finish@alignment\end@eqnarray');
221 DefConstructor('\@@eqnarray OptionalMatch:* AlignmentBody:\end@eqnarray',
222     sub {
223     my($document,$star,$body,%props)=@_;
224     $document->openElement('cnx:equation',refnum=>$props{refnum},id=>$props{id});
225     $document->openElement('ltx:Math',mode=>'display');
226     $document->openElement('ltx:XMath');
227     constructAlignment($document,$body,attributes=>{name=>'eqnarray'});
228     $document->closeElement('ltx:XMath');
229     $document->closeElement('ltx:Math');
230     $document->closeElement('cnx:equation'); },
231     mode=>'display_math',
232     beforeDigest=>sub { alignmentBindings('rcl'); },
233     properties=> sub { ($_[1] ? RefStepID('equation') : RefStepCounter('equation')) },
234     afterDigest=>sub {
235     $_[1]->setProperty(body=>$_[1]->getArg(2));},# So we get TeX
236     reversion=>'\\begin{eqnarray#1}#2\\end{eqnarray#1}');
237 </ltxml>

```

displaymath We redefine the abbreviate display math environment to use the CNXML equation tags, everything else stays the same.¹¹

EdNote(11)

```

238 <*cls>
239 \newcommand{\litem}[2] [] {\item[#1]\label{#2}}
240 </cls>
241 <*ltxml>
242 Tag('cnx:item', autoClose=>1);
243 DefConstructor('\item[]', "<cnx:item>?#1(<cnx:name>#1</cnx:name>)" );
244 DefConstructor('\litem[] {}', "<cnx:item id=' #2'>?#1(<cnx:name>#1</cnx:name>)" );
245 DefConstructor('\itemize@item[]',
246     "<cnx:item id=' #id'>?#1(<cnx:name>#1</cnx:name>)",
247     properties=>sub{ RefStepItemCounter(); });
248 DefConstructor('\enumerate@item[]',
249     "<cnx:item id=' #id'>?#1(<cnx:name>#1</cnx:name>)",
250     properties=>sub{ RefStepItemCounter(); });
251 DefConstructor('\description@item[]',
252     "<cnx::item id=' #id'>?#1(<cnx:name>#1</cnx:name>)",
253     properties=>sub{ RefStepItemCounter(); });
254 AssignValue(itemlevel=>0);
255 DefEnvironment('{itemize}',
256     "<cnx:list id=' #id' type='itemize'>#body</cnx:list>",

```

¹¹EDNOTE: check LaTeX.ltxml frequently and try to keep in sync, it would be good, if the code in LaTeXML.ltxml could be modularized, so that the cnx/ltx namespace differences could be relegated to config options

```

257         properties=>sub { beginItemize('itemize'); });
258 DefEnvironment('{enumerate}',
259     "<cnx:list type='enumerate' id='#id'>#body</cnx:list>",
260     properties=>sub { beginItemize('enumerate'); });
261 DefEnvironment('{description}',
262     "<cnx:list type='description' id='#id'>#body</cnx:list>",
263     properties=>sub { beginItemize('description'); });
264 </ltxml>

```

The next set of commands and environments are largely presentational, so we just skip them.

```

265 <*ltxml>
266 DefEnvironment('{center}', '#body');
267 DefEnvironment('{minipage}{}', '#body');
268 DefEnvironment('{small}', '#body');
269 DefEnvironment('{footnotesize}', '#body');
270 DefEnvironment('{tiny}', '#body');
271 DefEnvironment('{scriptsize}', '#body');
272 </ltxml>
273 <*ltxml>
274 DefConstructor('\ref Semiverbatim', "<cnx:cnxn target='#1'>&LookupValue('LABEL@#1')</cnx:cnxn>");
275 </ltxml>

```

3.4 Statements

cexample

```

276 <cls>
277 \srefaddidkey{example}
278 \omdaddkey{example}{name}
279 \newenvironment{cexample}[1][\omdsetkeys{example}{#1}
280 {\ifx\example@name\empty\else\noindent\bfseries{\example@name}\fi}}
281 {}
282 </cls>
283 <*ltxml>
284 DefKeyVal('example', 'id', 'Semiverbatim');
285 DefEnvironment('{cexample}OptionalKeyVals:example',
286     "<cnx:example %&KeyVals(#1)>#body</cnx:example>");
287 </ltxml>

```

cexercise The `cexercise`, `cproblem` and `csolution` environments are very simple to set up for L^AT_EX. For the L^AT_EXML side, we simplify matters considerably for the moment by restricting the possibilities we have on the CNXML side: We assume that the content is just one `<cnx:para>` element for the `<cnx:problem>` and `<cnx:solution>` elements.¹²

```

288 <cls>
289 \newcounter{cexercise}

```

¹²EDNOTE: relax this when we have automated the generation of `cnx:para` elements

```

290 \srefaddidkey{cexercise}
291 \omdaddkey{cexercise}{name}
292 \newenvironment{cexercise}[1] [] {\omdsetkeys{cexercise}{#1}
293 {\ifx\cexercise@name\@empty\else\stepcounter{cexercise}\noindent\bfseries{\cexercise@name~\arab
294 {}
295 \srefaddidkey{cproblem}
296 \newenvironment{cproblem}[1] [] {\omdsetkeys{cproblem}{#1}}{}{}
297 \srefaddidkey{csolution}
298 \newenvironment{csolution}[1] [] {\omdsetkeys{csolution}{#1}}{\par\noindent\bfseries{Solution}}{}
299 \</cls>
300 \<*ltxml>
301 DefKeyVal('cexercise','id','Semiverbatim');
302 DefKeyVal('cexercise','name','Semiverbatim');
303 DefEnvironment('{cexercise}OptionalKeyVals:cexercise',
304     "<cnx:exercise ?&defined(&KeyVal{#1,'id'}) (id='&KeyVal{#1,'id'})>"
305     . "#body"
306     . "</cnx:exercise>");
307 DefKeyVal('cproblem','id','Semiverbatim');
308 DefKeyVal('cproblem','name','Semiverbatim');
309 DefEnvironment('{cproblem}OptionalKeyVals:cproblem',
310     "<cnx:problem ?&defined(&KeyVal{#1,'id'}) (id='&KeyVal{#1,'id'})>"
311     . "?&defined(&KeyVal{#1,'name'}) (<cnx:name>&KeyVal{#1,'name'}</cnx:name>\n) ()"
312     . "#body"
313     . "</cnx:problem>");
314 DefKeyVal('csolution','id','Semiverbatim');
315 DefKeyVal('csolution','name','Semiverbatim');
316 DefEnvironment('{csolution}OptionalKeyVals:cproblem',
317     "<cnx:solution ?&defined(&KeyVal{#1,'id'}) (id='&KeyVal{#1,'id'})>"
318     . "?&defined(&KeyVal{#1,'name'}) (<cnx:name>&KeyVal{#1,'name'}</cnx:name>\n) ()"
319     . "#body"
320     . "</cnx:solution>");
321 \</ltxml>

```

crule

```

322 \<cls>
323 \srefaddidkey{rule}
324 \omdaddkey{rule}{name}
325 \omdaddkey{rule}{type}
326 \newenvironment{crule}[1] [] {\omdsetkeys{rule}{#1}%
327 {\noindent\bfseries{\rule@type:}\ifx\rule@name\@empty\else~(\rule@name)\fi}}%
328 {}
329 \</cls>
330 \<*ltxml>
331 DefKeyVal('rule','id','Semiverbatim');
332 DefKeyVal('rule','name','Semiverbatim');
333 DefKeyVal('rule','type','Semiverbatim');
334 DefEnvironment('{crule}OptionalKeyVals:rule',
335     "<cnx:rule ?&defined(&KeyVal{#1,'id'}) (id='&KeyVal{#1,'id'}) type='&KeyVal{#1,"
336     . "?&defined(&KeyVal{#1,'name'}) (<cnx:name>&KeyVal{#1,'name'}</cnx:name>\n) ()"
337     . "\n#body\n"

```



```

338      . "</cnx:rule>\n");
339 </ltxml>

statement
340 <*cls>
341 \srefaddidkey{statement}
342 \newenvironment{statement}[1] [] {\omdsetkeys{statement}{#1}}{}
343 </cls>
344 <*ltxml>
345 DefKeyVal('statement','id','Semiverbatim');
346 DefEnvironment('{statement} OptionalKeyVals:statement','<cnx:statement %&KeyVals(#1)>#body</cnx:statement>');
347 </ltxml>

proof
348 <*cls>
349 \srefaddidkey{proof}
350 \newenvironment{proof}[1] [] {\omdsetkeys{proof}{#1}}{}
351 </cls>
352 <*ltxml>
353 DefKeyVal('proof','id','Semiverbatim');
354 DefEnvironment('{proof}OptionalKeyVals:proof','<cnx:proof %&KeyVals(#1)>#body</cnx:proof>');
355 </ltxml>

definition
356 <*cls>
357 \srefaddidkey{definition}
358 \omdaddkey{definition}{term}
359 \omdaddkey{definition}{seealso}
360 \newenvironment{definition}[1] [] {\omdsetkeys{definition}{#1}{\noindent\bfseries{Definition:}}}{\omdsetkeys{definition}{#1}{\noindent\bfseries{Definition:}}}
361 </cls>
362 <*ltxml>
363 DefKeyVal('definition','id','Semiverbatim');
364 DefKeyVal('definition','term','Semiverbatim');
365 DefKeyVal('definition','seealso','Semiverbatim');
366 DefEnvironment('{definition}OptionalKeyVals:definition',
367      "<cnx:definition ?&defined(&KeyVal(#1,'id'))(id='&KeyVal(#1,'id')')()>\n"
368      . "?&defined(&KeyVal(#1,'term'))(<cnx:term>&KeyVal(#1,'term')</cnx:term>\n)()"
369      . "\n#body\n"
370      . "?&defined(&KeyVal(#1,'seealso'))(<cnx:seealso><cnx:term>&KeyVal(#1,'term')</cnx:term></cnx:definition>\n");
371      . "</cnx:definition>\n");
372 </ltxml>

cmeaning
373 <*cls>
374 \srefaddidkey{meaning}
375 \newenvironment{cmeaning}[1] [] {\omdsetkeys{meaning}{#1}}{}
376 </cls>
377 <*ltxml>
378 DefKeyVal('meaning','id','Semiverbatim');

```

```

379 DefEnvironment('{cmeaning}OptionalKeyVals:meaning', '<cnx:meaning %&KeyVals(#1)>#body</cnx:meanin
380 </ltxml>

```

3.5 Conexxions

cnxn

```

381 <*cls>
382 \omdaddkey{cnxn}{document}
383 \omdaddkey{cnxn}{target}
384 \omdaddkey{cnxn}{strength}
385 \newcommand{\cnxn}[2][ ]{% keys, link text
386 {\omdsetkeys{cnxn}{#1}{\underline{#2}}\footnote{\ttfamily\@ifx\cnxn@document\empty\cnxn@docum
387 \newcommand\@makefntext[1]{\parindent 1em\noindent\hb@xt@1.8em{\hss\@makefnmark}#1}
388 </cls>
389 <*ltxml>
390 DefKeyVal('cnxn','document','Semiverbatim');
391 DefKeyVal('cnxn','target','Semiverbatim');
392 DefKeyVal('cnxn','strength','Semiverbatim');
393 DefConstructor('\cnxn OptionalKeyVals:cnxn {}','<cnx:cnxn %&KeyVals(#1)>#1</cnx:cnxn>');
394 </ltxml>

```

link

```

395 <*cls>
396 \omdaddkey{link}{src}
397 \newcommand{\link}[2][ ]{\omdsetkeys{link}{#1}\underline{#2}}
398 </cls>
399 <*ltxml>
400 DefKeyVal('link','src','Semiverbatim');
401 DefConstructor('\link OptionalKeyVals:link {}','<cnx:link %&KeyVals(#1)>#2</cnx:link>');
402 </ltxml>

```

cfigure The **cfigure** only gives us one of the possible instances of the `<figure>` element^{13,14} In \LaTeX , we just pipe the size information through to `includegraphics`, in \LaTeXML , we construct the CNXML structure¹⁵

```

403 <*cls>
404 \srefaddidkey{cfigure}
405 \omdaddkey{cfigure}{type}
406 \omdaddkey{cfigure}{caption}
407 \newcounter{figure}
408 \newcommand{\cfigure}[3][ ]{% cnx_keys, graphicx_keys, path
409 \begin{center}%
410 \includegraphics[#2]{#3}%
411 \omdsetkeys{cfigure}{#1}\sref@target%
412 \ifx\cfigure@caption\empty\else
413 \par\noindent Figure\refstepcounter{figure} {\arabic{figure}}: \cfigure@caption%

```

¹³EDNOTE: extend that

¹⁴EDNOTE: do more about required and optional keys in arguments.

¹⁵EDNOTE: what do we do with the `graphicx` information about size,... CSS?

```

414 \protected@edef\@currentlabel{\arabic{figure}}%
415 \sref@label{id{Figure \thefigure}\fi
416 \end{center}}
417 \</cls>
418 \<*txml>
419 DefKeyVal('cfigure','id','Semiverbatim');
420 DefKeyVal('cfigure','name','Semiverbatim');
421 DefKeyVal('cfigure','type','Semiverbatim');
422 DefKeyVal('cfigure','caption','Semiverbatim');
423 DefConstructor('\cfigure OptionalKeyVals:cfigure Semiverbatim Semiverbatim',
424               "<cnx:figure ?&defined(&KeyVal(#1,'id'))(id='&KeyVal(#1,'id')')()>"
425               . "?&defined(&KeyVal(#1,'name'))(<cnx:name>&KeyVal(#1,'name')</cnx:name>\n)()"
426               . "<cnx:media type='&KeyVal(#1,'type')' src='#3'/>"
427               . "?&defined(&KeyVal(#1,'caption'))(<cnx:caption>&KeyVal(#1,'caption')</cnx:caption>\n"
428               . "</cnx:figure>");
429 \</txml>

```

ccite

```

430 \<*cls>
431 \omdaddkey{ccite}{src}
432 \newcommand{\ccite}[2][\omdsetkeys{ccite}{#1}\emph{#2}]
433 \</cls>
434 \<*txml>
435 DefKeyVal('ccite','src','Semiverbatim');
436 DefConstructor('\ccite OptionalKeyVals:ccite {}','<cnx:cite %&KeyVals(#1)>#2</cnx:cite>');
437 \</txml>

```

term

```

438 \<*cls>
439 \newcommand{\term}[1]{\bfseries\underline{#1}}
440 \</cls>
441 \<*txml>
442 DefConstructor('\term[]{}','<cnx:term>#2</cnx:term>');
443 \</txml>

```

3.6 Metadata

metadata

```

444 \<*cls>
445 \omdaddkey{metadata}{version}
446 \omdaddkey{metadata}{created}
447 \omdaddkey{metadata}{revised}
448 \newsavebox{\metadatabox}
449 \newenvironment{metadata}[1][\%
450 {\noindent\hfill\begin{lrbox}{\metadatabox}
451 {\begin{minipage}{.8\textwidth}%
452 {\Large\bfseries CNX Module: \cnx@name\hfill\strut}\\[2ex]]%
453 {\end{minipage}\end{lrbox}\fbox{\usebox{\metadatabox}\hfill}
454 % \newenvironment{metadata}[1][\%

```

```

455 % {\noindent\strut\hfill\begin{lrbox}{\metadatabox}\begin{minipage}{10cm}%
456 % {\strut\hfill\Large\bfseries CNX Module: \cnx@name\hfill\strut}\\[2ex]}%
457 % {\end{minipage}\end{lrbox}\fbox{\usebox\metadatabox}\hfill\strut\\[3ex]}
458 \end{cls}
459 \end{*txml}
460 DefKeyVal('metadata','version','Semiverbatim');
461 DefKeyVal('metadata','created','Semiverbatim');
462 DefKeyVal('metadata','revised','Semiverbatim');
463 DefEnvironment('{metadata}OptionalKeyVals:metadata',
464     "<cnx:metadata>\n"
465     . "<md:version>&KeyVal('#1','version')</md:version>\n"
466     . "<md:created>&KeyVal('#1','created')</md:created>\n"
467     . "<md:revised>&KeyVal('#1','revised')</md:revised>\n"
468     . "#body\n"
469     . "</cnx:metadata>");
470 \end{*txml}

```

authorlist

```

471 \end{cls}
472 \newenvironment{authorlist}{\bfseries{Authors}:~}{\\[1ex]}
473 \end{cls}
474 \end{*txml}
475 DefEnvironment('{authorlist}',"<md:authorlist>#body</md:authorlist>");
476 \end{*txml}

```

maintainerlist

```

477 \end{cls}
478 \newenvironment{maintainerlist}{\bfseries{Maintainers}:~}{\\[1ex]}
479 \end{cls}
480 \end{*txml}
481 DefEnvironment('{maintainerlist}',"<md:maintainerlist>#body</md:maintainerlist>");
482 \end{*txml}

```

cnxauthor

```

483 \end{cls}
484 \srefaddidkey{auth}
485 \omdaddkey{auth}{honorific}
486 \omdaddkey{auth}{firstname}
487 \omdaddkey{auth}{other}
488 \omdaddkey{auth}{surname}
489 \omdaddkey{auth}{lineage}
490 \omdaddkey{auth}{email}
491 \newcommand{\cnxauthor}[1][\omdsetkeys{auth}{#1}\auth@first~\auth@sur,}
492 \end{cls}
493 \end{*txml}
494 DefKeyVal('auth','id','Semiverbatim');
495 DefKeyVal('auth','firstname','Semiverbatim');
496 DefKeyVal('auth','surname','Semiverbatim');
497 DefKeyVal('auth','email','Semiverbatim');

```

```

498 DefConstructor('cnxauthor OptionalKeyVals:auth',
499     "<md:author id='&KeyVal('#1','id')'>\n"
500     . "?&defined(&KeyVal(#1,'honorific'))(<md:honorific>&KeyVal('#1','honorific')</md:honorifi
501     . "?&defined(&KeyVal(#1,'firstname'))(<md:firstname>&KeyVal('#1','firstname')</md:firstnam
502     . "?&defined(&KeyVal(#1,'other'))(<md:other>&KeyVal('#1','other')</md:other>\n)()"
503     . "?&defined(&KeyVal(#1,'surname'))(<md:surname>&KeyVal('#1','surname')</md:surname>\n)()"
504     . "?&defined(&KeyVal(#1,'lineage'))(<md:lineage>&KeyVal('#1','lineage')</md:lineag
505     . "?&defined(&KeyVal(#1,'email'))(<md:email>&KeyVal('#1','email')</md:email>\n)()"
506     . "</md:author>\n");
507 </ltxml>

maintainer
508 <*cls>
509 \newcommand{\maintainer}[1] [] {\omdsetkeys{auth}{#1}\auth@first~\auth@sur,}
510 </cls>
511 <*ltxml>
512 DefConstructor('\maintainer OptionalKeyVals:auth',
513     "<md:maintainer id='&KeyVal('#1','id')'>\n"
514     . "?&defined(&KeyVal(#1,'honorific'))(<md:honorific>&KeyVal('#1','honorific')</md:honorifi
515     . "?&defined(&KeyVal(#1,'firstname'))(<md:firstname>&KeyVal('#1','firstname')</md:firstnam
516     . "?&defined(&KeyVal(#1,'other'))(<md:other>&KeyVal('#1','other')</md:other>\n)()"
517     . "?&defined(&KeyVal(#1,'surname'))(<md:surname>&KeyVal('#1','surname')</md:surname>\n)()"
518     . "?&defined(&KeyVal(#1,'lineage'))(<md:lineage>&KeyVal('#1','lineage')</md:lineag
519     . "?&defined(&KeyVal(#1,'email'))(<md:email>&KeyVal('#1','email')</md:email>\n)()"
520     . "</md:maintainer>\n");
521 </ltxml>

keywordlist
522 <*cls>
523 \newenvironment{keywordlist}{\bfseries{Keywords}:~}{\[/1ex]}
524 </cls>
525 <*ltxml>
526 DefEnvironment('{keywordlist}', "<md:keywordlist>\n#body\n</md:keywordlist>");
527 </ltxml>

keyword
528 <*cls>
529 \newcommand{\keyword}[1]{#1,}
530 </cls>
531 <*ltxml>
532 DefConstructor('\keyword {}', "<md:keyword>#1</md:keyword>");
533 </ltxml>

cnxabstract
534 <*cls>
535 \newenvironment{cnxabstract}%
536 {\par\noindent\strut\hfill\begin{minipage}{10cm}{\bfseries{Abstract}:~}}%
537 {\end{minipage}\hfill}
538 </cls>

```

```
539 <|xml>
540 DefEnvironment('{cnxabstract} OptionalKeyVals:cnxabstract',
541               "<md:abstract>\n#body\n</md:abstract>\n");
542 1;
543 </|xml>
```

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

authorlist,	maintainerlist=authorlist,	statement,	environments:;cpara,
	maintainerlist	proof (en-	cnote=cpara,
	(environment),	vironment),	cnote,
	6	4	4
c*section=c*section	definition,	cmeaning=definition,	environments:;crule,
(environment),	4	cmeaning (en-	statement,
ccontent=ccontent	vironment),	5	proof=crule,
(environment),	4		statement,
cequation=cequation	environments:;authorlist,		proof,
(environment),	4	maintainerlist=authorlist,	4
cexample=cexample		maintainerlist,	environments:;definition,
(environment),	4	6	cmeaning=definition,
cexercise,	cproblem,	environments:;c*section=c*section,	cmeaning,
csolution=cexercise,	4	4	5
cproblem,	environments:;ccontent=ccontent,		environments:;keywordlist,
csolution (en-	4		keyword=keywordlist,
vironment),	7	environments:;cequation=cequation,	keyword,
		6	6
cfigure=	\subitem **\cfigure+,	\usage{7}	environments:;metadata=metadata,
cnxabstract=cnxabstract	environments:;cexample=cexample,	keywordlist,	
(environment),	6	4	key-
cnxauthor,maintainer=	\subitem **\cnxauthor,maintainer+,	keyword{6} (en-	word=keywordlist,
cnxmodule=cnxmodule	cproblem, csolution=cexercise,	vironment),	6
(environment),	4		
cnxn=	\subitem **\cnxn+,	\usage{5}	link=
cpara,	cnote=cpara,	csolution,	7
cnote (en-	vironments:;cnxabstract=cnxabstract,	metadata	
vironment),	4	6	(environment),
crule,	statement,	environments:;cnxmodule=cnxmodule,	6
proof=crule,	4		term=
			\subitem **\term+, \usage{6}

References

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