LaTeXML *The Manual*

A LaTeX to XML Converter;
0.7.0

Bruce R. Miller

October 13, 2010

Contents

Chapter 1

Introduction

For many, LaTeX is the prefered format for document authoring, particularly those involving significant mathematical content and where quality typesetting is desired. On the other hand, content-oriented XML is an extremely useful representation for documents, allowing them to be used, and reused, for a variety of purposes, not least, presentation on the Web. Yet, the style and intent of LaTeX markup, as compared to XML markup, not to mention its programmability, presents difficulties in converting documents from the former format to the latter. Perhaps ironically, these difficulties can be particularly large for mathematical material, where there is a tendency for the markup to focus on appearance rather than meaning.

The choice of LaTeX for authoring, and XML for delivery were natural and uncontroversial choices for the Digital Library of Mathematical Functions[1]. Faced with the need to perform this conversion and the lack of suitable tools to perform it, the DLMF project proceeded to develop thier own tool, LaTeXML, for this purpose.

Design Goals The idealistic goals of LaTeXML are:

- Faithful emulation of TeX's behaviour.
- Easily extensible.
- Lossless; preserving both semantic and presentation cues.
- Uses abstract LaTeX-like, extensible, document type.
- Determine the semantics of mathematical content
  (*Good* Presentation MathML, eventually Content MathML and OpenMath).

As these goals are not entirely practical, or even somewhat contradictory, they are implicitly modified by *as much as possible*. Completely mimicing TeX's, and LaTeX's, behaviour would seem to require the sneakiest modifications to TeX, itself; redefining LaTeX's internals does not really guarantee compatibility. "Ease of use" is, of course,

[1] `http://dlmf.nist.gov`

in the eye of the beholder. More significantly, few documents are likely to have completely unambiguous mathematics markup; human understanding of both the topic and the surrounding text is needed to properly interpret any particular fragment. Thus, rather than pretend to provide a "turn-key" solution, we expect that document-specific declarations or tuning to be necessary to faithfully convert documents. Towards this end, we provide a variety of means to customize the processing and declare the author's intent. At the same time, especially for new documents, we encourage a more logical, content-oriented markup style, over a purely presentation-oriented style.

Overview of this Manual Chapter 2 describes the usage of LaTeXML, along with common use cases and techniques. Chapter **??** describes the system architecture in some detail. Strategies for customization and implementation of new packages is described in Chapter **??**. The special considerations for mathematics, including details of representation and how to improve the conversion, are covered in Chapter **??**. An overview of outstanding issues and planned future improvements are given in Chapter **??**.

Finally, the Appendices give detailed documentation the system components: Appendix **??** describes the command-line programs provided by the system; Appendices **??** and **??** describes the core and utility Perl modules comprising the system, while Appendix **??** describes the postprocessing modules; Appendix **??** describes the XML schema used by LaTeXML; finally, Appendix **??** gives an overview of the warning and error messages that LaTeXML may generate.

If all else fails, you can consult the source code, or the author.

Chapter 2

Using LaTeXML

The main commands provided by the LaTeXML system are

latexml for converting TeX and BibTeX sources to XML.

latexmlpost for various postprocessing tasks including conversion to HTML, processing images, conversion to MathML and so on.

The usage of these commands can be as simple as

```
latexml doc.tex | latexmpost --dest=doc.xhtml
```

to convert a single document into HTML, or as complicated as

```
latexml --dest=1.xml ch1
latexml --dest=2.xml ch2
            ...
latexml --dest=b.xml b
latexml --dest=B.xml B.bib
latexmlpost --prescan --db=my.db --bib=B.xml --dest=1.xhtml 1
latexmlpost --prescan --db=my.db --bib=B.xml --dest=2.xhtml 2
            ...
latexmlpost --prescan --db=my.db --bib=B.xml --dest=b.xhtml b
latexmlpost --noscan --db=my.db --bib=B.xml --dest=1.xhtml 1
latexmlpost --noscan --db=my.db --bib=B.xml --dest=2.xhtml 2
            ...
latexmlpost --noscan --db=my.db --bib=B.xml --dest=b.xhtml b
```

to convert a whole set of documents, including a bibliography, into a complete interconnected site.

How best to use the commands depends, of course, on what you are trying to achieve. In the next section, we'll describe the use of `latexml`, which performs the conversion to XMLThe following sections consider a sequence of successively more complicated postprocessing situations, using `latexmlpost`, by which one or more TeX sources can be converted into one or more web documents or a complete site.

Additionally, there is a convenience command `latexmlmath` for converting individual formula into various formats.

2.1 Basic XML Conversion

The command

latexml *options* --destination=*doc*.xml *doc*

converts the TEX document doc.tex, or standard input if - is used in place of the filename, to XML. It loads any required definition bindings (see below), reads, tokenizes, expands and digests the document creating an XML structure. It then performs some document rewriting, parses the mathematical content and writes the result, in this case, to doc.xml; if no --destination is suppplied, it writes the result to standard output. For details on the processing, see Chapter **??**, and Chapter **??** for more information about math parsing.

BIBTEX processing If the source file has an explicit extension of .bib, or if the --bibtex option is used, the source will be treated as a BIBTEX database.

Note that the timing is different than with BIBTEX and LATEX. Normally, BIBTEX simply selects and formats a subset of the bibliographic entries according to the .aux file; all TEX expansion and processing is carried out only when the result is included in the main LATEX document. In contrast, latexml processes and expands the entire bibliography when it is converted to XML; the selection of entries is done during post-processing. One implication is that latexml does not know about packages included in the main document; if the bibliography uses macros defined in such packages, the packages must be explicitly specified using the --preload option.

Useful Options The number and detail of progress and debugging messages printed during processing can be controlled using

--verbose *or* --quiet

They can be repeated to get even more or fewer details.

Directories to search (in addition to the working directory) for various files can be specified using

--path=*directory*

This option can be repeated.

Whenever multiple sources are being used (including multiple bibliographies), the option

--documentid=*id*

should be used to provide a unique ID for the document root element. This ID is used as the base for id's of the child-elements within the document, so that they are unique, as well.

See the documentation for the command latexml for less common options.

Loading Bindings Although LaTeXML is reasonably adept at processing TeX macros, it generally benefits from having its own implementation of the macros, primitives, environments and other control sequences appearing in a document because these are what define the mapping into XML. The LaTeXML-analogue of a style or class file we call a LaTeXML-binding file, or *binding* for short; these files have an additional extension `.ltxml`.

In fact, since style files often bypass structurally or semantically meaningful macros by directly invoking macros internal to LaTeX, LaTeXML actually avoids processing style files when a binding is unavailable. The option

`--includestyles`

can be used to override this behaviour and allow LaTeXML to (attempt to) process raw style files. [A more selective, per-file, option may be developed in the future, if there is sufficient demand — please provide use cases.]

LaTeXML always starts with the `TeX.pool` binding loaded, and if LaTeX-specific commands are recognized, `LaTeX.pool` as well. Any input directives within the source loads the appropriate binding: `\documentclass{article}` or `\usepackage{graphicx}` will load the bindings `article.cls.ltxml` or `graphicx.sty.ltxml`, respectively; the obsolete `\documentstyle{article}` directive is also recognized. An `\input` directive will search for files with both `.tex` and `.sty` extensions; it will prefer a binding file if one is found, but will load and digest a `.tex` if no binding is found. An `\include` directive (and related ones) search only for a `.tex` file, which is processed and digested as usual.

There are two mechanisms for customization: a document-specific binding file *doc*`.latexml` will be loaded, if present; the option

`--preload=`*binding*

will load the binding file *binding*`.ltxml`. The `--preload` option can be repeated; both kinds of preload are loaded before document processing, and are processed in order.

See Chapter **??** for details about what can go in these bindings; and Appendix **??** for a list of bindings currently included in the distribution.

2.2 Basic Postprocessing

In the simplest situation, you have a single TeX source document from which you want to generate a single output document. The command

`latexmlpost` *options* `--destination=`*doc*`.xhtml` *doc*

or similarly with `--destination=`*doc*`.html`, will carry out a set of appropriate transformations in sequence:

- scanning of labels and ids;
- filling in the index and bibliography (if needed);
- cross-referencing;

- conversion of math;
- conversion of graphics and picture environments to web format (png);
- applying an XSLT stylesheet.

The output format affects the defaults for each step and is determined by the file extension of `--destination`, or by the option

```
--format=(xhtml|html|xml)
```

html both math and graphics are converted to png images; the stylesheet `LaTeXML-html.xslt` is used.

xhtml math is converted to Presentation MathML, other graphics are converted to images; the stylesheet `LaTeXML-xhtml.xslt` is used.

xml no math, graphics or XSLT conversion is carried out.

Of course, all of these conversions can be controlled or overridden by explicit options described below. For more details about less common options, see the command documentation `latexmlpost`, as well as Appendix **??**.

Scanning The scanning step collects information about all labels, ids, indexing commands, cross-references and so on, to be used in the following postprocessing stages.

Indexing An index is built from `\index` markup, if `makeidx`'s `\printindex` command has been used, but this can be disabled by

```
--noindex
```

The index entries can be permuted with the option

```
--permutedindex
```

Thus `\index{term a!term b}` also shows up as `\index{term b!term a}`. This leads to a more complete, but possibly rather silly, index, depending on how the terms have been written.

Bibliography Bibilographic data from BibTeX can be provided with the option

```
--bibliography=bibfile.xml
```

The bibliography would have typically been produced by running

```
latexml --dest=bibfile.xml bibfile.bib
```

Note that the XML file, bibfile, is not used to directly produce an HTML-formatted bibliography, rather it is used to fill in the `\bibliography{..}` within a TEX document.

Cross-Referencing In this stage, the scanned information is used to fill in the text and links of cross-references within the document. The option

```
--urlstyle=(server|negotiated|file)
```

can control the format of urls with the document.

server formats urls appropriate for use from a web server. In particular, trailing `index.html` are omitted. (default)

negotiated formats urls appropriate for use by a server that implements content negotiation. File extensions for `html` and `xhtml` are omitted. This enables you to set up a server that serves the appropriate format depending on the browser being used.

file formats urls explicitly, with full filename and extension. This allows the files to be browsed from the local filesystem.

Math Conversion Specific conversions of the mathematics can be requested using the options

```
--mathimages                      # converts math to png images,
--presentationmathml or --pmml # creates Presentation \MathML
--contentmathml or --cmml      # creates Content \MathML
--openmath or --om             # creates \OpenMath
```

(Each of these options can also be negated if needed, eg. `--nomathimages`) It must be pointed out that the Content MathML and OpenMath conversions are currently rather experimental.

More than one of these conversions can be requested, and each will be included in the output document. However, the option

```
--parallelmath
```

can be used to generate parallel MathML markup, provided the first conversion is either `--pmml` or `--cmml`.

Graphics processing Conversion of graphics (eg. from the `graphic(s|x)` packages' `\includegraphics`) can be enabled or disabled using

```
--graphicsimages or --nographicsimages
```

Similarly, the conversion of `picture` environments can be controlled with

```
--pictureimages or --nopictureimages
```

An experimental capability for converting the latter to SVG can be controlled by

```
--svg or --nosvg
```

Stylesheet If you wish to provide your own XSLT or CSS stylesheets, the options

```
--stylesheet=stylesheet.xsl
--css=stylesheet.css
```

can be used. The `--css` option can be repeated to include multiple stylesheets; for example, the distribution provides several in addition to the `core.css` stylesheet which is included by default.

`navbar-left.css` Places a navigation bar on the left.

`navbar-right.css` Places a navigation bar on the left.

`theme-blue.css` Colors various features in a soft blue.

`amsart.css` A style appropriate for many journal articles.

To develop such stylesheets, a knowledge of the LATEXML document type is necessary; See Appendix **??**.

2.3 Splitting the Output

For larger documents, it is often desirable to break the result into several interlinked pages. This split, carried out before scanning, is requested by

```
--splitat=level
```

where *level* is one of `chapter`, `section`, `subsection`, or `subsubsection`. For example, `section` would split the document into chapters (if any) and sections, along with separate bibliography, index and any appendices. (See also `--splitxpath` in `latexml`.) The removed document nodes are replaced by a Table of Contents.

The extra files are named using either the id or label of the root node of each new page document according to

```
--splitnaming=(id|idrelative|label|labelrelative)
```

The relative foms create shorter names in subdirectories for each level of splitting. (See also `--urlstyle` and `--documentid` in `latexml`.)

Additionally, the index and bibliography can be split into separate pages according to the initial letter of entries by using the options

```
--splitindex  and --splitbibliography
```

2.4 Site processing

A more complicated situation combines several TEX sources into a single interlinked site consisting of multiple pages and a composite index and bibliography.

Conversion First, all TEX sources must be converted to XML, using `latexml`. Since every target-able element in all files to be combined must have a unique identifier, it is useful to prefix each identifier with a unique value for each file. The `latexml` option `--documentid=`*`id`* provides this.

Scanning Secondly, all XML files must be split and scanned using the command

```
latexmlpost --prescan --dbfile=DB --dest=i.xhtml i
```

where *DB* names a file in which to store the scanned data. Other conversions, including writing the output file, are skipped in this prescanning step.

Pagination Finally, all XML files are cross-referenced and converted into the final format using the command

```
latexmlpost --noscan --dbfile=DB --dest=i.xhtml i
```

which skips the unnecessary scanning step.

2.5 Individual Formula

For cases where you'd just like to convert a single formula to, say, MathML, and don't mind the overhead, we've combined the pre- and post-processing into a single, handy, command `latexmlmath`. For example,

```
latexmlmath --pmml=- \frac\b\pm\sqrt\{b^2-4ac\\}\2a\
```

will print the MathML to standard output. To convert the formula to a `png` image, say `quad.png`, use the option `--mathimage=quad.png`.

2.6 Optimized Processing

When one is interested in processing jobs beyond a single document, conversion efficiency, adaptability and throughput are of highest importance. Two such scenarios are processing jobs converting large collections of documents and multi-user web applications, building on LATEXML as a conversion backend.

The `latexmld` daemon, as well as the `latexmls` and `latexmlc` server-client setup can facilitate such purposes. Note that these binaries are in **beta stage** and are currently under heavy development.

```
latexmld options --destination=doc.xhtml doc
```

latexmls

latexmlc