International
Virtual
Observatory
Alliance

The IvoaTEX Document Preparation System

Version 1.2

IVOA Note 2018-01-30

Working group
Standards and Processes
This version
http://www.ivoa.net/documents/ivoatexDoc/20180130
Latest version
http://www.ivoa.net/documents/ivoatexDoc
Previous versions
Version 1.1
Version 1.0
Author(s)
Markus Demleitner, Mark Taylor, Paul Harrison, Marco Molinaro
Editor(s)
Markus Demleitner
Version Control
Revision 4724, 2018-01-29 15:20:52 +0100 (lun. 29 janv. 2018)
https://volute.g-vo.org/svn/trunk/projects/ivoapub/ivoatexDoc/ivoatexDoc.tex

Abstract

This note describes the IvoaTEX document preparation system for IVOA standards and notes. IvoaTEX supports the production of PDF and HTML renderings of the documents with sources in plain text suitable for version control, as is desirable for normative texts. This note contains a user guide as well as a discussion of IvoaTEX's dependencies and its implementation. It refers to version 1.0 of the software.

Status of this document

This is an IVOA Note expressing suggestions from and opinions of the authors. It is intended to share best practices, possible approaches, or other perspectives on interoperability with the Virtual Observatory. It should not be referenced or otherwise interpreted as a standard specification.

A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/documents/.

Contents

Acknowledgments

IvoaTeX heavily draws from experiences made with previous markup-based document preparation systems, in particular LaTeX classes and infrastructure created by Sebastién Derriere and Mark Taylor, as well as Paul Harrison's XML-based ivoadoc system.

We thank tth's author, Ian Hutchinson, for generous technical support and prompt provision of solutions in the upstream source where necessary.

1 Introduction

Creating and developing standards is a big part of the operations of the International Virtual Observatory Association (IVOA). As these are normative texts, attention to detail is very important, and being able to rigorously track changes to the documents is highly advantageous.

Standards are also often developed cooperatively, which means that capabilities for branching and merging are desirable. This strongly suggests employing version control systems for document authoring. Change tracking in software designed for editing office documents, to the extent it is supported at all, usually requires significant manual intervention, is optional, often used incorrectly, and frequently lacks interoperability. Led by these considerations, it was decided that IvoaTeX would have to be based on plain text source files.

As mandated by the IVOA Document Standards (Hanisch and Arviset et al., 2010), finished documents have to be at least available in PDF, while an additional HTML rendering for online use is recommended. A document preparation system should thus be able to produce documents in these formats in at least acceptable quality.

With these constraints in mind, several possible solutions were investigated. Paul Harrison's ivoadoc system[1] went for XHTML as an input format and used XSLT2 and XML-FO as document processors. While this facilitated several interesting features – for instance, automatic extraction and formatting of XML schema fragments or straightvorward embedding of RDFa markup for machine-readable examples –, it turned out that tooling issues were severe (e.g., reliable use of SGML catalogs[2], non-free hyphenation patterns, classpath issues) and the use of XML-FO for PDF generation yielded inferior renderings with little prospect for improvements by third parties. Also, authors disliked writing HTML tags.

Other options considered for source languages included docbook or one of the newer lightweight markup languages (ReStructuredText, markdown, etc). In each case, there were concerns either regarding the system's power and flexibility or its ease of installation and maintenance.

Meanwhile, several documents – to mention just a few, SAMP, VOTable, and VOUnits – had successfully used TEX-based systems typically derived from work done in the early 2000s by Sebastién Derriere. IVOATEX essentially is a generalisation of these standards' formatting systems, also inheriting from them the use of the `make` tool to automate workflows.

IVOATEX was extended to relieve document editors from some of the bookkeeping involved with producing IVOA standards and provide authors with uniform solutions for common problems in standards typesetting.

In the remainder of this document, we give quick-start instructions on installation and authoring in sect. 2 and continue a more thorough discussion of IVOATEX's facilities with a special focus on enabling proper automatic production of both HTML and PDF output in sect. 3. In sect. 4, additional details on the implemenation are given for the benefit of authors planning to extend IVOATEX. We close with a discussion of open issues and desirable developments.

2 Installation and Quick Start

2.1 Dependencies

IVOATEX is designed to work more or less out of the box on common POSIX-compliant systems; no non-free software is required for operation. Its main uncommon dependency is the tth translator, which is a relatively compact program written in highly portable C generated through lex used to translate LaTeX to HTML. As it is compact and portable, it is delivered with

[1] https://volute.g-vo.org/svn/trunk/projects/ivoapub/ivoadoc

[2] This is important as retrieval of DTDs and similar data from their commonly used system identifiers (i.e., typically W3C web servers) is at least undesirable and in practice causes massive delays in formatting due to rate limiting on the part of the W3C.

IvoaTEX and built on demand. Since IvoaTEX's tth may at times offer some enhancements over the upstream tth, using a system-installed tth is discouraged.

The remaining dependencies include:

- A LaTeX distribution with some commonly available packages (calc, graphicx, xcolor, ifthen, doc, paralist, url, natbib, caption, hyperref). It is recommended to install TeXLive.
- A sufficiently capable implementation of `make`, with GNU's implementation recommended.
- The XSLT1 processor `xsltproc` (a different processor can be used, but that would probably require custom make rules).
- The `gcc` compiler (another C compiler could be used; the central makefile should probably be amended to allow easier changes here).
- The `zip` archiver for package generation.
- `imageMagick` and `ghostscript` if vector graphics is to be processed (which includes the architecture diagram).
- `librsvg2-bin` to let imageMagick process SVG.

On Windows, it is recommended to run IvoaTEX within cygwin, where all dependencies can easily be installed from cygwin's repository.

On Debian-derived systems, the dependencies should be present after running a distribution-specific adaption of

```
apt-get install build-essential texlive-latex-extra zip xsltproc\
  texlive-bibtex-extra imagemagick ghostscript cm-super librsvg2-bin
```

(cm-super contains vector versions of computer modern fonts in T1 encoding), on RPM-based systems something like

```
yum install texlive-scheme-full libxslt make gcc zip\
  ImageMagick ghostscript
```

should pull in everything that is necessary.

With OS X, a convenient way to obtain the dependencies is to install MacPorts[3] and then run

```
port install ImageMagick libxslt ghostscript +full
```

[3] https://www.macports.org/

The canonical OS X TEX distribution is the MacTEX version of TEXLive[4]. It is also possible to build TEX using MacPorts (with `port install texlive`), but this may result in a slightly non-standard distribution.[5]

To see if the prerequisites are there and compatible with IvoaTEX, try building an updated version of this document from its volute source:

```
svn co https://volute.g-vo.org/svn/trunk/projects/ivoapub/ivoatexDoc
cd ivoatexDoc
make biblio  # update the bibliography
make forcetex # make a PDF ignoring timestamps
make ivoatexDoc.html # make an html document
make package # make a zipfile for IVOA submission
```

During HTML generation, various diagnostics both from tth and from xsltproc (unknown commands, unexpected end tags, and the like) are expected at this point and no reason for alarm; we work on reducing the amount of spurious error messages.

2.2 Basic IvoaTEX operation

For ease of installation and robustness, IvoaTEX for now is designed to be used from within a subdirectory of the directory containing the document sources (rather than being installed globally). Given that it is fairly compact, having one copy per document seems acceptable.

So, the first step to use IvoaTEX is to create a development directory:

```
export DOCNAME=SampleDoc
# this would be your document's short name, e.g., RegTAP, SIAv2)
mkdir $DOCNAME
```

The DOCNAME – which will turn up in URLs, standard identifiers, and the like – should be chosen to be both succinct and expressive, and it should not contain non-alphanumeric characters (the examples given here assume that, too). A name like `SimpleDALRegExt` probably marks the upper limit in terms of length.

While it is clearly preferable if authors use IVOA's designated common version control system – at this point, this is Volute[6] – from the outset of document development, it is possible to operate it locally as well.

[4] https://www.tug.org/mactex/

[5] See http://tex.stackexchange.com/questions/97183/. Also, MacPorts does add `texlive` as a dependency on many packages, and so frequently *insists* on trying to build it; if you want to prevent this, there is some discussion at http://comments.gmane.org/gmane.os.apple.macports.user/21526.

[6] http://volute.g-vo.org

2.2.1 Installation from Archive (without Volute)

Without version control, it is sufficient to obtain IVOATEX from a distribution site and unpack it into the future document directory:

```
cd $DOCNAME
curl http://soft.g-vo.org/ivoatex/ivoatex-latest.tar.gz \
  | tar -xvzf -k
```

2.2.2 Installation with SVN version control

While it is of course possible to keep IVOATEX in checkouts, too, the recommended and more elegant way is to use `svn:externals`.

```
export VOLUTEBASE="https://volute.g-vo.org/svn/trunk/projects"
export WG=?????
# this would be the Working Group name as represented in
# volute subdirectories: dal, dm, edu, grid, registry
svn import $DOCNAME $VOLUTEBASE/$WG/$DOCNAME
rm -r $DOCNAME
svn co $VOLUTEBASE/$WG/$DOCNAME $DOCNAME
cd $DOCNAME
svn propset svn:externals\
  "ivoatex $VOLUTEBASE/ivoapub/ivoatex" .
svn update
svn propset svn:ignore . --file ivoatex/svn-ignore.txt
```

This has the advantage that IVOATEX updates are automatically pulled in and that it is trivial to feed back bibliography additions and patches to IVOATEX itself.

2.2.3 Beginning the document

Main metadata in the Makefile For convenience, the document production should start from some common templates which are part of the IVOATEX distribution:

```
cp ivoatex/Makefile.template Makefile
cp ivoatex/document.template $DOCNAME.tex
cp ivoatex/archdiag-full.xml archdiag.xml # Notes don't need that
```

The next step is to fill out the makefile template. As of the formatting of this document, this template looks like this:

```
# ivoatex Makefile.  The ivoatex/README for the targets available.

# short name of your document (edit $DOCNAME.tex; would be like RegTAP)
DOCNAME = ????

# count up; you probably do not want to bother with versions <1.0
DOCVERSION = 1.0
```

```
# Publication date, ISO format; update manually for "releases"
DOCDATE = ???

# What is it you're writing: NOTE, WD, PR, REC, PEN, or EN
DOCTYPE = ???

# Source files for the TeX document (but the main file must always
# be called $(DOCNAME).tex
SOURCES = $(DOCNAME).tex role_diagram.pdf

# List of image files to be included in submitted package (anything that
# can be rendered directly by common web browsers)
FIGURES = role_diagram.svg

# List of PDF figures (figures that must be converted to pixel images to
# work in web browsers).
VECTORFIGURES =

# Additional files to distribute (e.g., CSS, schema files, examples...)
AUX_FILES =

include ivoatex/Makefile
```

All lines with question marks must be filled out. The document date is the publication date, which can be significantly different from the current date. After its initial setting, it should only be changed at the time of submission to the document archive. It is always in DALI-style ISO format (Dowler and Demleitner et al., 2013), e.g., 2014-03-31.

`SOURCES` is used in dependency processing. It would be amended when the source file is split into separate files or if material is included into the document, e.g., via `lstinputlisting`. Graphics files included do not need to be given here.

`FIGURES` must contain the names of all bitmap graphics included in the document; files missing here will be missing from the package for distribution to the IVOA document repository, which will break the HTML rendering. As to `VECTORFIGURES`, see sect. 3.4.2.

`AUX_FILES` is intended for files that should be included in the upload to the IVOA document repository while not taking part in the actual formatting. This in particular concerns XML Schema files, for which the IVOA maintains a separate repository, but is by no means limited to them.

Additional metadata in the LaTeX source The template for the TeX source contains several lines with multiple question marks. These must be filled out as well.

As illustrated in the template, both `author` and `previousversion` support an optional argument giving an URL; for `author`, it should normally point to the respective person's page in the IVOA wiki, for `previousversion`,

it should point to the landing page of the respective document version in the IVOA document repository. Further automation for maintaining document history certainly is desirable, and the authors welcome ideas for how this might look.

The Archtecture Diagram An architecture diagram is only necessary for documents on the recommendation track. Notes may have one, but since they themselves are not included on it, that is rather unusual. If you do not want to include an architecture diagram, remove `role_diagram.svg` from `FIGURES` and `role_diagram.pdf` from `SOURCES` in the makefile.

To prepare an architecture diagram, copy `ivoatex/archdiag-full.xml` to `role_diagram.xml`. Then edit `role_diagram.xml` and remove all references to standards unrelated to the current document. As a rule, all `rec` elements for standards not mentioned in "Role within the IVOA Architecture" should be removed. Finally, use a *`thisrec`* element for the current standard. An architecture diagram for VOResource would thus be specified like this:

```
<archdiag xmlns="http://ivoa.net/archdiag">
  <thisrec name="VOResource" x="55" y="155"/>
  <rec name="RegTAP" x="55" y="180"/>
  <rec name="RegistryInterface" x="55" y="205"/>
  <!-- and a few more -->
</archdiag>
```

The standard-specific architecture diagram will be built by saying `make role_diagram.svg` and can be viewed using, for instance, common web browsers. Please refrain from editing the SVG with vector graphics programs. In the PDF diagrams, a PDF version of the diagram is required. This is built using `make role_diagram.pdf`, and a stand-in will be built if your system lacks the software to do this conversion. It should be fine to use common vector graphic tools like inkscape to perform the conversion in that case. As long as the tooling situation regarding LaTeXand SVG is as unsatisfactory as it is, please commit both `archdiag.svg` and `archdiag.pdf` to the version control system.

Source code conventions IVOATEX requires the source to be written in UTF-8 encoding, since the references shipped with it are in UTF-8. Authors are urged to keep lines shorter than 72 characters in input files whenever possible in order to keep diffs useful and readable. When edits are made, paragraphs should not normally be reflowed to avoid large diffs for minor edits. Authors desiring a reflow after many edits are encouraged to concentrate them in a separate, reflow-only commit.

Note

Simply running `latex` or `pdflatex` directly (rather than through make) is *not* supported with IVOATEX. Due to the non-global installation of the support files, the TEX run needs a special environment that is prepared by the makefile.
Also note that bibliography processing must be initated manually by running `make biblio`; unless authors checked in the bbl file this produces, this must be run after a document checkout.

Makefile targets The PDF version of the document is built by the makefile's default rule, so running `make` will usually by enough. This will produce a file `$DOCNAME.pdf`.

Other makefile targets for author use include:

- `biblio` updates the bibliography (i.e., runs BIBTEX); running this is necessary after one of the bibliography files is updated or when a new publication is referenced from the document.
- `forecetex` rebuilds the PDF unconditionally (e.g., when TeX asks to be rerun)
- `$DOCNAME.html` generates an HTML rendering of the document; at this point, this will typically emit quite a bit of spurious diagnostics. Unfortunately, real problems may hide within. Therefore, we currently recommend a visual inspection of the resulting HTML before submission.
- `package` generates a zip file containing everything needed for publication in the IVOA's document repository. Obviously, `DOCVERSION`, `DOCTYPE`, and `DOCDATE` in the Makefile should be updated as necessary before this target is built. The result is a zip file with a name compliant to the IVOA document standards (Hanisch and Arviset et al., 2010).
- `generate` is used to update machine-generated content in the document, typically by the main editor. See section 3.9 for details.

2.3 Examples

Examples for IVOATEX use include this document[7] or RegTAP[8]. At the time of writing, about half a dozen IVOA documents within Volute are based on IVOATEX.

[7] https://volute.g-vo.org/svn/trunk/projects/ivoapub/ivoatexDoc
[8] https://volute.g-vo.org/svn/trunk/projects/registry/regtap

3 Authoring documents

While IvoaTeX documents can be written much like any other TeX document, it is advisable to follow certain standards and use special facilities for common appearance, easier development, and possible evolution of IvoaTeX itself.

3.1 IvoaTeX Features

IvoaTeX provides a small set of macros and environments designed to ease standards authoring. These include:

`author`, `previousversion`
: these are discussed in sect. 2.2.3.

`xmlel`
: a macro for marking up XML element or attribute names and similar.

`vorent`
: for a name taken from VOResource or its extensions, usually an element or attribute name.

`admonition`
: This is an environment for displayed boxes, intended for notes, tips, and the like. It takes an argument giving the head of the box, e.g.,

```
\begin{admonition}{Note}
Admonitions should not be overdone.
Also, they are floating insertions.
\end{admonition}
```

`bigdescription`
: This is an environment for definition lists in the style of HTML *dl*, and it will be translated into one. Use `\item[term]` for the term to be defined (the construct this item is in is a `bigdescription`).

The intention behind macros like `xmlel` and `vorent` is that such terms are typeset uniformly across documents. Further semantic markup like this is planned for future releases, and document authors are encouraged to contribute terms.

Also note that the title page is generated by the abstract environment. Thus, all IvoaTeX documents must have an abstract within the `abstract` environment.

3.2 Listings, Verbatim Material

IvoaTeX documents should use the `listings` package to include source code snippets, XML fragments and the like. It can be used after a set of declarations like

```
\usepackage{listings}
\lstloadlanguages{XML,sh}
\lstset{flexiblecolumns=true,tagstyle=\ttfamily,
  showstringspaces=False}
```

is included in the document preamble (i.e., before its `\begin{document}`). The languages preloaded should be adapted to the document's needs. Additional languages supported that are likely relevant in a VO context include C, fortran, python, SQL, and java, specified as above case-insensitively.

The setup in the example (flexible columns, tags in typewriter, no explits blanks even in strings) is recommended for including XML source.

Actual listings are obtained with code like

```
\begin{lstlisting}[language=XML]
<example id="empty"/>
\end{lstlisting}
```

Alternatively, entire files can be included like this:

```
\lstinputlisting[language=XSLT]{makeutypes.xslt}
```

In the PDF rendering, the listings are pretty-printed. In the HTML rendering, the content is, currently, simply included in *pre* elements.

If a more compact rendering of listings is desired, for instance, because larger portions of source code are required in the document, listings' `basicstyle` option should be used together with one of LaTeX standard size macros. This could be in the argument of `lstset` in the preamble for a global setting, or on a case-by-case basis as in

```
\begin{lstlisting}[language=tex,basicstyle=\footnotesize]
Other sizes include \tiny, \scriptsize, \small, \normalsize, \large
\end{lstlisting}
```

As of version 1.0 of IVOATEX, only `footnotesize` is actually formatted by the CSS embedded in the HTML document, and we believe listings should not be much smaller than that anyway. As the options are translated into CSS classes, it is fairly easy to add further formatting functionality on a document-by-document basis, though.

3.3 References and Bibliography

3.3.1 Built-in Bibliographies

IVOATEX documents should use natbib and BIBTEX to manage references. The package comes with two default bibliographies:

- `ivoatex/ivoabib.bib` containing records for many publications likely to be cited by IVOA documents. It also contains "historical" records for IVOA standards, the BibTeX tags of which start with `std:`. Do not use these any more in new documents, they are only present in order to not break legacy documents.

- `ivoatex/docrepo.bib` containing records for IVOA documents listed in ADS. At this writing, this concerns all recommendations. A selection of Notes will be added at a later point.

Changes to the document that introduce new references or changes to the bibliography require that `make biblio` is run before changes become visible.

IvoaTeX comes with a bibliography style of its own, derived from `agsm.bst`. The custom bibliography style was derived to optimise some types of sources uncommon outside of the VO community, in particular IVOA recommendations and notes. Users are welcome to improve `ivoatex/ivoa.bst`.

3.3.2 Citation Style

As usual in natbib, actual references are made through either writing `\citep{tag}`, yielding a form like "(Einstein 1905)", or `\citet{tag}`, yielding a form like "Einstein (1905)". IvoaTeX does not support variant forms of citep and citet (i.e., those with optional arguments) yet; they will work in PDF output but fail in HTML. Contributions to improve this are welcome.

To reference an IVOA recommendation, locate its bibcode either using ADS or directly within `ivoatex/docrepo.bib` and then use one of the cite macros. The preferred style is to introduce a short name for the standard once with a citation and then use that short name in the remainder of the document to have expressive texts not overciting. For instance,

```
IVOA Identifiers \citep{2007ivoa.spec.0314P} introduces URIs to
reference Registry records, which are typically transmitted in
VOResource \citep{2008ivoa.spec.0222P} format. Both VOResource
and IVOA Identifiers are based on various W3C standards.
```

will come out as

> IVOA Identifiers (Plante and Linde et al., 2007) introduces URIs to reference Registry records, which are typically transmitted in VOResource (Plante and Benson et al., 2008) format. Both VOResource and IVOA Identifiers are based on W3C standards.

Recommendations should always be cited in the last version available at the time of writing[9]. If works in progress (working drafts, proposed recom-

[9] If a published recommendation is missing in the bibliography, this can be fixed by cd-ing into the ivoatex folder and saying `make docrepo.bib`.

mendations are to be cited, authors should create a local bibliography (see below).

If a non-temporary source (like a journal article or a software programme) does not already have a record in `ivoabib.bib`, authors are welcome to contribute new records there.

3.3.3 Local Bibliographies

When a reference is really only relevant to a single document or is conceptually non-permanent – this, in particular, pertains to Working Drafts or Proposed Recommendations to be cited – should be kept in a local bibliography. To enable such a document-local bibliography, the bibliography declaration at the foot of the document needs to be changed to

```
\bibliography{ivoatex/ivoabib,ivoatex/docrepo,localrefs}
```

Then a file `localrefs.bib` is created and checked into the version control repository. An example BibTeX record for an in-progress document is

```
@Misc{wd:Datalink,
  author={Patrick Dowler and Francois Bonnarel and
    Laurent Michel and Tom Donaldson and David Languignon},
  editor={Patrick Dowler},
  howpublished={{IVOA Working Draft 22 October 2013}},
  title={DataLink},
  year=2013,
  url={http://ivoa.net/documents/DataLink/20131022/}
}
```

Note that *howpublished* contains the precise document date and the *url* points to the actual versioned landing page, not the generic one for the standard.

3.4 Graphics

3.4.1 Bitmap Graphics

IvoaTeX supports all bitmap graphics formats that pdflatex supports. In practice, authors are encouraged to restrict themselves to JPEG, PNG, and possibly GIF. Currently, identical images are used for both PDF and HTML renderings. The recommended pattern for figures is

```
\begin{figure}[th]
\begin{center}
\includegraphics[width=0.9\textwidth]{mydiagram.png}
\end{center}
\caption{A diagram of what this is about.}
```

```
\label{fig:mydiag}
\end{figure}
```

This gives LaTeX some leeway in placing the figure, defines the image size in units of the page width, and centers the image itself.

All bitmap graphics in a document must be listed in the makefile's `FIGURES` variable. If they are not, the HTML rendering will be broken.

3.4.2 Vector Graphics

The only vector graphics format supported in IvoaTeX is PDF. PDF files can be directly used in `includegraphics`. The names of such figures must be listed in the makefile's `VECTORFIGURES` variable.

From `VECTORFIGURES`, IvoaTeX arranges that, when a PDF figure `foo.pdf` is used, the HTML target depends on a file called `foo.png`. This PNG can be generated automatically by IvoaTeX using a combination of ghostscript and ImageMagick. It may sometimes be preferable to perform a custom conversion by hand (e.g., more compact representation with bilevel source images), in which case the pre-rendered PNG should be included in the version controled repository. This also has the advantage that neither ghostscript nor ImageMagick are build dependencies of the document.

3.5 Tables

In tables, rules should be used sparingly. The standard pattern for tables is something like

```
\begin{table}[th]
\begin{tabular}{p{0.35\textwidth}p{0.64\textwidth}}
\sptablerule
\textbf{Column Head}&\textbf{Another column head}\\
\sptablerule
A value & Another value\\
A value in row 2& And so on\\
\sptablerule
\caption{A sample table}
\label{table:extable}
\end{tabular}
\end{table}
```

The `sptablerule` used here inserts a horizontal rule with some extra spacing and will be rendered consistently in both PDF and HTML. It should not, as a rule, used between table rows, it is intended primarily to delimit the table itself as well as the the heading and the body.

3.6 Hyperlinks

While IVOATEX puts no restrictions on the usage of hyperref features, the preferred way to include links in IVOATEX documents is to use the `url` macro, i.e., use the URL itself as the anchor text. In this way, the link remains (to some extent) usable even if the document is printed. The alternative two-argument `href` should generally be avoided as it fails on paper. For instance,

```
(this is bad:) The \href{http://ivoa.net}{IVOA} has issued
\href{http://ivoa.net/documents}{many standards}.
```

would severely degrade when printed and is hence discouraged, whereas

```
The IVOA\footnote{\url{http://ivoa.net}} has issued many
standards, all of which can be retrieved from
\url{http://ivoa.net/documents}.
```

works properly on all of IVOATEX's target media.

With non-HTTP URIs, it is recommended to use hyperref's `nolinkurl` macro (rather than an unadorned `texttt` or similar); advantages include that line breaking is better with nolinkurl, less manual escaping is necessary, and, if desired, such URIs can be more easily styled. An example:

```
The value of the \xmlel{standardID} attribute will be
be \nolinkurl{ivo://ivoa.net/std/Registry#OAI-2.0}.
```

3.7 Editorial tools

When authoring standards, it is sometimes necessary to include editorial comments of the type "Need to clarify" or "Specification incomplete". We recommend to use the todonotes package for such pieces of text[10]. The recommended usage is like

```
...
\usepackage{todonotes}
...
\begin{document}
\todo{This is an example for a editorial note}.
```

This is an example for a editorial note

A rendering of such a to-do note is shown in this paragraph . In HTML output, only the simple `todo` macro without options is supported, and text is simply displayed inline right now. Note in particular that todonote's `obeyDraft` and `obeyFinal` package options are ignored. Although IVOATEX does not enforce it (yet), finished recommendations should have no todo items in them.

[10] Full documentation is available at http://www.tex.ac.uk/CTAN/macros/latex/contrib/todonotes/todonotes.pdf.

While todonotes is a useful tool for standards development, we discourage the use of packages to mark up changes, as maintaining such markup usually is very hard, and version control offers a more manageable solution to the problem such packages attempt to solve.

3.8 Version Control System Information

It is recommended to include basic metadata obtained from the version control system into IVOATEX Documents where available. Basic support for Volute (i.e., subversion) is built into IVOATEX. It is based on subversion's keyword substitution, which therefore needs to be enabled on the main document source file. While any keys might be used, IVOATEX's default support deals with:

```
svn propset svn:keywords "Date Rev URL" $DOCNAME.tex
```

(only Rev is mandatory). In the document itself, the text for subversion's replacement needs to be included. Inititally, this must look like this:

```
\SVN$Rev: 0 $
\SVN$Date: 0 $
\SVN$URL: 0 $
```

Subversion will, at every commit, enter the current values in a controlled fashion, which will then be picked up by IVOATEX's macros. Due to the way this information is processed by TEX, after adding the above text, a commit must be made before the document can be built again.

For non-English locales, subversion's substitution poses the problem that localized information will be entered into the document, which is unwelcome for many resaons (non-ASCII characters, ambiguous date formats). We therefore recommend to run SVN in the C locale, at least while working on IVOA documents. Users affected with on-C locales could in a Bourne-like shell use a construct like

```
alias svn='LC_ALL=C svn'
```

and forget about the problem.

3.9 Generated Content

Sometimes it is desirable to have parts of a document generated through some sort of ivoatex-external process. Examples include copying documentation from XML Schema files into the standard document or obtaining column metadata for standard data models from a live TAP_SCHEMA.

For such cases, IVOATEX offers a python script `update_generated.py`, which is executed by the `generate` make target. It simply looks for structured comments in the main document and replaces what is between them

with generated contents. The opening line consists of a TEXcomment introducer, a blank, the literal `GENERATED:`, another blank, and a command line. The closing line is a comment consisting of `/GENERATED`.

On running `make generate`, the material between the opening and the closing line is replaced by the output of the command.

For instance, in the following snippet the material between the comments was inserted by `make generate`:

% GENERATED: echo This is generated content
`This is generated content`

% /GENERATED

`make generate` will not replace any content in the document source if even just one command fails to execute as indicated by the command's return code; it is transactional in this sense.

In addition to allowing arbitrary shell commands, `update_generated.py` has a facility that allows calling special python functions from within documents: Commands starting with an exclamation mark ("!") are translated to calls to appropriately named python functions defined within `update_generated.py`.

Currently, there are two such builtin commands; both are somewhat experimental features that may change when more experience has been gathered as to their usefulness.

One command is `taptable`, which extracts documentation from a live `TAP_SCHEMA`[11]. The access URL of the live TAP service must be specified in an environment variable TAPURL in the Makefile, for instance

export `TAPURL=http://dc.g-vo.org/tap`

Then, in the document one can use the structured comment

% GENERATED: !taptable tap_ schema.tables
% /GENERATED

After a `make generate`, the LaTeX source for a table describing the columns of `tap_schema.tables` will be between the two comment lines; rerunning `make generate` will replace that content with a refreshed version.

The second command implemented is `schemadoc` that formats documentation for a type in an XML schema file. This only works properly if the XML schema defines a *`vm:targetPrefix`* element in the way pioneered by Ray Plante in VOResource.[12] With an appropriately instrumented schema, a document can say

[11] It is in use in the Discovering Data Collections Within Services IVOA Note.

[12] See `http://volute.g-vo.org/svn/trunk/projects/registry/VOResource` for an example of a schema file instrumented for schemadoc.

% GENERATED: !schemadoc SchemaSource.xsd MyType
% /GENERATED

to produce documentation for the schema type *MyType* within the XML schema file `SchemaSource.xsd`.

IvoaTeXwill never execute `update_generated.py` as part of a dependency chain; it is intended that `make generate` must always been manually triggered. On the one hand, this is because its dependencies cannot be generically modelled, given that arbitrary commands can be executed. Document authors are also discouraged from providing such dependency information – it is fairly common that content generation depends on the availability of external resources (e.g., databases or network services), and a document build should not fail just because these are unavailable.

We mention in passing that generated content puts potentially executable material into documents, which is of course an attack vector for malicious software. However, calling `make update` is no additional security risk, as presumably the author of the makefile is identical to the document author, and the makefile can already contain arbitrary commands that would be executed on the calling user's behalf.

4 Customisation and Development

This section discusses aspects of IvoaTeX that are more technical in nature. Authors with a modicum of TeX expertise are nevertheless encouraged to read it.

4.1 Technical Overview

The central files in IvoaTeX processing are

ivoa.cls
: The class file, inheriting from LaTeX's article class. The file defines the markup rules for PDF processing, including titlepage generation and extra macros and environments. Its content is ignored for HTML generation.

tthdefs.tex
: This file protects its contents from normal TeX processing by a `\iftth` conditional. This way, only tth sees definitions made here. Each special feature defined in `ivoa.cls` has a counterpart here, giving rules for translation to HTML. This usually encompasses emitting some HTML before and after the argument of a TeX construct, where material between `\begin{html}` and `\end{html}` is included literally in the HTML document.

tth-ivoa.xslt
: An XSLT stylesheet that postprocesses tth's output and performs some operations that would be inconvenient to implement in `tthdefs.tex`, in particular for the formatting of the opening material.

Makefile
: This makefile is included by the user makefile in the document directory proper. It defines the rules given above as well as some extra housekeeping rules like package building and building tth from its source.

4.2 Semantic Markup

In order to make it support rich, semantic markup, IVOATEX needs to be continuously developed. In particular, it is good practice to define macros for marking up values of certain datatypes, as with IVOATEX's `xmlel` and `vorent`. Thus, whenever a document has multiple instances of such values, authors should define macros and use these. For instance, RegTAP deals with lots of concepts from its own database schema and hence has

```
\definecolor{rtcolor}{rgb}{0.15,0.4,0.3}
\newcommand{\rtent}[1]{\texttt{\color{rtcolor} #1}}
```

in its document preamble to define markup for "RegTAP entity", whereas this note, as it mentions many words with a special meaning to TEX, has

```
\definecolor{texcolor}{rgb}{0.4,0.1,0.1}
\newcommand{\texword}[1]{\texttt{\color{texcolor} #1}}
```

Such macros will be included in IVOATEX itself rather than an individual document's preamble when they prove useful in multiple documents.

4.3 Custom Macros and Environments

The tth translator used by IVOATEX ignores `usepackage`. Many common packages are natively supported, but those that are not in general need specific handling, and sometimes support is somewhat spotty. For instance, the `nolinkurl` macro is not supported natively by tth, and hence in `tthdefs.tex` there is code to the effect of

```
\newcommand{\nolinkurl}[1]{%
  \special{html:<span class='nolinkurl'>}#1\special{html:</span>}}
```

When a document requires special markup, it is likely that different implementations will be necessary for PDF and HTML output. Using `iftth` the implementations for the current output mode can be selected (without the `newif` mentioned in the tth documentation, as that is already performed in `tthdefs.tex`.

For instance, RegTAP has many inline tables that need special spacing the PDF rendering, whereas normal tables will do for them in HTML. It therefore has in its preamble the definitions

```
\iftth
  \newenvironment{inlinetable}{}{}
\else
  \newenvironment{inlinetable}{\vskip 1ex\vfil
    \penalty8000\vfilneg%
    \hbox to\hsize\bgroup\hss}
  {\hss\egroup\vspace{8pt}}
\fi
```

4.4 Custom CSS

If you find you need custom CSS to fix HTML formatting, you should probably talk to IVOATEX's authors first. There are, however, legitimate cases when something needs extra styling in HTML that comes out right without further effort in the PDF output. In such cases, a custom CSS file can be added to a repository (it must then also be added to `SOURCES` in the Makefile in order for it to be delivered with the document package).

The document itself would then use the IVOATEX's `customcss` macro in its preamble with the CSS file name as an argument. For example, the source for this document says

```
\iftth
 \newcommand{\comicstuff}[1]{
    \begin{html}<span class="comic">#1</span>\end{html}}
\else
  \newcommand{\comicstuff}[1]{(HTML exclusive material)}
\fi
```

in its preamble. With this (and a CSS file that can be inspected in the distribution),

```
\comicstuff{If this is comic sans, your web browser is permissive.}
```

becomes: (HTML exclusive material)

4.5 Migration from Ivoadoc

To ease migration from documents authored in ivoadoc, IVOATEX comes with an XSLT stylesheet writing out a starting point for an ivoatex version. While many desirable features (e.g., extraction of titlepage information) are not implemented and translated tables are incomplete, the styleheet should still save time. For XHTML-compliant ivoadoc sources, the stylesheet is used like this:

```
xsltproc ivoatex/fromivoadoc.xslt ivoadoc_source.html
```

4.6 Maintenance of the Architecture Diagram

The IVOA architecture diagram is introduced in Arviset and Gaudet et al. (2010) to visualise the standards landscape. All IVOA recommendations should have an architecture diagram showing the current standard as well as the related standards within that landscape.

Within IvoaTeX, architecture diagrams are produced in Scalable Vector Graphics (SVG). The source is in `ivoatex/archdiag-full.xml`, specifying the location of the recommendations (in *rec* elements) and the documents on the recommendation track (in *prerec* elements). The figure has a design size of 800×600 "pixels". Note that SVG is not really pixel-based – the numbers are just convenient, unitless floating point numbers, and the conversion of these coordinates to physical ones is done at render time.

In the diagram, the standards shoud be limited to the inner box, i.e., the zone between 50, 100 and 750, 500. Standards boxes are 90×18 pixels large, which is set in the `format-standard` template in `make-archdiag.xslt`. When standard boxes are aligned, the vertical distance of their centers should be 25 pixels, the horizontal distance 100 pixels. To keep the diagram lively, standards not obviously grouped with others may be placed "off-grid".

The specifications in `archdiag-full.xml`, as well as those in the authors' `archdiag.xml`, are ad-hoc XML interpreted by the stylesheet `ivoatex/make-archdiag.xml`. That stylesheet is written such that all three levels of the architecture diagram can be created. The `Makefile` in `ivoatex` has the necessary rules, but in contrast to the author rules, they are expected to be executed *within* the ivoatex directory.

So, to create the full versions of the three levels of the architecture diagram, do something like

```
cd ivoatex
make archdiag-l0.svg
make archdiag-l1.svg
make archdiag-l2.svg
```

The resulting svg files can be viewed in common browsers or in vector graphics programs like inkscape. The latter can also be used to find good positions for new elements (cursor coordinates are shown in the footline, but the direction of the y axis is reversed versus the SVG coordinates). Please do not use graphical tools to edit the diagram itself – the goal of the current architecture is to make edits transparent and have a clear and simple specification of the standards themselves in a file producing meaningful diffs.

5 Desirable Features to be Implemented

A major drawback of IVOATEX's HTML output is that paragraphs are not actually marked up as such. Due to the TEX processing model, their reconstruction is non-trivial. Hence in the generated HTML, source-level paragraphs are rendered as text nodes separated by empty HTML paragraph elements. It would probably be possible to rectify this in the XSLT postprocessing.

An automated way to maintain the in-document history (i.e., the sequence of `\previousversions` in the preamble) and better support to generate the change log would be desirable.

The hardcoded and part-manual way to run LATEXand BibTEXshould be made smarter, perhaps along the lines of latex-make.

A Changes from Previous Versions

A.1 Changes from Version 1.1

- Added material on the SVG architecture diagram, removed references to old PNG-based workflow.

A.2 Changes from Version 1.0

- Changed google code URLs to volute.g-vo.org ones.
- Documented new facilities for generating material, with extra focus on auto-documenting XML schema.
- Added advice on citing IVOA recommendations.
- Re-targeting for IVOATEX1.0 (rather than 0.4 before)

References

Arviset, C., Gaudet, S. and the IVOA Technical Coordination Group (2010), 'IVOA architecture', IVOA Note.
http://www.ivoa.net/documents/Notes/IVOAArchitecture

Dowler, P., Demleitner, M., Taylor, M. and Tody, D. (2013), 'Data access layer interface, version 1.0', IVOA Recommendation.
http://www.ivoa.net/documents/DALI/20131129/

Hanisch, R. J., Arviset, C., Genova, F. and Rino, B. (2010), 'IVOA document standards, version 1.2', IVOA Recommendation.
http://www.ivoa.net/documents/DocStd/

Plante, R., Benson, K., Graham, M., Greene, G., Harrison, P., Lemson, G., Linde, T., Rixon, G., Stébé, A. and IVOA Registry Working Group (2008), 'VOResource: an XML Encoding Schema for Resource Metadata Version 1.03', IVOA Recommendation 22 February 2008, arXiv:1110.0515.
http://adsabs.harvard.edu/abs/2008ivoa.spec.0222P

Plante, R., Linde, T., Williams, R. and Noddle, K. (2007), 'IVOA Identifiers Version 1.12', IVOA Recommendation 14 March 2007, arXiv:1110.0512.
http://adsabs.harvard.edu/abs/2007ivoa.spec.0314P