OASIS

OASIS Specification Publishing in DocBook XML Version 0.5

Working Draft 03

8 July 2010

Specification URIs:

This Version:

http://docs.oasis-open.org/templates/DocBook/spec-0.5/oasis-specification-0.5-wd03.html (Authoritative)
http://docs.oasis-open.org/templates/DocBook/spec-0.5/oasis-specification-0.5-wd03.pdf
http://docs.oasis-open.org/templates/DocBook/spec-0.5/oasis-specification-0.5-wd03.xml

Previous Version:

http://docs.oasis-open.org/templates/DocBook/spec-0.4/oasis-specification-0.4.html
http://docs.oasis-open.org/templates/DocBook/spec-0.4/oasis-specification-0.4.pdf
http://docs.oasis-open.org/templates/DocBook/spec-0.4/oasis-specification-0.4.xml

Latest Version:

http://docs.oasis-open.org/templates/DocBook/oasis-specification/oasis-specification.html (Authoritative)
http://docs.oasis-open.org/templates/DocBook/oasis-specification/oasis-specification.pdf
http://docs.oasis-open.org/templates/DocBook/oasis-specification/oasis-specification.xml

Technical Committee:

OASIS TC Administration

Author:

G. Ken Holman, Associate member <gkholman@CraneSoftwrights.com>

Related Work:

This methodology supersedes guidelines for the DocBook XML-based authoring and publishing of OASIS specifications described in http://docs.oasis-open.org/templates/DocBook/spec-0.4/oasis-specification.html.

Abstract:

This Working Draft 03 describes an environment for writing and publishing an OASIS specification using DocBook XML. It is an internal OASIS support document and not the basis of an OASIS specification in and of itself.

Status:

This is a work in progress contributed to the OASIS TC administration and does not at this time represent the consensus of any particular OASIS Technical Committee. There are no plans to make this a formal Committee Specification as it is merely an internal document made available to committee members to support the publishing process.

Please send comments on this document to G. Ken Holman at `<gkholman@CraneSoftwrights.com>`.

Notices

Table of Contents

Appendixes

1. Introduction

This document details an environment [Spec-0.5-ZIP] and methodology for writing an OASIS specification document using XML markup, and publishing the resulting document to HTML and printed results conforming to OASIS layout conventions.

While this has been prepared before, a new version of this environment and methodology is required to accommodate the latest conventions adopted by the OASIS TC Administration.

The stylesheets for this version of this environment will remain static while the prose methodology documents and samples may be revised from time to time. Should it be necessary to revise the stylesheets, this version of the environment will left unchanged on the OASIS web site and a new version of the stylesheets and prose documents will be started. Check the persistent location noted above for any revisions to this environment.

These stylesheets use XSLT [XSLT] as the transformation expression language to produce the final HTML result. XSLT is also used to produce the intermediate XSL-FO [XSL-FO] pagination semantics, which in turn are available to be processed to produce printable results.

1.1. Terminology

1.1.1. Key words

The key words *must*, *must not*, *required*, *shall*, *shall not*, *should*, *should not*, *recommended*, *may*, and *optional* are to be interpreted as described in [RFC Keywords]. Note that for reasons of style, these words are not capitalized in this document.

1.2. Normative References

[DocBook] OASIS Committee Draft 4.4 The DocBook 4.4 Document type [http://www.docbook.org/specs/cd-docbook-docbook-4.4.pdf], OASIS January 27, 2005, *OASIS DocBook 4.4 Document Type 27 Jan 2005 HTML [http://www.docbook.org/specs/cd-docbook-docbook-4.4.html], , DocBook 4.4 XML DTD [http://www.docbook.org/xml/4.4/docbook-xml-4.4.zip], , DocBook Stylesheets Version 1.69.1 [http://sourceforge.net/projects/docbook/files/docbook-xsl-doc/1.69.1/]*.

[Spec-0.5-ZIP] OASIS Specification Publishing in DocBook XML Version 0.5 ZIP package [http://docs.oasis-open.org/templates/DocBook/spec-0.5/oasis-specification-0.5-wd03.zip] 8 July 2010

[RFC Keywords] S. Bradner Key words for use in RFCs to Indicate Requirement Levels [http://www.ietf.org/rfc/rfc2119.txt] Internet Engineering Task Force, March 1997

[XSL-FO] Anders Berglund Extensible Stylesheet Language (XSL) Version 1.1 [http://www.w3.org/TR/2001/REC-xsl-20011015/] W3C Recommendation 5 December 2006

[XSLT] James Clark XSL Transformations (XSLT) Version 1.0 [http://www.w3.org/TR/1999/REC-xslt-19991116] W3C Recommendation 16 November 1999

1.3. Non-Normative References

[ant] Apache Software Foundation Ant Java-based Build Tool [http://ant.apache.org] (Another Neat Tool).

[FOP] Apache Software Foundation Formatting Objects Processor [http://xmlgraphics.apache.org/fop].

[Ibex Signature Edition] Visual Programming Limited. Ibex XSL-FO Processor [http://www.xmlpdf.com/ibex-downloads-signed.html].

[Saxon] Michael Kay Saxon 6.5.5 XSLT 1.0 Processor [http://saxon.sourceforge.net/saxon6.5.5/index.html].

[xjparse] Norm Walsh xjparse XML validation invocation [http://nwalsh.com/java/xjparse].

[xml-assoc] James Clark Associating Style Sheets with XML documents Version 1.0 [http://www.w3.org/1999/06/REC-xml-stylesheet-19990629]. W3C Recommendation 29 June 1999.

2. Installation and use

These stylesheets are zipped [Spec-0.5-ZIP] in a turnkey environment ready to unpack for offline use. It is used to validate DocBook XML, render HTML and create XSL-FO suitable for processing with an XSL-FO engine (which is not included). Of course one is not obliged to use the processors referenced as one can use any XML DTD validation tool and any XSLT processor.

The package can be downloaded from http://docs.oasis-open.org/templates/DocBook/spec-0.5/oasis-spec-ification-0.5-wd03.zip and is expected to be unzipped into a local directory for offline use. The examples in this documentation assume the package is unzipped in a Windows environment into the `z:\` directory, and in a Linux environment in one's home directory. For example purposes, it is assumed that the ZIP file contents (all files starting with `spec-0.5/`) are installed in the `z:\oasis\spec-0.5\` directory or `~/oasis/spec-0.5/`.

See Appendix C, *Publishing choreography and orchestration (Non-Normative)* for details on a set of invocations for validating OASIS specification documents and producing an HTML rendering suitable for web browsers and an XSL-FO result suitable for an XSL-FO engine. Neither a browser nor an XSL-FO engine are themselves included in the package.

2.1. Package contents

The significant directories of the package are as follows, shown for a Linux environment but identical in a Windows environment:

```
~/oasis/  - base directory for installation
  /spec-0.5                 - base directory for rendering
  /spec-0.5/css             - HTML appearance
  /spec-0.5/stylesheets     - HTML and XSL-FO production stylesheets
  /spec-0.5/template        - example specification in XML
  /spec-0.5/docbook         - base directory for DocBook DTD
  /spec-0.5/docbook/xsl     - base directory for stylesheet library
  /spec-0.5/docbook/xsl/fo  - base directory for XSL-FO library
  /spec-0.5/docbook/xsl/html - base directory for HTML library
```

The DocBook DTD components in the `docbook/` directory were obtained from http://www.docbook.org/xml/4.4/docbook-xml-4.4.zip.

The DocBook XSL components in the `docbook/xsl/` directory were obtained from http://prdown-loads.sourceforge.net/docbook/docbook-xsl-1.69.1.zip, where the base directory of the unzipped package is renamed "`xsl/`". This allows one to obtain another release of the DocBook stylesheets, unzip them into a directory, rename the base directory and replace the `xsl/` installation directory. The `xsl/fo/` and `xsl/html/` directories should be in place when the unpacking, renaming and replacing is completed.

2.2. Engaging the DocBook DTD

To assert in the online OASIS environment that your XML document satisfies the constraints of DocBook, point to the posted copy of the DTD:

```
<!DOCTYPE article
  PUBLIC "-//OASIS//DTD DocBook XML V4.4//EN"
"http://www.oasis-open.org/docbook/xml/4.4/docbookx.dtd"
```

To assert in an offline local environment that your XML document satisfies the constraints of DocBook, point to the installation copy of the DTD:

```
<!DOCTYPE article
  PUBLIC "-//OASIS//DTD DocBook XML V4.4//EN"
"file:///z:/oasis/spec-0.5/docbook/docbookx.dtd"
```

Validating the XML document using the embedded DTD reference will report any constraint violations in your XML.

2.3. The OASIS specification HTML stylesheets

Use this stylesheet in the offline environment to create the final online version to send to OASIS TC Administration (this will use embedded references to online CSS stylesheets and the online base URI in accordance with OASIS TC Admin requirements):

```
z:\oasis\spec-0.5\stylesheets\oasis-specification-html.xsl
```

Use this stylesheet during development and local review to create a test instance (this will use local CSS stylesheets and the local base URI):

```
z:\oasis\spec-0.5\stylesheets\oasis-specification-html-offline.xsl
```

In each case the resulting file should be suitable for viewing in a web browser.

2.4. The OASIS specification XSL-FO stylesheet

To convert a DocBook instance following the conventions used in this specification into an instance of XSL-FO assuming A4 page dimensions or US-letter page dimensions, use one of these stylesheets:

```
z:\oasis\spec-0.5\stylesheets\oasis-specification-fo-a4.xsl
z:\oasis\spec-0.5\stylesheets\oasis-specification-fo-us.xsl
```

The resulting file should be suitable for an XSL-FO engine to render into a printed format such as a PDF file.

Note

Some XSL-FO engines will automatically translate relative URI strings in links to absolute URI strings as part of the formatting process. This will leave absolute references to your local file system in the generated PDF, rather than the desired relative URI strings needed. Usually the resulting PDF, when not encrypted or locked, can be hacked to manually translate the absolute URI strings to the required relative URI strings.

2.5. Stylesheet invocation parameters

There are two invocation parameters with defaults:

- `automatic-output-filename=no`

  - when set to "`no`" the stylesheet processor invocation output target is not ignored

- when set to "`yes`" the stylesheets will automatically produce the output file named using the content of two child elements of `<article-info>` as follows:

    `<productname>-<productnumber>`

- `oasis-base=???` (HTML only)

    - when set to "`no`" there is no addition of a `<base>` metadata element in the HTML result

        - "`no`" is the default for `oasis-specification-html-offline.xsl`

    - when set to "`yes`" the stylesheets will create an HTML `<base>` metadata element extracting the URI from the first `<articleinfo><release-info>` element where the `role=` attribute contains the reserved partial string "`OASIS-specification-this`" and the URI contains the string "`.htm`"

        - "`yes`" is the default for `oasis-specification-html.xsl`

3. Authoring in XML

An important objective of using XML markup when writing content is to separate what you are writing from how it is formatted and presented. Moreover, describing the individual components of your writing uniquely can allow machine processing of your content. Such machine processing can identify constructs and process them individually as required for the processed result.

The vocabulary of XML markup is the level of granularity used to identify constituent information items, and the collection of element and attribute labels applied to the granules.

Applying styling to documents is an example of machine processing. The processed result is a formatted representation of your document. When many authors use the same vocabulary, or a given author creates many documents with the same vocabulary, a single set of processes will produce consistently formatted results across the document set.

This process is analogous to using styles found in most desktop publishing applications, however by re-moving from the author the ability to inject arbitrary formatting of information items in their content, two benefits are realized: (a) the author no longer needs to think about formatting, only about appropriately labeling the information items in the content; and (b) authors cannot inadvertently format components of a document with incorrect or inconsistent results.

Many writers do, however, prefer the use of their favorite desktop publishing applications and OASIS has posted at http://docs.oasis-open.org/templates/ a number of templates for widely-adopted word processing programs to be used in the creation of OASIS specifications. Users must be diligent in the use of styles in order to ensure their work conforms to the presentation conventions requested of OASIS. There is an obli-gation incumbent on the writer to verify their work has not violated the requested conventions, and there are no automated tools with which to validate the constraints have been respected.

When creating OASIS specifications using XML the burden of formatting is placed on the stylesheets, not on the writer. The obligation of the writer is only to be conformant to the document model for which the stylesheets have been designed, and there are automated validation tools with which the writer can validate the constraints have not been respected.

Any resulting violations to the formatting conventions can usually be ascribed to the stylesheets and, when repaired, the authored content usually does not need to change (barring any need to distinguish in the XML an ambiguity that is then being distinguished in the stylesheets).

3.1. DocBook vocabulary

This environment and methodology is based on using the `<article>` document type of the DocBook [DocBook] vocabulary for authoring the content in XML markup. These explanatory documents do not at-tempt to teach the DocBook vocabulary, as there are many online and printed references and manuals on the use of DocBook.

Note that there is a layer of additional constraints imposed on top of the DocBook metadata constructs in order to appropriately identify the constituent metadata components of the OASIS requirements for speci-fications. No validation constraints for this additional metadata layer are included in this environment, so it is the responsibility of the writer to visually confirm these needs have been satisfied in the rendered results. Missing components are typically the result of the writer not having correctly used the metadata indicators within the standard DocBook constructs.

The additional constraints are illustrated and documented in a pro forma template included with these ma-terials.

3.2. DocBook augmentations for OASIS specifications

These OASIS specification stylesheets recognize the following additional facilities or interpretations when using the DocBook vocabulary:

- `<articleinfo><productname>` (see Section 2.5, "Stylesheet invocation parameters")

- `<articleinfo><productnumber>` (see Section 2.5, "Stylesheet invocation parameters")

- `<articleinfo><releaseinfo role="OASIS-specification-this">`*{uri}*

  - reserved role identifying a URI belonging under the "This Version" heading

- `<articleinfo><releaseinfo role="OASIS-specification-previous">`*{uri}*

  - reserved role identifying a URI belonging under the "Previous Version" heading

- `<articleinfo><releaseinfo role="OASIS-specification-latest">`*{uri}*

  - reserved role identifying a URI belonging under the "Latest Version" heading

- release information role suffix `"-draft"`

  - the "*{uri}*" content is not interpreted as a URI but only simply as DocBook content

- release information role suffix `"-authoritative"`

  - the entry is suffixed with the string "(Authoritative)" as is required by OASIS conventions for one of the renditions

- `<articleinfo><releaseinfo role="committee">`*{text}*

  - identifying the technical committee section

- `<articleinfo><legalnotice role="related">`*{content}*

  - identifying the related work section

- `<articleinfo><legalnotice role="namespaces">`*{content}*

  - identifying the namespaces section

- `<articleinfo><legalnotice role="status">`*{content}*

  - identifying the status section

- `<articleinfo><legalnotice role="notices">`*{content}*

  - identifying the notices section

- `<phrase role="keyword">`*{text}*`</phrase>`

  - uppercase the text assuming the text is in English

- `<table role="font-size-XX">`

  - use a text font of XX for the content of cells in a table, as in `<table role="font-size-90%">`:

Table 1. Table with font size of 90%

row 1, column 1	row 1, column 2
row 2, column 1	row 2, column 2

or as in `<table role="font-size-50%">`:

Table 2. Table with font size of 50%

row 1, column 1	row 1, column 2
row 2, column 1	row 2, column 2

... rather than using the body font size:

Table 3. Table with no use of font size role attribute

row 1, column 1	row 1, column 2
row 2, column 1	row 2, column 2

- `<?pb?>`
  - inject a page break (PDF only)
- `<?lb?>`
  - inject a line break

3.3. Pro forma template

This environment includes a pro forma template of a DocBook XML instance with placeholders for all the OASIS specification metadata found in the word processing templates. The XML template has one of every item of metadata and can be used as a guideline when creating the metadata for a new document. The prose of the document includes a detailed narrative of the declaration and use of each of the metadata components.

The template also includes some limited guidelines to the use of the DocBook vocabulary that have been carried over from previous versions of the documentation. With time it is hoped to embellish more in this section for the benefit of the reader.

The pro forma template is included as part of the package of this environment and if the document you are reading is part of a completely installed collection then it can be found at template/spectools-docbook-template-wd03.xml and rendered at template/spectools-docbook-template-wd03.html. Please see Appendix B, *Sample specification template* for an analysis of the XML document.

3.4. Stylesheet association

Not included in the final published XML source of OASIS specifications are the stylesheet association [xml-assoc] processing instructions that are very handy conveniences to use during the authoring process. This methodology mandates their removal from the final published documents, but encourages their use when writing in order to streamline the writer's review of the formatted content.

Stylesheet association is not widely employed in the general XML community, typically due to (a) limited awareness by writers of their benefit; and (b) limited conforming support in web browsers.

An XSLT stylesheet association processing instruction is structured with two pseudo attributes as follows, indicating the type of the stylesheet being engaged and the location of the stylesheet file producing the HTML rendition:

```
<?xml-stylesheet type="text/xsl" href="place-URL-here"?>
```

The benefits of having embedded a stylesheet association processing instruction at the top of one's XML document are realized by opening the XML document being written in a web browser that (a) is aware of the processing instruction; (b) has a conforming XML processor with which to process the stylesheet files; and (c) has a conforming XSLT processor with which to engage the stylesheet files. The act of opening the XML document in the browser engages the HTML stylesheet against the XML content producing the HTML rendition on the browser canvas. At any time during the writing process, merely refreshing the web browser canvas re-engages the stylesheet producing the rendition of the revised content. This removes the necessity to engage the standalone invocation of the stylesheet environment to produce a reviewable rendition.

For example, this document was authored in an environment where the locally-installed offline version of the publishing environment is available in the local directory `../spec-0.5/`, thus allowing the following stylesheet association processing instruction placed at the top of the XML file before the document type declaration to render the document in an XSLT-aware web browser:

```
<?xml-stylesheet type="text/xsl" href="file:///z:/oasis/
spec-0.5/stylesheets/oasis-specification-html.xsl"?>
```

When a document is being authored in an environment where the remotely-installed online version of the publishing environment is available, the following stylesheet association processing instruction would perform equally as well (modulo the latency aspects of accessing resources over the Internet):

```
<?xml-stylesheet type="text/xsl" href="http://docs.oasis-open.org/
templates/DocBook/spec-0.5/stylesheets/oasis-specification-html.xsl"?>
```

Many people believe that stylesheet association is a transient property of an XML document and should not be included in any "official" normative XML content. This methodology, therefore, mandates that any such processing instruction that might be used by the author during the writing phase be removed from the final published XML instance.

Appendix A. Acknowledgments (Non-Normative)

This specification environment was written with the generous and appreciated assistance of Mary McRae, Will Norris and Jon Bosak.

Appendix B. Sample specification template

This section describes the instance structure and metadata of the DocBook XML instance template/spec-tools-docbook-template-wd03.xml, a template geared for use with the specification stylesheets. The summarized markup copied from the instance is split into successive `programlisting` fragments interspersed with a narrative regarding how to use the markup in the fragment. Braces "{}" are used to indicate replaceable information and the braces should not be in the final document. There is no obligation to use the techniques used below, though what is below is successfully producing the result you are reading.

For the purposes of example in the code below, the locally-installed copy of the OASIS publishing environment is in the directory `z:/oasis/spec-0.5/`.

The XML Declaration uses `ISO-8859-1` instead of `UTF-8` as the content is being typed and remembering the two-byte codes of `UTF-8` is sometimes awkward. Alternatively, using `US-ASCII` limits the typing entirely to 7-bit ASCII characters, requiring accented and other high-bit characters to be entered with entities.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

The following is an editing convenience keeping in an XML comment the PUBLIC and SYSTEM identifiers for the document type declaration for both the online publishing environment and the local publishing environment. Note how the actual identifiers in this snippet are for the local publishing environment while in the instance you will find the online publishing environment set being used in the document type declaration. When switching back and forth it is a convenience to have a few lines handy for copy/paste.

```
<!--
For online publishing use:
<?xml-stylesheet type="text/xsl"
href="http://docs.oasis-open.org/templates/DocBook/spec-0.5/stylesheets/
oasis-specification-html.xsl"?>
<!DOCTYPE article
  PUBLIC "-//OASIS//DTD DocBook XML V4.4//EN"
"http://www.oasis-open.org/docbook/xml/4.4/docbookx.dtd"

For offline publishing use:
<?xml-stylesheet type="text/xsl"
href="file:///z:/oasis/spec-0.5/stylesheets/oasis-specification-html.xsl"?>
<!DOCTYPE article
  PUBLIC "-//OASIS//DTD DocBook XML V4.4//EN"
"file:///z:/oasis/spec-0.5/docbook/docbookx.dtd"
-->
<?xml-stylesheet type="text/xsl"
href="http://docs.oasis-open.org/templates/DocBook/spec-0.5/stylesheets/
oasis-specification-html.xsl"?>
<!DOCTYPE article
  PUBLIC "-//OASIS//DTD DocBook XML V4.4//EN"
"http://www.oasis-open.org/docbook/xml/4.4/docbookx.dtd"
[
```

In the above the XML Stylesheet Association [xml-assoc] processing instruction used during the authoring process is a convenience for reviewing the rendered version of the authored XML markup when the XML document is open in a conforming browser supporting the association of XSLT stylesheets. After saving any edits to the XML document it is only necessary to refresh the browser to view the rendered results, without requiring any intervening XSLT process.

```
<?xml-stylesheet type="text/xsl"
href="file:///z:/oasis/spec-0.5/stylesheets/oasis-specification-html.xsl"?>
```

The internal declaration subset uses general entities in order to ensure consistent use of strings throughout the document. The `&name;` entity has the base name of the document. The `&version;` entity has the version of the document. The `&standard;` has the document status which is used in meta data and is often referenced in the prose along the lines of "`This &standard; includes the ...`".

```
[<!--the document properties-->
<!ENTITY name "spectools-docbook-template">
<!ENTITY pversion "0.4">
<!ENTITY version "0.5">
<!ENTITY stage "&stage;">
<!ENTITY standard "{stage and revision}">
<!ENTITY this-loc
"http://docs.oasis-open.org/templates/DocBook/spec-&version;/template">
<!ENTITY previous-loc
"http://docs.oasis-open.org/templates/DocBook/spec-&pversion;/template">
<!ENTITY latest-loc
"http://docs.oasis-open.org/templates/DocBook/oasis-specification/template">
<!ENTITY pubdate "{8 July 2010}">
```

The document element for OASIS standards is always `article`. The status is reflected in an attribute:

```
<article status="&standard;">
```

The metadata is kept in the DocBook `articleinfo` element, starting with the source code control, the name and the version of the document. The stylesheets can use the these information items in the synthesis of filenames (see Section 2.5, "Stylesheet invocation parameters"). Note the use of the entity references for consistency.

```
<articleinfo>

<title>{Specification Title Version X.X}</title>

<releaseinfo role="cvs">
$<!---->Id: spectools-docbook-template-wd03.xml,v 1.4 2010/07/08 12:28:15 admin Exp $
</releaseinfo>

<productname>&name;</productname>
<productnumber>&stage;</productnumber>
```

The links to the various instantiations of the document and supporting materials are summarized using `releaseinfo` elements with the reserved roles (see Section 3.2, "DocBook augmentations for OASIS specifications") for as many files as you have. OASIS rules require the identification of the authoritative publication. Note again the use of the entity references for consistency.

```
<releaseinfo role="OASIS-specification-this-authoritative"
>&this-loc;/&name;-&stage;.html</releaseinfo>
<releaseinfo role="OASIS-specification-this"
>&this-loc;/&name;-&stage;.pdf</releaseinfo>
<releaseinfo role="OASIS-specification-this"
>&this-loc;/&name;-&stage;.xml</releaseinfo>

<releaseinfo role="OASIS-specification-previous"
```

```
>&previous-loc;/&name;.html</releaseinfo>
<releaseinfo role="OASIS-specification-previous"
>&previous-loc;/&name;.pdf</releaseinfo>
<releaseinfo role="OASIS-specification-previous"
>&previous-loc;/&name;.xml</releaseinfo>

<releaseinfo role="OASIS-specification-latest-authoritative"
>&latest-loc;/&name;.html</releaseinfo>
<releaseinfo role="OASIS-specification-latest"
>&latest-loc;/&name;.pdf</releaseinfo>
<releaseinfo role="OASIS-specification-latest"
>&latest-loc;/&name;.xml</releaseinfo>
```

The committee is indicated using `role="committee"`.

```
<releaseinfo role="committee"><ulink url="http://www.oasis-open.org/
committees/">OASIS {official name of technical committee} TC</ulink>
</releaseinfo>
```

The various people sited in the document are in the `authorgroup` element, using `editor` and `author` for their respective players, and the `othercredit` element for the committee chairs.

```
<authorgroup>
<editor>
  <firstname>{Jane}</firstname><surname>{Doe}</surname>
  <affiliation>
    <orgname>{Example Corporation}</orgname>
    <address><email>{jane.doe@example.com}</email></address>
  </affiliation>
</editor>
<author>
  <firstname>{John}</firstname><surname>{Able}</surname>
  <affiliation>
    <orgname>{Other Example Corporation}</orgname>
  </affiliation>
</author>
<othercredit>
  <firstname>{Mary}</firstname><surname>{Baker}</surname>
  <affiliation>
    <address><email>{mary.baker@example.com}</email></address>
  </affiliation>
</othercredit>
<othercredit>
  <firstname>{James}</firstname><surname>{Butcher}</surname>
  <affiliation>
    <address><email>{james.butcher@example.com}</email></address>
  </affiliation>
</othercredit>
</authorgroup>
```

The publishing date is constrained by OASIS conventions to the following format:

```
<pubdate>&pubdate;</pubdate>
```

The copyright information is embedded using the `copyright` element, but note that this information has to be entered as well in the text of the notices.

```
<copyright>
  <year>YYYY</year>
  <holder>OASIS Open, Inc. All Rights Reserved.</holder>
</copyright>
```

When you want to relate this document to other works use `legalnotice` with `role="related"`.

```
<legalnotice role="related"><title>Related Work</title>
  <para>This specification replaces or supersedes:</para>
   ...
</legalnotice>
```

When you want to document the namespaces use `legalnotice` with `role="namespaces"`.

```
<legalnotice role="namespaces"><title>Declared XML Namespaces</title>
<para><ulink url="http://docs.oasis-open.org/">
http://docs.oasis-open.org/</ulink></para>
</legalnotice>
```

The abstract has its own DocBook element.

```
<abstract>
  <para>{This specification defines...}</para>
</abstract>
```

For the status use `legalnotice` with `role="status"`.

```
<legalnotice role="status"><title>Status</title>
  <para>{Describe the status and stability ...}</para>
   ...
</legalnotice>
```

For the notices use `legalnotice` with `role="notices"` ensuring you use the appropriate version of the legal notices from the markup in this instance.

```
<legalnotice role="notices"><title>Notices</title>
  <para>Copyright © OASIS Open 20??. All Rights Reserved.</para>
   ...
</legalnotice>
```

Thus ends the document metadata. The sections and annexes follow.

```
</articleinfo>
<section id="s.introduction">
...
<appendix>
...
```

Appendix C. Publishing choreography and orchestration (Non-Normative)

There is no obligation to automate the publishing process by choreographing the invocation of validation and production processes, however this methodology offers example orchestrations of these. Note that but for not having a core Ant [ant] task for the invocation of an XSL-FO processor, these guidelines would include such an example using that cross-platform Java-based build tool. It that absence, therefore, this section describes both a Windows command prompt and a Linux shell prompt that you may wish to configure for your own use.

The sample choreography invokes three separate processes in turn to produce the results. These processes are made available by Windows batch files and Linux shell scripts that are not included.

C.1. Windows batch file preparation

The example Windows batch file choreography in template/spectools-docbook-template.bat assumes the presence of three support batch files on the command path summarized as follows:

- ```
  call xmldtd.bat  input-XML-file
  ```

- ```
  call xslt.bat  input-XML-file  stylesheet-XSL-file  output-file
  ```

- ```
  call xslfo-pdf.bat  input-FO-file  output-PDF-file
  ```

C.2. Linux shell script preparation

The example Linux shell script choreography in template/spectools-docbook-template.sh assumes the presence of three support shell scripts summarized as follows:

- ```
  sh xmldtd.sh  input-XML-file
  ```

- ```
  sh xslt.sh  input-XML-file  stylesheet-XSL-file  output-file
  ```

- ```
  sh xslfo-pdf.sh  input-FO-file  output-PDF-file
  ```

C.3. Sample invocations

The stylesheets have been tested with Saxon for XSLT and Antenna House, Ibex and RenderX for XSL-FO.

While any conforming XML, XSLT and XSL-FO engine can be used in a given environment, the following Java-based applications and their respective invocations are suitable. There is no obligation, however, to use these as any conforming implementations of validation, transformation and pagination can be used. Note these guidelines leave it up to the reader to download the required support and to appropriately set the required CLASSPATH environments for the following invocations to work.

To validate that an XML instance conforms to the constraints of a formal DTD expression, xjparse [xjparse] can be used to satisfy the xmldtd script as follows:

```
java com.nwalsh.parsers.xjparse %1
```

To transform and XML instance using and XSLT expression, Saxon [Saxon] can be used to satisfy the `xslt` script as follows:

```
java -jar saxon.jar -o %3 %1 %2
```

To transform an XSL-FO instance into a PDF file, FOP [FOP] can be used to satisfy the `xslfo-pdf` script as follows (though note that there may be some conformance challenges with FOP that might present some problems):

```
java org.apache.fop.apps.Fop -fo %1 -pdf %2
```

To transform an XML instance into PDF in a single step, there is the `stylesheets/xslfo-xml-pdf.bat` example invocation file in the package. This invocation takes three arguments (in order): the XML input instance, "`us`" or "`a4`" (no quotes) to indicate the page size, and the name of the PDF output file. This requires the `saxon.jar` XSLT 1.0 executable[Saxon] and the Ibex Signature Edition created for stylesheets from Crane Softwrights[Ibex Signature Edition] (renamed as `ibex-crane-ss.jar`). The digital signature manifest `oasis-specification-fo-manifest.xml` is tied to the stylesheets as published, thus the processor will not work with these stylesheets if they are modified in any way. The support file `ibexconfig.xml` is used to configure font information.