NISO JATS 1.0 Preview Stylesheets: Technical Documentation

Journal Information
Journal ID: N/A
ISSN: N/A

Article/Issue Information
Publication date: 2012

NISO JATS 1.0 Preview Stylesheets: Technical Documentation

Mulberry Technologies, Inc.

This document provides technical information regarding the HTML and PDF Preview stylesheets for the NISO Journal Article Tag Suite Publishing 1.0 tag set. (See http://dtd.nlm.nih.gov/publishing/.) It is intended for technical users, that is, personnel responsible for installing, configuring, and maintaining or extending these stylesheets. For information intended for general users regarding the scope and application of these stylesheets, see the *Users' Guide* in this distribution. For instructions on how to run these stylesheets on Journal Publishing content using a basic configuration, see the *Quick Start Guide*.

General overview

The various tasks performed by stylesheets in this distribution have been distinguished so they may be mixed and matched to meet specific local requirements, including:

- Generating HTML, PDF, or XHTML output.
- Autoformatting structured citation elements (`element-citation`) according to different sets of guidelines (we include stylesheets supporting the NLM/Pubmed citation style and an APA-like citation style).
- Converting OASIS (CALS/Docbook) tables appearing in JATS documents into HTML (for display or conversion to PDF).
- Filtering content based on the values of `specific-use` attributes assigned to elements in the source data.

These various operations are performed in stages as follows:

Preprocesses

Preprocesses are all designed to perform operations needed by some processes but not others, so they can be excluded when not needed. Generally speaking it is possible to run more than one preprocess, as long as they do not directly conflict with one another. When they do, it can be expected that the first process in the chain to handle a particular element or structure in the input will be the one to take effect.

Some but not all variants of NISO JATS will use OASIS tables instead of HTML tables; in support of these, a stylesheet to convert OASIS tables into HTML (which is handled by the main preview stylesheets) is also provided as a pre-process. This is documented further below.

All preprocessing stylesheets accept arbitrary (valid) JATS Publishing 1.0 XML and emit the same format: they are "modified identity transformations", whose operations are limited to a particular subset of elements related to their functional requirements. (The OASIS tables stylesheet emits the subset of HTML tables used by JATS and recognized by the main stylesheets. The citation preprocessors accept accept arbitrary JATS input and produce JATS in "simple citation" form, on which see below.)

Because some preprocesses perform operations that are not practically achieved in modifiable, maintainable XSLT 1.0, they use XSLT 2.0 and

require an XSLT 2.0 processor. Others will work in either an XSLT 1.0 or 2.0 processor.

Brief descriptions of each preprocess appear below. Additionally, all stylesheets in this package are documented internally.

Main preview processes

There are two main preview stylesheets in this distribution. `jats-html.xsl` formats JATS Publishing 1.0 in HTML for display in web browsers.[1] `jats-xslfo.xsl` converts JATS Publishing 1.0 XML into XSL formatting objects, ready for rendering by an XSL formatting engine into PDF output.

Both main preview stylesheets accept a filtered form of JATS Publishing 1.0, in which structured citation elements have been replaced with formatted citations. The assumption here is that many processes will need to include some form of citation processing in order to format citations in an appropriate style. For purposes of this documentation, this filtered form can be called "Simple citation" JATS Publishing 1.0 (it is specified in more detail below). Some projects may have citations that are already formatted using `mixed-citation` elements, which provides for inline punctuation and hence formatting within the source data. Since all the citation preprocessors respect punctuation that is already in place, any of them may be used as the preprocessor in such cases.

If input data should happen already to conform to the "Simple citation" format, a citation preprocessor may be dispensed with altogether. Some projects may wish to do this — "unbundling" the citation formatting to manage it up front or by different means altogether — in order to be able to use a wider range of XSLT processors, including XSLT processors in web browsers such as Internet Explorer, Firefox or Safari, to render their previews directly. The main preview stylesheets are implemented in XSLT 1.0 in order to enable this.

Post-process

Some JATS Publishing 1.0 data may include MathML, so the rendering of MathML may be an issue. The PDF preview stylesheet simply passes MathML through to be rendered by the formatter; many commercial FO engines now support MathML. On the HTML side, however, things are more complex since MathML is not supported by generic web browsers displaying HTML.

Some current browsers now support MathML display in XHTML output. For these purposes, an HTML to XHTML converter has been provided as a post-process. More details are provided below.

Stylesheets in this package

While the preview stylesheets in this distribution are intended as fairly comprehensive solutions, the supporting stylesheets may serve simply as demonstrations of how to achieve common tasks. Projects should feel free to copy, modify or extend any of them in order to support local requirements better.

---

[1] Unfortunately, due to tradeoffs forced by discrepancies in their modeling of block vs inline elements in the Journal Publishing tag set and HTML, not all results of the HTML preview stylesheet can be guaranteed to be valid HTML. But none of the known issues have hindered correct rendering in browsers.

Summary of stylesheets, dependencies, inputs and outputs

Stylesheet	Purpose	Requires	Input format	Output format
Preprocessors				
`xslt/citations-prep/jats-PMCcit.xsl`	Formats unpunctuated citations according to NLM/Pubmed guidelines	XSLT 1.0	JATS Publishing 1.0	"Simple citation" JATS Publishing 1.0
`xslt/citations-prep/jats-APAcit.xsl`	Formats unpunctuated citations according to APA guidelines	XSLT 2.0	JATS Publishing 1.0	"Simple citation" JATS Publishing 1.0
`xslt/prep/jats-webfilter.xsl`	Excludes content with `specific-use` = "print-only"	XSLT 1.0	JATS Publishing 1.0	JATS Publishing 1.0
`xslt/prep/jats-printfilter.xsl`	Excludes content with `specific-use` = "web-only"	XSLT 1.0	JATS Publishing 1.0	JATS Publishing 1.0
`xslt/oasis-tables/oasis-exchange-html.xsl`	Converts OASIS Open tables into HTML	XSLT 2.0	OASIS tables	HTML tables
`xslt/oasis-exchange-support.xsl`	Provides functionality in support of `oasis-exchange-html.xsl` and `oasis-table.sch` (Schematron)	XSLT 1.0	JATS Publishing 1.0	JATS Publishing 1.0
Main JATS Publishing 1.0 Preview stylesheets				
`xslt/main/jats-html.xsl`	Formats Journal Publishing data in HTML	XSLT 1.0	"Simple citation" JATS Publishing 1.0	HTML (or XML-wf HTML)
`xslt/main/jats-xslfo.xsl`	Formats Journal Publishing data as PDF	XSLT 1.0, XSL-FO formatter[i]	"Simple citation" JATS Publishing 1.0	PDF
Postprocessor				
`xslt/post/xhtml-ns.xsl`	Converts HTML to XHTML (for support of W3C MathML)	XSLT 1.0	HTML	XHTML
Saxon shell ("wrapper") stylesheets				
`shells/saxon/jats-APAcit-html.xsl`	Formats Journal Publishing data in HTML with APA-like citations	XSLT 2.0/Saxon	JATS Publishing 1.0	HTML
`shells/saxon/jats-PMCcit-html.xsl`	Formats Journal Publishing data in HTML with NLM/Pubmed citations	XSLT 2.0/Saxon	JATS Publishing 1.0	HTML
`shells/saxon/jats-PMCcit-web-html.xsl`	Formats Journal Publishing data in HTML with NLM/Pubmed citations, excluding "print-only" content	XSLT 2.0/Saxon	JATS Publishing 1.0	HTML

`shells/saxon/jats-PMCcit-xhtml.xsl`	Formats Journal Publishing data with NLM/Pubmed citations in XHTML (allows for MathML)	XSLT 2.0/Saxon	JATS Publishing 1.0	XHTML
`shells/saxon/jats-APAcit-xslfo.xsl`	Formats Journal Publishing data as PDF with APA-like citations	XSLT 2.0/ Saxon, XSL-FO formatter [i]	JATS Publishing 1.0	PDF
`shells/saxon/jats-PMCcit-xslfo.xsl`	Formats Journal Publishing data as PDF with NLM/Pubmed citations	XSLT 2.0/ Saxon, XSL-FO formatter [i]	JATS Publishing 1.0	PDF
`shells/saxon/jats-PMCcit-print-fo.xsl`	Formats Journal Publishing data as PDF with NLM/Pubmed citations, excluding "web-only" content	XSLT 2.0/ Saxon, XSL-FO formatter [i]	JATS Publishing 1.0	PDF
`shells/saxon/jats-PMCcit-oasis-print-fo.xsl`	Formats Journal Publishing data as PDF with NLM/Pubmed citations and OASIS tables, excluding "web-only" content	XSLT 2.0/ Saxon, XSL-FO formatter [i]	JATS Publishing 1.0	PDF
XProc pipelines				
These provide the same conversions under the same names as the Saxon shells, except with a different suffix (`xpl` not `xsl`) since they are XProc pipeline specifications not (extended) XSLT 2.0 stylesheets. They are all located in the `shells/xproc` subdirectory.				

[i] Tested with AntennaHouse 4.3 with MathML support.

Running the stylesheets

Stylesheets included in this distribution make it possible to run processing pipelines easily. For various reasons, users and publishers may wish to modify these methods.

Using the provided shell stylesheets and pipelines

The easiest way to apply either of the preview stylesheets is to use one of the shell ("wrapper") stylesheets or XProc pipelines. The shell stylesheets can be run with Saxon, the XSLT 2.0 engine, and rely on Saxon-specific extensions to XSLT 2.0; the XProc pipelines can be run with any conforming XProc engine that supports XSLT 2.0 stages (no extensions are needed). Either process applies a set of transformations in sequence quickly, with a minimum of memory or disk overhead. Each one calls one or the other main preview stylesheet, with different pre- or postprocessors depending on its intended application.

The Saxon shell stylesheets provided also depend on the module `main/shell-utility.xsl` to function. This module must be in place, and its location in relation to the stylesheets named in the shell must be stable, for things to work.

Because running the XSL-FO Preview stylesheet also entails running an XSL-FO formatter, there are slightly different considerations in setting things up for each type of output:

HTML preview pipeline

Either from the command line or using an XML editor or XML/XSLT IDE, invoke an XProc processor with an XProc pipeline, or Saxon with a shell stylesheet, to apply it to a Journal Publishing 3.0 file or to a subdirectory of such files. If a DTD is referenced by the file, make sure it is available. Place the results in the location of your choice. Copy the `jats-preview.css` file next to the HTML results. You are done.

XSL-FO preview pipeline

Two alternatives are available:

- Configure your XSL-FO engine to run a transformation using an XProc engine or Saxon, and invoke the transformation from the formatter.
- Run the transformation in the same way as when producing HTML, to generate an XML XSL-FO file that can be processed by the XSL-FO formatting engine.

The tradeoff is that while the first method requires a (one-time) configuration, it is easier for the user. It is a common feature of XSL-FO engines to allow configuring them to use the XSLT transformation engine of your choice.[2]

Extending the Saxon shell stylesheet or writing your own

The shell stylesheets included in this package all have the same logic. Stylesheets to be called in and applied to the source data, in sequence, are named in a variable in the stylesheet. Inserting or removing components is as easy as changing this variable.

For example, the stylesheet `jats-PMCcit-xhtml.xsl` has the variable `$processes` declared as follows:

```
<xsl:variable name="processes">
  <step>../../xslt/citations-prep/jats-PMCcit.xsl</step>
  <step>../../xslt/preview/jats-html-preview.xsl</step>
  <step>../../xslt/post/xhtml-ns.xsl</step>
</xsl:variable>
```

Add a step to this declaration:

```
<xsl:variable name="processes">
  <step>../../xslt/prep/jats-webfilter.xsl</step>
  <step>../../xslt/citations-prep/jats-PMCcit.xsl</step>
  <step>../../xslt/main/jats-html-preview.xsl</step>
  <step>../../xslt/post/xhtml-ns.xsl</step>
</xsl:variable>
```

Now, an additional preprocess `prep/jats-webfilter.xsl` is run before the citation preprocessing step.

Of course, we recommend copying a shell with a new name and making changes there, in order to preserve the integrity of files in the distribution.

All paths given are relative to the shell stylesheet's location in the subdirectory structure.

Note that configuration of the output serialization using the `xsl:output` instruction is designated by the shell stylesheet. Accordingly, a shell stylesheet intended to generate plain

---

[2] The Journal Publishing 3.0 XSL-FO preview stylesheet was tested with the AntennaHouse XSL formatter, version 4.3. Instructions for configuring AntennaHouse to use an alternative XSLT processor can be found in their user documentation; the settings to change are under the **Format / Format Option Setting** menu option. A tip: test your invocation your processor from the command line first, so you can paste accurate settings into the dialog box.

HTML might have `method="html"` indicated in its `xsl:output`, while a shell stylesheet generating XHTML should have `method="xhtml"`.

Extending an XProc pipeline or writing your own

Of course, the same possibilities are available when using XProc, with many more. For example an XProc pipeline could included DTD or Schematron validation along with processing of documents (or only of valid documents, if preferred), processing sets of documents together, etc.

XProc is a W3C-standard XML process pipelining technology, documented on line.

Designing and running your own pipeline

If you don't want to use XProc or rely on the XSLT extensions supported by Saxon, you can build your own pipelines to apply the stylesheets, using the framework of your choice. Shell scripts, batch files, scripting languages, and build tools such as Unix `make` or Apache Ant can all be used for this purpose. The principle is simple: each step in the pipeline generates results that are taken as the input for the next step in the pipeline. Serialization may follow each step (if each pipeline reads its input from the file system) or may be performed only after the last step (if steps are connected using event streams or if intermediate results are held in memory).

Keep in mind that an XSLT engine will still be required to run the separate steps, and that several of the stylesheets provided do require an XSLT 2.0 processor.

Running without a pipeline

The stylesheets can also be run individually, without any pipeline automation at all, just by applying preprocessor stylesheets to the input, if needed, and then applying the appropriate main preview stylesheet to the output from the preprocessors.

For input that contains no bibliographic citations, or in which all the bibliographic citations are pre-formatted and conform to the "simple citation" format defined below, no preprocessors are necessary and the main preview stylesheet can be run standalone in any XSLT 1.0 or 2.0 processor (including client-side web browsers or other XSLT-capable display engines).

Note, however, that most JATS input that contains any citations at all will benefit from being passed through one of the citation preprocessors, even if all the citations in the input are `mixed-citation` and have the appropriate punctuation: the main preview stylesheets work best if the only elements within citations in the input are presentation-oriented elements; the preprocessors strip out all other elements inside of citations, and replace them with appropriate presentation-oriented markup.

Tweaks and extensions to the main preview stylesheets

These stylesheets provide for basic formatting of Journal Publishing data for preview; they are not designed to serve as production stylesheets. Their functionality can be extended or modified, however.[3] It is recommended that modifications make use of the XSLT import mechanism: write your modified templates in a separate stylesheet module that imports the main stylesheet module. The modifications can then be maintained separately in one place.

---

[3] The overall architecture of this stylesheet suite, with coverage of approaches to customizing it, was described in a paper at JATS-Con 2010. A written version is online at http://www.ncbi.nlm.nih.gov/books/NBK47104/.

CSS stylesheet

HTML results are generated with a link to `jats-preview.css`, given as a relative path that expects the named file to be in the same subdirectory as the HTML result file (or the XML source file if it is processed directly in a browser).

This can be altered at runtime by overriding the value of the `css` parameter supplied to the main HTML Preview stylesheet. This allows a CSS stylesheet with another name to be used or a stylesheet to be located elsewhere. If an empty string is provided, the HTML results will have no link to an external CSS.

Of course, altering `jats-preview.css` or providing your own CSS stylesheet is an easy way to alter the look and feel of output without editing any XSLT. Many of the basic settings in the HTML presentation, including fonts, rules, colors (including the colors of warnings and generated text) may be customized in this way.

Enabling autonumbering in the XSLT

As described in the *Users' Guide*, the Preview stylesheet does not generate labels with automatic numbering for elements in presentation, except the `fn` and `ref` elements as described there. However, the stylesheets are provided with variables that can be used as switches to turn on autonumbering, along with templates that can be modified (or overridden, in a customization layer) in order to control the formats of the numbers. See the stylesheets, especially the templates in mode `label-text`.

Support of OASIS tables

A conversion stylesheet is included in this distribution for converting OASIS Exchange XML tables, as specified in OASIS Technical Memorandum TR 9901:1999, into the subset of HTML tables used by JATS. This is in order that JATS variants using OASIS tables may also use these preview stylesheets; but the OASIS conversion logic will work on any data containing OASIS tables (in the namespace `http://docs.oasis-open.org/ns/oasis-exchange/table`), making them into HTML equivalents (not namespace-qualified). As with other components of this distribution, developers are free to extend and modify this stylesheet and to integrate it into their systems.

This module is designed so it can be imported into another (XSLT 2.0) stylesheet, although in this distribution it is presented as a separate pipeline step. (This is in order that the main preview stylesheets may still use XSLT 1.0, for the broadest possible support.) Developers familiar with the architecture of XSLT may wish to integrate it more tightly by calling it into another stylesheet (using `xsl:import`) instead of using it as a separate step.

In addition, this distribution also includes a Schematron schema for use with OASIS tables, which must be checked against constraints over and above validity to a DTD or schema. This Schematron will ensure that tables are "square" (rows and columns are all filled, allowing for implied table cells, row and column spanning etc.), that attributes are given consistently, and so forth. Since some analytic operations are provided using XSLT stylesheet functions, this Schematron requires a processor that permits foreign elements (in this case, XSLT elements) to be present.

Specification of "simple citation" NISO JATS 1.0

Projects that wish to dispense with automated formatting of citations, to extend the automated processing provided, or to develop their own citation formatting logic, will need to know the subset of Journal Publishing 3.0 that can be reliably converted by the preview stylesheets. This profile introduces additional constraints on elements related to bibliographic citations: `element-citation`, `nlm-citation`, `mixed-citation`

and their children (as well as the analogous `citation` element in NLM v2.3), along with the related elements `product`, `related-article` and `related-object`.

- `element-citation` and `nlm-citation` are not recognized or processed specially. A preprocessor should convert them into `mixed-citation`, providing their contents with punctuation in the process. (This is what the packaged citation preprocessors do.) If these elements appear, preview stylesheets will present them without formatting.
- Along with literal text (including punctuation), the following elements are recognized inside `mixed-citation`, `product`, `related-article` and `related-object`:

  `bold`, `italic`, `monospace`, `overline`, `roman`, `sans-serif`, `sc`, `strike`, `underline`, `inline-graphic`, `label`, `email`, `ext-link`, `uri`, `sub`, `sup`, `styled-content`

- The following elements are permitted in citation elements in Journal Publishing 3.0, but are *excluded* from the simple citation profile. A preprocessor should strip them from content or convert them into presentation-oriented elements (listed above), with appropriate punctuation, according to the rules of the chosen citation format. If they appear, these element can be expected to be processed by a preview stylesheet according to default rules, which generally means their content will appear with no formatting consequences:

  `abbrev`, `alternatives`, `annotation`, `article-title`, `chapter-title`, `chem-struct`, `collab`, `comment`, `conf-date`, `conf-loc`, `conf-name`, `conf-sponsor`, `date`, `date-in-citation`, `day`, `edition`, `elocation-id`, `etal`, `fpage`, `gov`, `inline-formula`, `institution`, `isbn`, `issn`, `issue`, `issue-id`, `issue-part`, `issue-title`, `lpage`, `milestone-end`, `milestone-start`, `month`, `name`, `named-content`, `object-id`, `page-range`, `part-title`, `patent`, `person-group`, `private-char`, `pub-id`, `publisher-loc`, `publisher-name`, `role`, `season`, `series`, `size`, `source`, `std`, `string-name`, `supplement`, `trans-source`, `trans-title`, `volume`, `volume-id`, `volume-series`, `year`