Open Publication Structure (OPS) 2.0.1 v1.0.1

Recommended Specification September 4, 2010

THIS VERSION

http://www.idpf.org/doc_library/epub/OPS_2.0.1_draft.htm

LATEST VERSION

http://www.idpf.org/doc_library/epub/OPS_2.0_latest.htm

PREVIOUS VERSION

http://www.openebook.org/2007/ops/OPS_2.0_final_spec.html

DIFFS TO PREVIOUS VERSION

http://www.idpf.org/doc_library/epub/OPS_2.0.1_diffs_to_2.0.htm

Table of Contents

1.0: Overview

This specification, the Open Publication Structure (OPS), is one third of a triumvirate of modular specifications that make up the EPUB publication format. EPUB enables the creation and transport of reflowable digital books and other types of content as single-file digital publications that are interoperable between disparate EPUB-compliant reading devices and applications. EPUB encompasses a content container standard (Open Container Format – OCF), a packaging standard (Open Packaging Format – OPF), and this specification, a content markup standard.

1.1: Purpose and Scope

In order for electronic-book technology to achieve widespread success in the marketplace, Reading Systems need to have convenient access to a large number and variety of titles. The Open Publication Structure (OPS) Specification describes a standard for representing the content of electronic publications. Specifically:

- The specification is intended to give content providers (e.g. publishers, authors, and others who have content to be displayed) and publication tool providers, minimal and common guidelines that ensure fidelity, accuracy, accessibility, and adequate presentation of electronic content over various Reading Systems.
- The specification seeks to reflect established content format standards.
- The goal of this specification is to define a standard means of content description for use by purveyors of electronic books (publishers, agents, authors et al.) allowing such content to be provided to multiple Reading Systems and to ensure maximum presentational equivalence across Reading Systems.

Another related specification, the Open Packaging Format (OPF) Specification, defines the mechanism by which the various components of an OPS publication are tied together and provides additional structure and semantics to the electronic publication. Specifically, OPF:

- Describes and references all components of the electronic publication (e.g. markup files, images, navigation structures).
- Provides publication-level metadata.
- Specifies the linear reading-order of the publication.
- Provides fallback information for when extensions to OPS are employed.
- Provides a mechanism to specify a declarative global navigation structure (the NCX).

The OPF specification is separated from this OPS markup specification to modularize the described packaging methodology separate from the described content. This will help facilitate the use of the packaging technology by other standards bodies (e.g. DAISY) in non-OPS contexts.

A third specification, the Open Container Format (OCF) Specification, defines the standard mechanism by which all components of an electronic publication may be packaged together into a single archive for transmission, delivery, and archival purposes.

Together, these three standards constitute EPUB.

1.2: Definitions

CONTENT PROVIDER

A publisher, author, or other information provider who provides a publication to one or more Reading Systems in the form described in this specification.

DEPRECATED

A feature that is permitted, but not **recommended**, by this specification. Such features might be removed in future revisions. Conformant Reading Systems **must** support deprecated features.

EPUB

The publication format as defined by the OCF 2.0.1, OPF 2.0.1 and OPF 2.0.1 specifications.

EPUB PUBLICATION

A collection of OPS Documents, an OPF Package file, and other files, typically in a variety of media types, including structured text and graphics, packaged in an OCF container that constitute a cohesive unit for publication, as defined by the EPUB standards.

EPUB READING SYSTEM (OR READING SYSTEM)

A combination of hardware and/or software that accepts EPUB Publications and makes them available to consumers of the content. Great variety is possible in the architecture of Reading Systems. A Reading System **may** be implemented entirely on one device, or it **may** be split among several computers. In particular, a reading device that is a component of a Reading System need not directly accept OCF-Packaged EPUB Publications, but all Reading Systems **must** do so. Reading Systems **may** include additional processing functions, such as compression, indexing, encryption, rights management, and distribution.

EXTENDED MODULE

A module of a modularized XML vocabulary (i.e. a set of named modules is defined in its specification) that is not required to be supported by its specification (e.g. the XHTML ruby or forms modules in the OPS context).

INLINE XML ISLAND

An Inline XML Island is an XML document fragment using a non-Preferred Vocabulary or using an Extended Module that exists within an XHTML Preferred Vocabulary document within an OPS Publication.

NCX

A declarative global navigation definition (the Navigation Center eXtended or NCX).

NVDL

NVDL (*Namespace-based Validation Dispatching Language*) is a specification which allows for cross-schema validation of documents. This specification uses the NVDL language as a means of unambiguously defining how various schemas are handled within the context of OPS.

OCF

The Open Container Format defines a mechanism by which all components of an OPS Publication may be combined into a single file-system entity. OCF is defined by the OCF Specification.

OEBPS

The Open eBook Publication Structure. Previous versions of this specification (OPS) and its related specification, OPF, were unified into the single OEBPS specification. For this version, OEBPS was broken into OPS and OPF to aid modular adoption of the specifications. OEBPS 1.2 was the highest version of the previous unified specification.

OPF

The Open Packaging Format is the sister-standard to this standard. It defines the mechanism by which all components of a published work conforming to this standard along with metadata, reading order, and navigational information are packaged into an OPS Publication. OPF is defined by the OPF Specification.

OPF PACKAGE DOCUMENT

An XML document that describes the OPS contents of an EPUB Publication providing metadata, manifest, reading-order and navigation information for the publication.

OPS

The Open Publication Structure — this standard.

OPS CONTENT DOCUMENT

An XHTML, DTBook, or Out-Of-Line XML Island that conforms to this specification that may legally appear in an OPF Package Document `spine` element.

OPS CORE MEDIA TYPE

A MIME media type that all Reading Systems **must** support.

OPS PUBLICATION

A collection of OPS Content Documents, an OPF Package Document, and other files, typically in a variety of media types, including structured text and graphics, that constitute a cohesive unit for publication.

OUT-OF-LINE XML ISLAND

An Out-of-Line XML Island is an XML document, which exists within an OPS Publication, and is either not authored using a Preferred Vocabulary or is authored using a Preferred Vocabulary but uses Extended Modules. It is an entirely separate, complete, and valid XML document.

PREFERRED VOCABULARY

XML consisting only of OPS-supported XHTML modules and/or DTBook markup.

READER

A person who reads a publication.

READING DEVICE

The physical platform (hardware and software) on which publications are rendered.

READING SYSTEM

See EPUB Reading System.

XML DOCUMENT

An XML Document is a complete and valid XML document as defined in XML 1.0 (Fourth Edition) (http://www.w3.org/TR/2006/REC-xml-20060816/).

XML DOCUMENT FRAGMENT

Referred to as either a document fragment or as an XML Document Fragment, as defined in Document Object Model Level 1 (http://www.w3.org/TR/REC-DOM-Level-1/) but with the additional requirement that they be well-formed.

XML ISLAND

An Inline XML Island or an Out-of-Line XML Island.

XML NAMESPACES

Referred to as XML namespaces, or just namespaces, these must conform to XML Namespaces (http://www.w3.org/TR/2006/REC-xml-names-20060816/).

1.3: Relationship to Other Specifications

This specification combines subsets and applications of other specifications. Together, these facilitate the construction, organization, presentation, and unambiguous interchange of electronic documents:

1. *Extensible Markup Language (XML) 1.0 (Fourth Edition)* specification (http://www.w3.org/TR/2006/REC-xml-20060816/); and
2. *Namespaces in XML 1.0 (Second Edition)* (http://www.w3.org/TR/2006/REC-xml-20060816/); and

3. *Document Object Model (Core) Level 1* (http://www.w3.org/TR/REC-DOM-Level-1/level-one-core.html); and
4. *XML Pointer Framework* (http://www.w3.org/TR/xptr-framework/); and
5. *XHTML™ 1.1 - Module-based XHTML - Second Edition* specification (http://www.w3.org/TR/xhtml11/); and
6. *Specifications for the Digital Talking Book* (DTB) (http://www.niso.org/standards/resources/Z39-86-2005.html); and
7. *Scalable Vector Graphics (SVG) 1.1 Specification* (http://www.w3.org/TR/SVG11/); and
8. *Cascading Style Sheets, level 2* specification (http://www.w3.org/TR/REC-CSS2); and
9. *Unicode Standard, Version 4.0. Reading, Mass.: Addison-Wesley, 2003,* as updated from time to time by the publication of new versions. (See http://www.unicode.org/unicode/standard/versions for the latest version and additional information on versions of the standard and of the Unicode Character Database).; and
10. Particular MIME media types (http://www.ietf.org/rfc/rfc4288.txt and http://www.iana.org/assignments/media-types/index.html); and
11. *Associating Style Sheets with XML Documents* (http://www.w3.org/TR/xml-stylesheet); and
12. *Web Content Accessibility Guidelines 1.0* (http://www.w3.org/TR/WCAG10/); and
13. *RFC 2119: Key words for use in RFCs to Indicate Requirement Levels.* (http://www.ietf.org/rfc/rfc2119.txt); and
14. The OPF specification (http://www.idpf.org/doc_library/epub/OPF_2.0.1_draft.htm); and
15. *Namespace-based Validation Dispatching Language* (NVDL) (http://standards.iso.org/ittf/PubliclyAvailableStandards/c038615_ISO_IEC_19757-4_2006(E).zip)

1.3.1: Relationship to XML

OPS is based on XML because of its generality and simplicity, and because XML documents are likely to adapt well to future technologies and uses. XML also provides well-defined rules for the syntax of documents, which decreases the cost to implementers and reduces incompatibility across systems. Further, XML is extensible: it is not tied to any particular type of document or set of element types, it supports internationalization, and it encourages document markup that can represent a document's internal parts more directly, making them amenable to automated formatting and other types of computer processing.

- Reading Systems **must** be XML processors as defined in XML 1.0. All OPS Content Documents **must** be valid XML documents according to their respective schemas.

Reading Systems **may** support XML 1.1, but this feature is deprecated in version 2.0.1 (in favor of XML 1.0). Support for XML 1.1 will be removed in the next version of the specification.

1.3.2: Relationship to XML Namespaces

Reading Systems **must** process XML namespaces according to the XML Namespaces Recommendation at http://www.w3.org/TR/2006/REC-xml-names-20060816/.

Namespace prefixes distinguish identical names that are drawn from different XML vocabularies. An XML namespace declaration in an XML document associates a namespace prefix with a unique URI. The prefix can then be employed on element or attribute names in the document. Alternatively, a namespace declaration in an XML document **may** identify a URI as the default namespace, applicable to elements lacking a namespace prefix. The XML namespace prefix is separated from the suffix element or attribute name by a colon.

Example:

```
xmlns:ops="http://www.idpf.org/2007/ops"
```

The root element of all OPS Content Documents **must** explicitly specify the namespace of the document. For the XHTML Preferred Vocabulary, this namespace is *http://www.w3.org/1999/xhtml*. For the DAISY Talking Book Preferred Vocabulary, this namespace is *http://www.daisy.org/z3986/2005/dtbook/*. If the OPS namespace is used in a document it **must** be explicitly declared *http://www.idpf.org/2007/ops*. If a namespace prefix is used, it is **recommended** that authors bind the `ops` prefix to that namespace and not use `ops` as the prefix for other namespaces.

Example:

```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:ops=" http://www.idpf.org/2007/ops">
```

As OPS has additional functionality and validation requirements beyond the preferred document types and XML islands, there are other namespaces associated with OPS, which are used in specific contexts.

1.3.3: Relationship to NVDL

This specification uses the NVDL language (see http://standards.iso.org/ittf/PubliclyAvailableStandards/c038615_ISO_IEC_19757-4_2006(E).zip) as a means to unambiguously define the interaction between the various schemas used in this specification. NVDL allows for interaction and validation between various XML schema languages. See Appendix A for a normative NVDL definition of OPS.

This specification **does not** require the use of NVDL tools to validate OPS documents, although such tools are available and **may** be used for validation.

1.3.4: Relationship to XHTML and DTBook

This specification recognizes the importance of current software tools, legacy data, publication practices, and market conditions, and has therefore incorporated certain XHTML 1.1 Document Type Modules and DTBook as Preferred Vocabularies. This approach allows content providers to exploit current XHTML and DTBook content, tools, and expertise.

To minimize the implementation burden on Reading System implementers (who may be working with devices that have power and display constraints), the Preferred Vocabularies do not include all XHTML 1.1 elements and attributes. Further, the modules selected from the XHTML 1.1 specification were chosen to be consistent with current directions in XHTML.

Any construct deprecated in XHTML 1.1 is either deprecated or omitted from this specification; CSS-based equivalents are provided in most such cases. Style sheet constructs are also used for new presentational functionality beyond that provided in XHTML.

1.3.5: Relationship to CSS

This specification defines a style language based on CSS 2. (Note that the CSS 2.1 specification is currently still at "Working Draft" status.) The style sheet MIME type *text/x-oeb1-css* has been deprecated in favor of *text/css*.

The CSS-based style sheet constructs in this specification define required rendering functionality. To minimize the burden on Reading System developers and device manufacturers, not all CSS 2 properties are included. A few additional properties and values have been added to support page layout, headers, and footers. These, taken together, constitute the OPS CSS 2.0 required subset.

In a number of cases, this specification does not require Reading Systems to provide the full range of rendering that a standard CSS style sheet might request. For example, some Reading Systems will use monochrome displays. It would neither be acceptable to limit all Reading Systems to monochrome, nor to declare color use a non-standardized extension beyond OPS. In such cases, the CSS settings are allowed, and keep their meanings; but a conforming Reading System **may** gracefully degrade to a simpler rendering.

A conforming Reading System **must** render all OPS CSS 2.0 required subset properties. A Reading System **may** support CSS properties beyond the OPS CSS 2.0 required subset, however, any unsupported properties **must** be gracefully degraded per the CSS 2.0 specification.

This specification supports the `style` attribute (though deprecated), the **`style`** element, and externally linked style sheets. Reading Systems **must** perform XML-namespace handling while processing style sheets.

Style sheets can be associated with an OPS Content Document in several ways:

- by `style` attributes on specific XHTML elements (deprecated); and
- by **`style`** elements within the XHTML **`head`** element; and
- by an external style sheet identified on **`link`** elements in the XHTML **`head`** element; and
- by an external style sheet identified via the processing instruction **`xml-stylesheet`** (see Section 1.3.8).

The relative priority of the first three cases is as defined for XHTML 1.1 and CSS 2. Style sheets linked via a processing instruction are treated as if they had been linked via XHTML **`link`** elements preceding any actual XHTML **`link`** elements. As defined in the Conformance section, if no style sheet is defined or no applicable style is found for a given element, XHTML rendering is the default as defined elsewhere in this specification.

External style sheets linked via the XHTML **`link`** element or by the processing instruction **`xml-stylesheet`** **may** use CSS or any other style language, such as XSL (see http://www.w3.org/TR/xsl). Reading Systems are not required to support any style sheet language beyond the CSS specified herein.

Reading Systems that implement only the OPS CSS 2.0 required subset **may** ignore any style sheets using other style languages. Reading Systems that support extended style sheet functionality **may** choose among any of the other external style sheets. It is **strongly recommended** that unique MIME media types be defined for any style sheet languages supported beyond the OPS CSS 2.0 required subset, and that style sheets in those languages be detected by examining the MIME media type.

Use of the CSS position property values to achieve absolute positioning (i.e. *`absolute`* and *`fixed`*) is strongly discouraged.

1.3.6: Relationship to Unicode

Publications **may** use the entire Unicode character set, using UTF-8 or UTF-16 encodings, as defined by Unicode (see http://www.unicode.org/unicode/standard/versions). The use of Unicode facilitates internationalization and multilingual documents. However, Reading Systems are **not required** to provide glyphs for all Unicode characters.

Reading Systems **must** parse all UTF-8 and UTF-16 characters properly (as required by

XML). Reading Systems **may** decline to display some characters, but **must** be capable of signaling in some fashion that undisplayable characters are present. Reading Systems **must not** display Unicode characters merely as if they were 8-bit characters. For example, the biohazard symbol (0x2623) need not be supported by including the correct glyph, but **must not** be parsed or displayed as if its component bytes were the two characters "&#" (0x0026 0x0023).

To aid Reading Systems in implementing consistent searching and sorting behavior it is **required** that Unicode Normalization Form C (NFC) be used (See http://www.w3.org/TR/charmod-norm/).

1.3.7: MIME Media Types

This specification defines a list of OPS Core Media Types that all Reading Systems **must** support and publications **may** include. Publications **may** include resources of other media types, but each such resource **must** include an alternative resource of an OPS Core Media Type using methods defined in this specification or the OPF specification.

The OPS Core Media Types are:

MIME Media Type	Reference	Description
`image/gif`	http://www.w3.org/Graphics/GIF/spec-gif89a.txt	Used for raster graphics
`image/jpeg`	http://www.w3.org/Graphics/JPEG/	Used for raster graphics
`image/png`	RFC 2083	Used for raster graphics
`image/svg+xml`	http://www.w3.org/TR/SVG11/	Used for vector graphics
`application/xhtml+xml`	XHTML 1.1	Used for OPS Content Documents
`application/x-dtbook+xml`	http://www.niso.org/standards/resources/Z39-86-2005.html	Used for OPS Content Documents
`text/css`	CSS 2.0	Used for OPS CSS-subset style sheets
`application/xml`	http://www.w3.org/TR/xml11/	Used for Out-Of-Line XML Islands
`text/x-oeb1-document`	OEBPS 1.2 specification	Deprecated; Used for Basic or Extended OEBPS 1.0.1 and 1.2 Documents
`text/x-oeb1-css`	OEBPS 1.2 specification	Deprecated; Used for OEBPS 1.0.1 and 1.2 CSS-subset style

		sheets
`application/x-dtbncx+xml`	DTBook specification	The NCX

1.3.8: XML Style Sheet Processing Instruction

This specification includes support for the XML style sheet processing instruction `xml-stylesheet`, defined in the W3C Recommendation "Associating Style Sheets with XML Documents" (http://www.w3.org/TR/xml-stylesheet). This processing instruction is placed in the prolog of the XML document. It can appear multiple times as can `link`.

1.4: Conformance

The keywords "**must**", "**must not**", "**required**", "**shall**", "**shall not**", "**should**", "**recommended**", "**may**", and "**optional**" in this document **must** be interpreted as described in RFC 2119.

This section defines conformance for OPS Content Documents and Reading Systems.

1.4.1: OPS Content Document Conformance

This section defines conformance for OPS Content Documents.

1.4.1.1: OPS Content Document

A document is considered an OPS Content Document if and only if:

i. it uses a combination of the XHTML subset defined in this document and OPS-specific content extensions such as Inline XML Islands and Inline SVG; or
ii. it is a document with the MIME media type *`application/x-dtbook+xml`* which conforms to the DTB specification (http://www.niso.org/standards/resources/Z39-86-2005.html) and **must not** use OPS-specific content extensions such as Inline XML Islands or Inline SVG; or
iii. it is an XML document of any other MIME media type and is thus an Out-Of-Line XML Island

1.4.1.2: XHTML Content Document Requirements

A conformant XHTML Content Document **must** meet these conditions:

i. it is a well-formed XML document (as defined by XML 1.0); and
ii. it is encoded in UTF-8 or UTF-16; and
iii. it is a valid XML document according to the NVDL schema interaction provided in Appendix A; and
iv. it has a MIME media type of either *`application/xhtml+xml`* or *`text/x-oeb1-document`* (deprecated); and
v. all XHTML elements and attributes not contained in an Inline XML Island are drawn from the XHTML subset identified in this document.

1.4.1.3: DTBook Content Document Requirements

A document is a DTBook Content Document if and only if:

i. it is a well-formed XML document (as defined by XML 1.0); and

ii. it is encoded in UTF-8 or UTF-16; and
iii. it is a valid XML document, according to the DTBook DTD (http://www.daisy.org/z3986/2005/dtbook-2005-2.dtd); and
iv. it has a MIME media type of `application/x-dtbook+xml`.

1.4.1.4: Out-of-Line XML Island Content Document Requirements

A document is an Out-of-line XML Island Content Document if and only if:

i. it is a well-formed and valid XML document (according to its schema); and
ii. it is encoded in UTF-8 or UTF-16; and
iii. it has a MIME media type other than `application/xhtml+xml` (and is not an extended XHTML module), `text/x-oeb1-document` or `application/x-dtbook+xml`.

1.4.1.5: CSS Requirements

External CSS stylesheets referenced by OPS documents must use UTF-8 or UTF-16 encoding. Inclusion of Byte Order Mark (BOM) is mandatory for UTF-16 encoding and optional for UTF-8 encoding.

1.4.2: Reading System Conformance

This specification defines only one level of conformance for a Reading System. A Reading System is conformant if and only if it processes documents as follows:

When presented with an OPS Content Document the Reading System **must**:

i. correctly process the XML as required in the XML 1.0 specification, including that specification's requirements for the handling of well-formedness errors; and
ii. recognize all markup described as permitted in this specification and processes it consistently with the corresponding explanations in this specification and in those of XHTML 1.1, CSS 2, and DTBook (in case of any conflict, this specification takes precedence); and
iii. not render **`img`** or **`object`** elements of unsupported media types, in the absence of fallbacks. These fallbacks are clearly defined herein — **`img`** in Section 2.3.4 and **`object`** in Section 2.3.6; and
iv. verify the existence of the appropriate namespace specifications, as defined in the Relationship to XML Namespaces section above.
v. correctly process CSS stylesheets encoded in UTF-8, with or without a Byte Order Mark (BOM), and encoded in UTF-16, with a BOM.

1.4.3: Compatibility with Future Versions

It is the intent of the contributors to this specification that subsequent generations of this specification continue in currently established directions. Specifically:

- Content format standards will be compatible with W3C, IETF, and other applicable standards; and
- Future versions of this specification are expected to continue to improve alignment with XML-based specifications, requiring further XML processing capability of OPS-conformant Reading Systems, and enabling support for other applicable XML-related standards; and
- Future versions of this specification are expected to improve the richness of hyperlinking in OPS documents; and

- Any required functionality not present in relevant official standards shall be defined in a manner consistent with its eventual submission to an appropriate standards body as extensions to existing standards; and
- Continued alignment with the affiliated OPF specification; and
- Continued alignment and integration with the OCF content packaging standard.

1.4.4: Compatibility of OPS Version 2.0

Version 2.0 of OPS offers fairly significant enhancements over its preceding version, OEBPS 1.2. Version 2.0 adds substantial functional enhancements over 1.2, these include: supporting enhanced control over content presentational fidelity, providing an enhanced extension mechanism, improving accessibility, and improving alignment with the standards upon which OPS is based. Specifically, the following are the most substantive additions:

- The XML processing bar has been raised from XML "well-formedness" to XML validity.
- XML namespace processing is now required.
- The concept of Basic and Extended documents has been removed and replaced with two XML Preferred Vocabularies and a new XML island-based extension mechanism.
- The OPS subset of the XHTML element set has been replaced with reference to specific modules within XHTML 1.1.
- The DTBook element set is now a Preferred Vocabulary.
- Support for SVG has been added.
- Support for embedded fonts has been added.
- The OEBPS 1.2 Package has been extended for enhanced navigation and accessibility and the documentation thereof has been moved into the, now separate, OPF specification.
- Older OEBPS-specific MIME types have been deprecated and replaced with more widely used standard MIME types.

While most changes from version OEBPS 1.2 to OPS 2.0 have been done via deprecation rather than removal of previous functionality, OEBPS version 1.2 is not a fully compatible subset of OPS 2.0 (e.g. new XML validity and namespace processing requirements).

1.5: Extensibility

XML Islands are the **recommended** mechanism for adding functionality, information or structure beyond that supported by the Preferred Vocabularies.

Out-Of-Line XML Islands **may** be used in any publication. Inline XML Islands **must not** be used within documents authored using other than the XHTML Preferred Vocabulary.

1.6: Accessibility

This specification incorporates features that ensure content can be made accessible to, and usable by, persons with reading disabilities. Existing accessibility features developed by the World Wide Web Consortium (W3C) for XHTML 1.1 for content accessibility are incorporated into this specification.

OPS XHTML Content Documents **should** be authored in accordance with the W3C Web Content Accessibility Guidelines 1.0 (http://www.w3.org/TR/WCAG10/) or, the Web Content Accessibility Guidelines 2.0 (http://www.w3.org/TR/WCAG20/) to ensure that the broadest possible set of users will have access to books delivered in OPS format.

In addition, recommendations from the W3C HTML 4.0 Guidelines for Mobile Access (http://www.w3.org/TR/NOTE-html40-mobile/) and the W3C Web Accessibility Initiative's proposed User Agent Guidelines (http://www.w3.org/TR/WD-WAI-USERAGENT/) ought to be reviewed and applied by OPS implementers to ensure that Reading Systems will be in conformance with accessibility requirements.

1.7: Future Directions

This specification is designed to take advantage of current practices while preparing for future developments. Although details of subsequent versions of this specification remain to be determined, it is the expectation of the Publication Structure Working Group that continued evolutionary development will occur. The themes driving the creation of version 2.0 of OPS are: standards compliance (e.g. full namespace support), accessibility support, support for a wide-range of XML document vocabularies, enhanced navigation support, and improved content presentational fidelity.

Other themes deemed important for future versions include: more rigorous separation of content and presentation, greater accessibility, better support for international content, Reading Device-specific presentation control and/or Reading Device profiles, enhanced support for inter-Publication linking, layering and managing markups (e.g. inking, highlighting, notes) within Publications, application-specific markup (e.g. math, chemical), multiple reading orders, and support for active content (e.g. multimedia, scripting), all while maintaining alignment with relevant standards. Additionally, maintaining backward compatibility to this version of this specification ought to remain a high priority. Future directions can be tracked at http://www.idpf.org.

2.0: OPS Content Document Vocabularies

2.1: Introduction

OEBPS 1.2 and its predecessors specified a "Basic" document vocabulary, drawn from the XHTML 1.1 document type, which all Reading Systems were required to support. This specification no longer creates its own subset of XHTML 1.1, but instead references entire XHTML modules, as described in XHTML Modularization 1.1 (http://www.w3.org/TR/xhtml-modularization/). In addition to XHTML, OPS adds the DAISY DTBook document type as a Preferred Vocabulary (see http://www.niso.org/standards/resources/Z39-86-2005.html).

The XHTML modules listed in this section and the DTBook vocabulary are considered the OPS Preferred Vocabularies. The concepts of "Basic" and "Extended" documents are no longer used by this specification. OPS publications **should** use one of the Preferred Vocabularies. Publications that use other vocabularies, either as entire documents or as XML fragments, **must** use the Inline or Out-Of-Line XML Island mechanisms, described elsewhere in this specification. XHTML modules beyond those listed here (Extended Modules) **may** be used in a conforming publication, but **must** follow the guidelines set forth in the XML Islands section of this document.

As with any DTD referenced from the DOCTYPE declaration, OPS XHTML Content Documents **must not** reference the XHTML DTD unless such documents are valid with respect to that DTD (i.e. it does not include Inline XML Islands or inline SVG).

2.2: XHTML Modules in the OPS Preferred Vocabulary

2.2.1: Required Modules

OPS requires all conforming Reading Systems to support the following modules consistent with their descriptions in the XHTML and HTML specifications, unless otherwise specified in this document.

XHTML 1.1 Module Name	Elements (non-normative)
Structure	**body**, **head**, **html**, **title**
Text	**abbr**, **acronym**, **address**, **blockquote**, **br**, **cite**, **code**, **dfn**, **div**, **em**, **h1**, **h2**, **h3**, **h4**, **h5**, **h6**, **kbd**, **p**, **pre**, **q**, **samp**, **span**, **strong**, **var**
Hypertext	**a**
List	**dl**, **dt**, **dd**, **ol**, **ul**, **li**
Object	**object**, **param**
Presentation	**b**, **big**, **hr**, **i**, **small**, **sub**, **sup**, **tt**
Edit	**del**, **ins**
Bidirectional Text	**bdo**
Table	**caption**, **col**, **colgroup**, **table**, **tbody**, **td**, **tfoot**, **th**, **thead**, **tr**
Image	**img**
Client-Side Image Map	**area**, **map**
Meta-Information	**meta**
Style Sheet	**style**
Style Attribute (deprecated)	`style` attribute
Link	**link**
Base	**base**

2.3: Certain Element and Attribute Semantic Differences From, and Restrictions Beyond, XHTML 1.1

As previously noted, the semantics and rendering behavior of the XHTML Preferred Vocabulary (elements, attributes, and associated attribute values) strictly follows that of XHTML 1.1. However, there are several restrictions beyond that of XHTML 1.1, as noted below. These restrictions have no effect on the XHTML 1.1 conformance of OPS documents.

2.3.1: General Comments on URI References

A number of attributes reference resources using URI values (Uniform Resource Identifier, see RFC 2396, http://www.ietf.org/rfc/rfc2396.txt). Depending on the particular attribute, the URI referenced resource can either be an abstract entity or a physical object.

Except where noted or where not applicable, Reading Systems **may** use or render a URI referenced physical resource not listed in the Manifest (i.e., it is not a component of the Publication), but they are **not required** to do so.

2.3.2: `body` Element

It is assumed, in formatting, that the default rendering for **`body`** is consistent with the CSS property **`page-break-before`** having been set to *`right`* (which behaves like *`always`* on one-page Reading Systems), but **may** be overridden by an appropriate style sheet declaration.

2.3.3: `cite` Attribute

The **optional** attribute `cite` **may** be used in **`blockquote`**, **`q`**, **`del`**, and **`ins`** to provide a URI citation for the element contents. Reading Systems are **not required** to process or use the referenced URI resource, whether or not the resource is listed in the Manifest.

2.3.4: `img` Element

The inline element **`img`** **should** only be used to refer to images with OPS Core Media Types of GIF (http://www.w3.org/Graphics/GIF/spec-gif89a.txt), PNG (RFC 2083), JPG/JFIF (http://www.w3.org/Graphics/JPEG) or SVG (http://www.w3.org/TR/SVG11/). The **required** URI attribute, `src`, is used to reference the image resource, which **must** be listed in the Manifest.

The **required** `alt` attribute **should** contain a brief and informative textual description of the image. This text may be used by Reading Systems as an alternative to, or in addition to, displaying the image. The text is also an acceptable fallback for an **`img`** with `src` referencing a non-OPS Core Media Type for which no viable fallback was found in the manifest. The `alt` textual description is useful for Reading Systems having limited resolution displays, or for non-visual presentation. Use of the **`object`** element is the preferred mechanism for including non-core media types in an OPS Content Document.

For long descriptions, the **optional** `title` attribute **may** be used. Reading Systems **may** display the `title` attribute in addition to or in place of the `alt` text or displaying the image.

For greater accessibility, it is **strongly recommended** that OPS Content Document authors include a URI reference in the optional `longdesc` attribute referencing a resource (such as another OPS Content Document in the Publication) describing the image in finer detail. Reading System developers are also strongly urged to recognize and render in an appropriate fashion (and with accessibility in mind) the resource specified in `longdesc`. For further information on the use of this attribute and related accessibility attributes, see http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/#gl-provide-equivalents.

2.3.5: `link` Element

The **`link`** element allows for the specification of various relationships with other documents. Reading Systems **must** recognize external style sheet references specified via the `href` attribute and the associated `rel` attribute (for the values `rel=`*`"stylesheet"`* and `rel=`*`"alternate stylesheet"`*.)

2.3.6: `object` and `param` Elements

The **object** element is the preferred method for generic object inclusion. When adding objects whose data media type is not drawn from the OPS Core Media Type list or which reference an **object** implementation using the `classid` attribute, the **object** element **must** specify fallback information for the **object**, such as another **object**, an **img** element, or descriptive text. Inline fallback information is provided as OPS content appearing immediately after the final **param** element that refers to the parent **object**. Descriptive text for the **object**, using inline content, an included OPS Content Document, or some other method, **should** be provided to allow access for people who are not able to access non-textual content.

The `classid` attribute of an **object** gives the URI value of an implementation for the **object**. Conformant Reading Systems are **not required** to render objects that use external implementations, although they may do so. The MIME media type values for the `codetype` and `type` attributes **must** match those specified in the Publication's Manifest.

The associated **param** empty element is used to specify initialization values for objects. The **param** element must only appear before the renderable content of an **object**. Reading Systems **must** examine only **param** elements that are direct children of the **object**.

Example:

```
<object classid="clsid:AFAFAFA-0101-1010-0101-ABABABABABA"
        codebase="http://www.example.com/SomeScriptingLanguage/">
   <param name="code" value="TicTacToe.class">
   <param name="codebase" value="html/">
   <param name="type" value="application/x-somescriptinglanguage">
   <param name="model" value="models/TicTacToe.model">
   <param name="scriptable" value="true">
   <object type="image/png" data="tictactoe.png">
         Tic-Tac-Toe, a <em>dull</em> game.
   </object>
</object>
```

2.3.7: script and noscript Elements

Reading Systems **must not** render the textual content of the **script** element, and **should not** execute the script itself. The content must be readable without execution of **script** elements.

If **noscript** is included, whose purpose is to display some message if the Reading System chooses not to execute the script, it **must** appear after the closing tag of the **script** it is associated with. Reading Systems **must**, by default, render the content contained in **noscript** if they do not execute **script** (as is **recommended**), the default of which can be overridden by CSS **display:***none*. Note that for XHTML 1.1 conformance the content model for noscript is **Block.mix** (block level elements plus the "level-independent" elements); it cannot directly contain text or inline elements and is identical to the content model for **body** and **blockquote**, even if **noscript** itself appears inline.

The attribute `type`, which specifies the scripting language for **script**, is **required**.

One potential problem with **script**, whose content model is **#PCDATA**, is that if the code contains the characters "<" and "&", there is a potential conflict with XML. Thus, these characters, if used, **must** either be escaped, or put into a **CDATA** section. Reading System developers who include script execution capability need to be aware of this potential problem.

2.3.8: type attribute of the style element

The `type` attribute of the **style** element is **required** (per XHTML 1.1 requirements) and

must be given the value of `text/css` or the deprecated `text/x-oeb1-css`. For browser rendering of an individual OPS Content Document as an XHTML 1.1 document, instances of the deprecated `text/x-oeb1-css` should be changed to `text/css`.

2.3.9: Value of `align` attribute

The value of *`char`* for the `align` attribute is not included in the OPS XHTML subset. To achieve similar formatting, use the CSS **`text-align`** property with a *`string`* value.

2.4: DTBook Preferred Vocabulary

2.4.1: Introduction

DTBook is an XML vocabulary defined in the DAISY/NISO standard, formally, the ANSI/NISO Z39.86-2005 Standard. This vocabulary is specifically designed for eBook content. Many structures not found in XHTML are included: footnotes, sidebars, annotations, page numbers, etc. Identifying the beginning of pages and the number of the page in the original print publication enables direct NCX navigation to the beginning of each page as reflected in the print publication. These pages will vary in size from the screens of information displayed by a typical Reading System. The retention of print page marking will enable a user to move to the identical location as someone who may be using a printed book.

It is strongly **recommended** that Content Providers select this XML Preferred Vocabulary for their educational publications and for content that is highly structured.

Many publishers may already have their content in DTBook as a result of the U.S. Individuals with Disabilities Education Improvement Act (IDEIA) of 2004 or other similar international legislation. Transforming it into an OPS publication can be accomplished with ease. Having the publication available using the DTBook vocabulary may help publishers market their publications to all students in the education market place, including those with print disabilities.

Section 4 of the DAISY/NISO Standard defines the DTBook DTD. This can be found on the NISO Web site at: http://www.niso.org/standards/resources/Z39-86-2005.html.

Additional information such as DTDs, issues tracking, samples, and more can be found on the DAISY Web site at: http://www.daisy.org/z3986/.

2.4.2: DTBook Usage Requirements

DTBook, as described in Section 4 of the DAISY/NISO Standard, **must** be followed in DTBook OPS Publications. The content **must** validate against the dtbook-2005-2.dtd, which was the latest version at the time of this writing. This DTD version, all previous versions, and all future versions will be available on the DAISY website. This will ensure that OPS Publications created under any version of the DTD remain valid.

It is essential that the semantics of the DTBook elements be applied correctly. To assist with this, a set of "Structure Guidelines" has been created. Documents created conforming to DTBook **should** use the Structure Guidelines for information regarding the correct application of the semantics of the elements. The Structure Guidelines can be found at: http://www.daisy.org/z3986/structure/.

2.4.2.1: Exceptions to Section 4 of the DAISY/NISO Standard

There is a single attribute on elements in DTBook called `smilref`. This attribute is used for SMIL (http://www.w3.org/TR/2005/REC-SMIL2-20050107/) coordination. SMIL is not supported in OPS Publications. Leaving this attribute out of DTBook documents in OPS Publications will not cause validation errors.

2.5: SVG

Many images, such as maps, charts, graphs, etc., originate from vector graphics systems, not photographs. Such images can be represented in a vector (as opposed to raster) format that describes the image in terms of lines, curves and absolutely positioned blocks of text (as opposed to an array of pixels). Replacing raster images with vector ones makes documents more accessible and searchable. This additional effort on the part of Reading Systems and authors can improve accessibility significantly. Vector images also improve the visual quality of documents (since vector images are inherently scalable) and tend to decrease document sizes.

OPS Reading Systems **must** support SVG (Scalable Vector Graphics) as an OPS Core Media Type.

2.5.1: General Notes on SVG Usage

OPS supports the full SVG 1.1 Recommendation. The only exception is that since OPS is not targeting interactive content. SVG animation and scripting features are not supported and **must not** be used by publication authors; a Reading System **should not** render such content. CSS styling of SVG **must** be fully supported.

Since OPS is, by definition, supported in EPUB Reading Systems, the `requiredExtension` attribute on SVG elements with the value *`http://www.idpf.org/2007/ops`* **must** always evaluate to true. See the SVG 1.1 Recommendation section "5.8.4 The requiredExtension attribute" for more information on `requiredExtension` attribute processing.

Text in SVG images **should** be selectable and searchable. SVG images **may** contain links (**`a`** elements) and thus may be used for navigation. If a Reading System supports "tabbing" through links, SVG links **must** be included.

Inline SVG is only supported within documents using the XHTML Preferred Vocabulary. It **must not** be used within DTBook documents.

2.5.2: SVG's Use as a Standalone Image File

SVG content can be used in OPS in any place where other image types can be used. This includes:

- References from XHTML **`img`** and **`object`** elements
- References from DTBook **`img`** elements
- References from SVG **`image`** elements
- References from CSS properties that take an image URI as a parameter

To ensure maximum portability, the dimensions of SVG images **should** always be specified on the top-level SVG element . As with bitmap images, referencing elements **may** override an SVG image's dimensions causing it to scale.

Each SVG image is considered a separate document for the purposes of styling. Referencing document style sheets do not apply to referenced SVG images, nor are CSS properties inherited by SVG images. Common styles **should** be achieved by creating style sheets that are referenced by both SVG and XHTML. Note that this is consistent with situations in which an **`object`** element references an XHTML element.

2.5.3: Mixing SVG and XHTML Mark-up in the Same Document

The other way of using SVG is to embed SVG mark-up directly in XHTML mark-up. The SVG root element (**`svg`**) **must** be considered replaced by its content in the CSS layout, so **`svg`** elements **may** be used anywhere XHTML **`img`** elements could be used. Other SVG elements **may** only be used as **`svg`** element descendants. Since SVG and XHTML elements have different namespaces, care needs to be taken to assign proper namespaces to each element using namespace prefixes or the `xmlns` attribute.

In this case, there is only one document and CSS style sheets apply to the document as a whole (including cascade and inheritance). In the usual manner, each element only makes use of the properties that are applicable to it. The root **`svg`** element's role in CSS layout is determined by the **`display`**, **`float`**, and **`position`** properties, just like it works for the **`img`** element. SVG image dimensions are determined by the `width` and `height` attributes on the root **`svg`** element (which are considered to be the replaced element's intrinsic dimensions in the CSS box layout) and the **`width`** and **`height`** properties assigned by CSS. See http://www.w3.org/TR/CSS2/conform.html for a definition of "replaced element" and "intrinsic dimensions." For portability, authors **should** avoid assigning XHTML-only CSS properties to SVG elements and vise-versa.

2.6: XML Islands

This section describes the markup and a processing model for the inclusion of generic XML content in OPS Publications.

2.6.1: Introduction to XML Islands

This section defines a method of including generic XML markup in OPS Publications. The motivation is to achieve the maximum backwards, sideways, and forwards compatibility with other XML vocabularies, including Extended OEBPS, as defined in OEBPS 1.2.

This is consistent with the stated future directions of OEBPS 1.2.

To that end, this document introduces the idea of "XML Islands." Two variants exist:

- Out-Of-Line XML Islands
- Inline XML Islands

Out-Of-Line XML Islands are any complete XML document that is not authored in one of the Preferred Vocabularies (i.e. DTBook or XHTML) or uses an Extended Module in an otherwise Preferred Vocabulary document.

An Inline XML Island is an XML document fragment using a non-Preferred Vocabulary or using an Extended Module of a Preferred Vocabulary that exists within an XHTML Preferred Vocabulary document within an OPS Publication. Inline XML Islands are also namespace-qualified to the OPS namespace and are not contained within **`object`** elements.

Inline XML Islands **must not** be used within documents using the DTBook Preferred Vocabulary; they **must** only be used within XHTML documents.

2.6.1.1: Use Cases

Following are the use cases that justify the existence of XML Islands in OPS:

1. **Backwards Compatibility:** There are an unknown number of documents that use Extended OEBPS 1.2 Documents which could be used in new Reading Systems. This document describes a method in which such documents could be used with

minimal additions such as a few extra attributes in the OPS manifest, the addition of a single namespace declaration and/or the creation of a CSS style sheet.

2. **Sideways Compatibility:** There are innumerable documents using non-Preferred XML vocabularies which exist and could be used in Reading Systems. This can be accomplished using the same methodology described above or using customized display semantics according to methods not available in a baseline OPS Reading System in conjunction with fallbacks to OPS Preferred Vocabularies.
3. **Forwards Compatibility:** Experimentation with vocabularies that do not yet exist is enabled.

2.6.1.2: Display Guidelines

OPS Reading Systems **must** follow these steps in handling the display of correctly marked up XML Islands:

1. **Semantic Styling:** A Reading System **may** do more with XML Island content than provided for by the baseline requirements of OPS. This could include expanded rendering capabilities based on the Reading System's intrinsic knowledge of the XML Island vocabulary (e.g. a TEI-reader, or a MathML reader).
2. **Fallback to OPS:** A Reading System **must** use one of the fallback mechanisms if it lacks intrinsic knowledge of an XML Island's vocabulary.

This allows the inclusion of non-Preferred Vocabulary content in a standardized way that lets a Reading System to do more than OPS requires as a baseline, while still allowing baseline Reading Systems to handle such content in a reasonable manner.

2.6.2: Out-of-Line XML Islands

An Out-Of-Line XML Island is a complete XML document that is not authored in one of the Preferred Vocabularies or uses an Extended Module in an otherwise Preferred Vocabulary document. Including an Out-Of-Line XML Island requires several declarations in various parts of the Publication.

2.6.2.1: Document Requirements

An Out-Of-Line XML Island must be a valid XML document(according to its schema) and meet the other conformance requirements defined in section "Out-of-Line XML Island Content Document Requirements."

The following is a simple Out-Of-Line XML Island:

```
<?xml version="1.0"?>
<!DOCTYPE example SYSTEM "example.dtd">
<example>
   <title>This Is An Example</title>
   <paragraph>This is a paragraph with a <huge>big</huge> section
   of text that could be styled with OPS CSS!</paragraph>
</example>
```

2.6.2.2: Fallback Requirements

The inclusion of an Out-Of-Line XML Island requires certain attributes and attribute values within a publication's manifest to specify fallback information for Reading Systems that cannot natively process the Out-Of-Line XML Island's vocabulary. See the OPF specification.

2.6.2.3: Linking Requirements

Links that point into Out-Of-Line XML Islands **must** be resolved by a compliant OPS Reading System as follows:

2.6.2.3.1: Document-Level Links

Links that point at an entire document in the publication **must** automatically resolve to the fallback document if the fallback document is used.

Link Source:

```
...<a href="chapter2.xml">Chapter 2</a>...
```

Preferred Document - `chapter2.xml`:

```
        <chapter>
                <title>Chapter 2</title>
                ...
        </chapter>
```

Fallback Document - `chapter2.html`:

```
<html>
        <head>
                <title>Chapter 2</title>
        </head>
        <body>
                ...
        </body>
</html>
```

Manifest Fragment:

```
<item href="chapter2.xml" id="chapter2"
      media-type="text/xml"
      required-namespace="http://www.example.com/chapter"
      fallback="chapter2fallback"/>
<item href="chapter2.html" id="chapter2fallback"
      media-type="application/xhtml+xml"/>
```

In this example, there are two possible behaviors. If the Reading System is capable of processing the document format of `chapter2.xml` then the link resolves to `chapter2.xml`. Otherwise, the link resolves to the fallback for `chapter2.xml`, which is `chapter2.html`.

2.6.2.3.2: Fragment Links

Links that include document fragments **must** resolve to the fallback item, using the same fragment identifier. In the event that the fragment identifier does not exist in the fallback, the Reading System **must** place the link entry point at the start of the fallback document.

2.6.3: Inline XML Islands

An inline XML Island is an XML document fragment using a non-Preferred Vocabulary or using an Extended Module of a Preferred Vocabulary that exists within an XHTML Preferred

Vocabulary document within an OPS Publication.

Processing Inline XML Islands is more complex than processing Out-Of-Line XML Islands, due to processing and display requirements. Out-Of-Line XML Islands are full XML documents, whose fallback options are defined in the OPF Package Document. Inline XML Islands are handled via a markup structure designed to simulate the common programming concept of a "switch" statement.

2.6.3.1: The `switch` Element and Contained Elements

The switch element is a concept used in both SVG 1.1 and SMIL 2.x; however, OPS has requirements that are not identical to those specifications. As a result, the OPS implementation of the switch concept differs. Specifically, author intent is required in an explicit and clear fashion. In addition, the concept of a base fallback mechanism is provided. The normative definition of the **`switch`** element and its contained elements can be found in the OPS Schema.

The **`switch`** element and all contained elements reside in the OPS Namespace (*`http://www.idpf.org/2007/ops`*).

2.6.3.1.1: `switch` Element

The **`switch`** element is the containing element for an Inline XML Island. It contains zero or more **`case`** elements, and must contain one **`default`** element. The element names have been selected to be easily understood by those familiar with the "switch" feature of many programming languages.

An example of markup using the **`switch`** element follows:

```
<ops:switch id="mathmlSwitch">

   <ops:case required-namespace="http://www.w3.org/1998/Math/MathML">
      <math xmlns="http://www.w3.org/1998/Math/MathML">
         <mrow>
            <mn>2</mn>
            <mo>&InvisibleTimes;</mo>
            <mi>x</mi>
         </mrow>
         <mrow>
            <mo>+</mo>
            <mi>y</mi>
            <mo>-</mo>
            <mi>z</mi>
         </mrow>
      </math>
   </ops:case>

   <ops:default>
      <p>2x + y - z</p>
   </ops:default>

</ops:switch>
```

2.6.3.1.2: `case` Element

The **`case`** element is used to contain non-Preferred Vocabulary or Extended Module markup. There are zero or more **`case`** elements in a **`switch`** element. The **`case`** element contains no content from the OPS namespace. This does not preclude it from containing content that is outside the OPS namespace, which is typically the case.

An OPS Reading System **must not** invalidate a document and **must** display nothing if a `case` element contains no content.

2.6.3.1.2.1: `required-namespace` Attribute

The `required-namespace` attribute specifies the namespace of the XML vocabulary that the Reading System **must** support in order to process the content of the `case` element. A Reading System **should** process the content of the first `case` element that has a `required-namespace` attribute value that it supports. Note that processing of extra-OPS XML vocabularies is **not required**. Only the island's fallback (`default` element) **must** be processed.

Documents that have a root element in the XHTML namespace **must not** use elements outside the XHTML namespace and, where allowed, the SVG namespace unless they are either specifically allowed there (e.g. SVG `metadata` element) or they are descendants of a `case` element that has a matching `required-namespace` attribute value.

2.6.3.1.2.2: `required-modules` attribute

Some XML vocabularies are broken down into distinct modules, which **may** be supported in part or in whole by a compliant application. These vocabularies have a set of named modules defined in their specification, and some or all of these modules are **optional** – such **optional** modules are deemed Extended Modules in the specification.

A modularized XML vocabulary is considered supported by the Reading System if the Reading System supports the least number of modules required by the vocabulary's specification – supporting at least all non-Extended Modules of the XML vocabulary.

An Inline XML Island authored in a Preferred Vocabulary by definition incorporates use of Extended Modules. In this case, and with non-Preferred Vocabulary islands utilizing Extended Modules, the `required-modules` attribute **must** be present along with the `required-namespace` attribute. The attribute value for `required-modules` **must** be a comma-separated list containing the name(s) of the Extended Modules used in the Inline XML Island. The names of the modules are not case-sensitive, unless specifically defined otherwise in the XML vocabulary specification. Spaces in module names **must** be replaced with "-" for listing in the `required-modules` attribute value. For XHTML, in the context of OPS, the Extended Modules include *`ruby`*, *`forms`*, *`server-side-image-map`*, and *`intrinsic-events`*.

Note that listing the names of non-Extended Modules in a `required-modules` attribute value is also allowed; such modules are always considered to be supported if the XML vocabulary is supported. This can be useful both for clarity and in the case where there is a possibility that some modules may become **optional** in the later revisions of the specification (e.g. the currently deprecated Style Attribute XHTML module in OPS).

It is allowed, and sometimes useful, to provide a `required-modules` attribute on a `case` specifying a non-Preferred Vocabulary Inline XML island — either for clarity or to specify Extended Modules used in the non-Preferred Vocabulary.

An example of using `required-modules` to specify an XHTML module outside of OPS's core supported modules follows:

```
<ops:switch id="XHTMLServerSideSwitch" xmlns:ops="http://www.idpf.org/2007/ops">
        <ops:case required-namespace="http://www.w3.org/1999/xhtml"
                required-modules="server-side-image-map">
                <a href="http://www.example.com/examplemap.map">
                <img src="example.gif" ismap="ismap"
                        title="Example Map"
                        alt="A picture of an example." />
                </a>
        </ops:case>
        <ops:default>
                <img src="example.gif" usemap="map"
                        title="Example Map"
                        alt="A picture of an example." />
                <map name="map">
                        <area href="example1.html"
                                alt="Link to the first example."
                                shape="rect" coords="0,0,118,28" />
                        <area href="example2.html"
                                alt="Link to the second example"
                                shape="rect" coords="184,0,276,28" />
                </map>
        </ops:default>
</ops:switch>
```

2.6.3.1.3: `default` Element

The **`default`** element is nearly identical to the **`case`** element. The purpose of the **`default`** element is to provide compliant markup as a baseline (or fallback) for display of the **`switch`**. The **`default`** element **must** contain only XML nodes that would be allowed at the point of the document where its parent **`switch`** element was used. More precisely, a document is considered valid if and only if replacing all **`switch`** elements with the content of their child **`default`** elements yields a valid document.

2.6.3.2: Processing Inline XML Islands

Inline XML Islands **must** be processed via the markup definitions found in the switch element and its contained elements.

2.6.3.3: Displaying Inline XML Islands

2.6.3.3.1: Advanced Display Semantics

Reading Systems **must** process **`case`** elements in the order they appear, and **should** render the content of the first supported **`case`** element recognized (i.e. the Reading System supports the vocabulary specified in the `required-namespace`). A Reading System must render only a single **`case`** or **`default`** element within a **`switch`**, ignoring the contents of all other **`case`** and **`default`** elements.

Reading Systems that do support the vocabularies of any of the **`case`** elements **must** render the contents of the **`default`** element.

2.6.3.3.2: Styling of Islands

When styling is applied to a document that contains Inline XML Islands, style sheet cascading and inheritance rules are used for the complete document tree, including any Inline XML Islands. However, OPS **`switch`**, **`case`**, and **`default`** elements **must** ignore all styling properties that are assigned to them. In particular, those elements **must not** create any boxes (e.g. in terms of CSS box layout), as XHTML elements would.

2.6.3.4: Linking Considerations

Deep linking into the content of Inline XML Islands cannot be accomplished consistently because various alternative Inline XML Islands **must not** contain elements with identical `id` attributes. It is therefore not possible to provide references that will resolve regardless of the **`case`** element selected for display.

2.6.3.4.1: Linking to `switch` Elements

Reading Systems **must** recognize and display links that target **`switch`** elements even though the **`switch`** has no display characteristics of its own.

2.6.3.4.2: Broken Links

Links into Inline XML Islands **must** be dropped without warning or error as they would be in a typical XHTML-compliant browser.

2.6.3.5: NCX Requirements

Links from an NCX to Inline XML Islands **must** work identically to other links to Inline XML Islands.

2.7: Rendering of Documents on Reading Systems

A number of elements and attributes permit semantics that are **not required** of all Reading Systems. For example, some devices could be monochrome, or provide mainly audio or tactile interfaces. In such cases this specification generally **requires** Reading Systems to accept all syntax (such as attribute values) permitted for the Preferred Vocabulary, but does **not require** that they be honored. For example, a Reading System **must** parse and recognize the `border` attribute on **`table`** elements, but **may** choose to treat all values other than *0* the same as *1*.

Note that this specification does not mandate specific rendering behavior for the OPS Preferred Vocabularies. Some Reading Systems could choose to express the intent of elements in presentation by closely following web browser usage — a blank line before a paragraph, but no first-line text-indent, for example. Other Reading Systems could gear their presentation towards sustained novel-like readability: for example, no extra whitespace between paragraphs, but text-indent on the first line of each. Still other systems, such as speech generators, could present particular elements or entire documents in completely different ways.

3.0: OPS Style Sheets

Like CSS style sheets, OPS style sheets are case-insensitive, except for the parts that are not under the control of CSS. In particular, OPS Content Documents are XML documents and, as such, their element names and attribute values are case-sensitive. Therefore, corresponding element names and attribute values in OPS style sheets are case-sensitive. Currently, this applies only to element names, attribute names, and attribute values used in selectors.

Where there are differences in the syntax specified by CSS1 and CSS2, OPS style sheets follow the CSS2 syntax. A list of these differences can be found in the Comparison of tokenization in CSS 2.1 and CSS1 section of the CSS2 Specification. OPS style sheets support the CSS construct of multiple declarations separated by semi-colons. Hence, the style sheet rules:

```
h1 { color: blue }
h1 { font-weight: bold }
h1 { font-size: 12pt }
```

are equivalent to:

```
h1 { color: blue;
     font-weight: bold;
     font-size: 12pt }
```

Multiple rules with identical declaration blocks can be combined into one rule by separating the selectors with commas. Thus the rules:

```
h1 {text-indent: 0em}
h2 {text-indent: 0em}
h3 {text-indent: 0em}
```

could be combined into the equivalent form:

```
h1, h2, h3 {text-indent: 0em}
```

OPS style sheets support all CSS white space characters. Specifically, the characters "space" (Unicode code 32), "tab" (9), "line feed" (10), "carriage return" (13), and "form feed" (12) can occur as whitespace. Comments of the syntax defined in the CSS2 specification may be used in OPS-conforming CSS style sheets.

This specification supports the inline `style` attribute (deprecated), the XHTML **`style`** element, and externally linked (via **`link`** elements or the **`xml-stylesheet`** processing instruction) style sheets. In DTBook, CSS style sheets can be included via the **`xml-stylesheet`** processing instruction. Thus, CSS styling can be applied to documents of both the XHTML and DTBook Preferred Vocabularies.

This specification assumes the use of selectors (see http://www.w3.org/TR/REC-CSS2/selector.html), cascade, and inheritance (see http://www.w3.org/TR/CSS2/cascade.html) be consistent with the definitions in the CSS2 Specification. For example, the CSS2 Specification defines **`!important`** rules (see http://www.w3.org/TR/CSS2/cascade.html#important-rules) to create a balance of power between author and user style sheets

If no style sheet is defined or no applicable style is found for a given XHTML OPS element, XHTML rendering is the default as defined elsewhere in this specification and the XHTML 1.1 specification.

This specification does **not require** that Reading Systems implement text-to-speech or other read-aloud technology. Reading Systems that do not implement such technology **may** ignore any CSS properties listed in this specification under the classification "aural style sheets," as well as the **`speak-header`** property listed under "Tables."

All properties apply to elements as defined in CSS. That is, most properties can apply to all elements, while a few are limited based on the value of the **`display`** property (for example, **`text-align`** only applies when the display type is *`block`*, not *`inline`*). Reading Systems are, however, **not required** to support every distinction; for example they **may** choose to map a specific length value for **`border-width`** to one of the named values for that property.

3.1: Selectors

Selectors specify the patterns that **must** be matched in the target document to determine the elements to which the style declaration(s) in the accompanying declaration block apply. If all conditions in the pattern are true for a certain element, the selector matches the element and the declarations in the declaration block are applied. This specification assumes a use of selectors that is consistent with the CSS2 Specification. Reading Systems **must** support all CSS2 selectors, including pseudo-elements and pseudo-classes. However, certain pseudo-classes may not be applicable to all Reading Systems (e.g. `hover` for devices with a touch screen) and may not result in any rendering on such devices.

The CSS specification defines the behavior of the `lang` pseudo-class, but leaves the definition of the language identifier assigned to each element to the mark-up specification. For OPS documents, the language identifier of an element can be determined in the following fashion:

1. If an element has `xml:lang` specified, the value of that attribute is the element's language identifier.
2. Otherwise, normal XML processing of ancestors must be used to determine the language identifier.
3. Otherwise (top-level element), the element's language identifier is taken from the first `dc:language` metadata element from the publication's OPF file.

3.2: Value Types

3.2.1: URI Values

For those properties that take a URI value, the URI **must** point to a document of appropriate media type for the property in question. All such referenced documents **must** be contained within the OPF Package Document’s manifest.

3.2.2: Integers and Real Numbers

Real numbers are denoted by **number**, integer values by **integer**. Either could have an **optional** sign value (one of "+" or "-"), though particular properties could restrict the ranges and sign of numeric values.

3.2.3: Length

All non-zero coordinate and size values **must** have specified units. All units defined by CSS1 and CSS2 are supported:

px	Pixels
ex	x-height of current font
em	font-size of current font
pt	Points
in	Inches
cm	Centimeters
mm	Millimeters
pc	Picas

3.2.4: Percentages

Where percentage units are supported, they are used as defined for each property in the CSS specifications for which they are an allowed value.

3.2.5: Color

Current browsers support a host of keyword color names. XHTML 1.1 defines 16 named colors, as well as numeric values. OPS style sheets **may** use all CSS2 forms, however Reading Systems are **not required** to distinguish all these colors for rendering (otherwise monochrome devices would necessarily be non-conforming, which is not the intent).

Black	
White	
Aqua	
Blue	
Fuschia	
Gray	
Green	
Lime	
Maroon	
Navy	
Olive	
Purple	
Red	
Silver	
Teal	
Yellow	
#rrggbb	six-digit hexadecimal
#rgb	three-digit hexadecimal
rgb(r, g, b)	integers in the range 0-255
rgb(r%, g%, b%)	floats in the range of 0.0% to 100.0%

3.2.6: Time

Units defined by CSS2 are supported:

s	Seconds
ms	Milliseconds

3.2.7: Frequency

Units defined by CSS2 are supported:

Hz	Hertz
kHz	Kilohertz

3.2.8: Strings

Strings **must** be quoted using either single or double quotes (Unicode codes 39 or 34, respectively). Nested strings **must** be escaped with a backslash (e.g. " a \"nested\" string"). To embed a line break in a string, use the escape "\A". The hexadecimal "A" is the line feed character in Unicode, but represents the generic notion of "newline" in CSS.

3.3: Properties

Default values for all supported CSS properties are as listed in CSS2.

The following table lists all CSS properties and values supported by this specification. Where not all values given in the CSS2 specification are listed for a given property, those values not listed are not supported by this specification. The column “Alternate display” indicates acceptable fallback display for CSS values that a Reading System cannot display as intended.

Properties that are unique to this specification have been underlined.

CSS Structure	Alternate Display	CSS2 Section
Media Types		7
`@media`		7.2.1
`aural`		7.3
`all`		7.3
Page Model		13.2
`@page`		13.2
`:left`		13.2.4
`:right`		13.2.4

:first		13.2.4
Box Model		8
Margins		8.3
margin-top, margin-bottom, margin-left, margin-right		8.3
<length>		
<percentage>		
margin [2]		8.3
auto	*0 [1]*	
Padding		8.4
padding-top, padding-bottom, padding-left, padding-right		8.4
<length>		
<percentage>		
padding [2]		8.4
Borders		8.5
border-top-width, border-bottom-width, border-left-width, border-right-width		8.5.1
thin		
medium		
thick		
<length>	*thin/medium/thick [3]*	
border-width [2]	*thin/medium/thick [3]*	8.5.1
border-top-color, border-bottom-color, border-left-color, border-right-color		8.5.2
<color>	[4]	
transparent		
border-color [2]		8.5.2
border-top-style, border-bottom-style, border-left-style, border-right-style		8.5.3
none		

hidden		
dotted	*solid*	
dashed	*solid*	
solid		
double	*solid*	
groove	*solid*	
ridge	*solid*	
inset	*solid*	
outset	*solid*	
border-style [2]		8.5.3
border-top, border-bottom, border-left, border-right [2]		8.5.4
border [2]		8.5.4
Visual Display Model		9
display [5]		9.2.5
none		
inline		
block		
run-in		
table		
inline-table		
table-row-group		
table-header-group		
table-footer-group		
table-column-group		
table-row		
table-column		
table-cell		
table-caption		
inherit		
oeb-page-head [6]		
oeb-page-foot [6]		

`float`		9.5.1
`left`		
`right`		
`none`		
`inherit`		
`clear`		9.5.2
`none`		
`left`		
`right`		
`both`		
`inherit`		
`direction`		9.1
`ltr`		
`rtl`		
`inherit`		
`unicode-bidi`		9.1
`normal`		
`embed`		
`bidi-override`		
`inherit`		
`oeb-column-number` [13]		
`auto`		
`<integer>`	*`1`*	
Visual Formatting Model Details		10
`width`		10.2
`<length>`		
`<percentage>`		
`auto`		
`inherit`		
`min-width`		10.4

<length>		
<percentage>		
inherit		
max-width		10.4
<length>		
<percentage>		
auto		
inherit		
height		10.5
<length>		
<percentage>		
auto		
inherit		
min-height		10.7
<length>		
<percentage>		
inherit		
max-height		10.7
<length>		
<percentage>		
none		
inherit		
line-height		10.8.1
normal		
<number>		
<length>		
<percentage>		
inherit		
vertical-align		10.8.1
baseline		
sub		

super		
top		
text-top	[7]	
middle		
bottom		
ext-bottom	[8]	
inherit		
Generated Content, Automatic Numbering, and Lists		12
`content [9]`		12.2
<string>		
inherit		
`list-style-type`		12.6.2
none		
disc		
circle		
square		
decimal		
decimal-leading-zero		
lower-roman		
upper-roman		
lower-greek	Decimal	
upper-greek	Decimal	
lower-alpha		
lower-latin		
upper-alpha		
upper-latin		
hebrew	Decimal	
armenian	Decimal	
georgian	Decimal	

cjk-ideographic	Decimal	
hiragana	Decimal	
katakana	Decimal	
hiragana-iroha	Decimal	
katakana-iroha	Decimal	
inherit		
list-style-position		12.6.2
inside		
outside		
inherit		
list-style [2]		12.6.2
Paged Media		13
page-break-before		13.3.1
auto		
always		
avoid		
left	[10]	
right	[10]	
inherit		
page-break-after		13.3.1
auto		
always		
avoid		
left	[10]	
right	[10]	
inherit		
page-break-inside		13.3.1
auto		

avoid		
inherit		
orphans		13.3.3
<integer>		
inherit		
widows		13.3.3
<integer>		
inherit		
Colors and Backgrounds		14
color		14.1
<color>	[4]	
inherit		
background-color		14.2.1
<color>	[4]	
transparent		
inherit		
Fonts		15
font-family		15.2.2
<family-name>		
sans-serif		
serif		
monospace		
inherit		
font-style		15.2.3
normal		
italic	[11]	
oblique	[11]	
inherit		
font-variant		15.2.3

normal		
small-caps		
font-weight		15.2.3
normal		
bold		
100-900	[3]	
inherit		
font-size		15.2.4
xx-small		
x-small		
small		
medium		
large		
x-large		
xx-large		
smaller		
larger		
<length>	[3]	
<percentage>	[3]	
inherit		
font [2]		15.2.5
Text		16
text-indent		16.1
<length>		
<percentage>		
inherit		
text-align		16.2
left		
right		
center		

justify		
inherit		
text-decoration		16.3.1
none		
line-through		
underline		
inherit		
white-space		16.6
normal		
pre		
nowrap		
inherit		
Tables		17
caption-side		17.4.1
top		
bottom		
left		
right		
inherit		
table-layout		17.5.2
fixed		
auto		
inherit		
speak-header		17.7.1
once		
always		
inherit		
Aural Style Sheets		19
volume		19.2
silent		

x-soft		
soft		
medium		
loud		
x-loud		
<percentage>	[3]	
0—100	[3]	
inherit		
speak		19.3
normal		
none		
spell-out		
inherit		
pause-before		19.4
<time>		
<percentage>		
inherit		
pause-after		19.4
<time>		
<percentage>		
inherit		
pause [2]		19.4
cue-before		19.5
<uri>		
none		
inherit		
cue-after		19.5
<uri>		
none		
inherit		
cue [2]		19.5

speech-rate		19.8
x-slow		
slow		
medium		
fast		
x-fast		
faster		
slower		
<number> [12]		
inherit		
voice-family		19.8
male		
female		
child		
inherit		
pitch		19.8
x-low		
low		
medium		
high		
x-high		
<frequency>		
inherit		
stress		19.8
0-100		
inherit		
richness		19.8
0-100		
inherit		
speak-punctuation		19.9
code		

none		
inherit		
speak-numeral		19.9
digits		
continuous		
inherit		

[1] Reading Systems **may** set the value of any **margin** property whose specified value is *auto* to *0*.

[2] This is a shorthand property. The syntax for its value is given in the CSS2 specification. Where this specification limits values or indicates alternate representations for properties abbreviated by this property, the same limits and alternate representations apply to this property.

[3] Reading Systems **may** map to one of the keyword values listed for this property.

[4] See Section 3.2.5 on color units.

[5] CSS2 provides a full description of the various table values and their correct renderings. Please refer to the CSS2 Tables specification (http://www.w3.org/TR/REC-CSS2/tables.html) for a more detailed discussion of the various table values.

CSS2 and XHTML provide similar but subtly different algorithms for rendering table data. These algorithms tend to generate the same results, but there are a few exceptions. In such cases, conforming Reading Systems **must** produce output consistent with the algorithm specified by CSS2.

When using tables, authors **should** follow the Techniques for Web Accessibility Guidelines (http://www.w3.org/TR/WAI-WEBCONTENT-TECHS/) for maintaining as much semantic information as possible. That document describes good practices for choosing how and when to use table tags, and when to use CSS properties. Specifically, see "Guideline 5: Create tables that transform gracefully" (http://www.w3.org/TR/WAI-WEBCONTENT-TECHS/#gl-table-markup).

[6] The content of an element assigned **display:** *oeb-page-head* **should** be presented only as a header, and the content of an element assigned **display:** *oeb-page-foot* **should** be presented only as a footer. Neither **should** be simply presented as if it were inline or block. Reading Systems, however, are free to present headers and footers either in special areas as usual for paper publications, or to make them available in another way. For example, a device with a small screen might instead pop them up on demand. For purposes of page layout, these display values are similar to block boxes with an absolute position (i.e. a **position** value of *fixed* or *absolute*). That is, they are removed from the normal flow and a new block box is created with its own flow. Margins, padding, and other block characteristics are determined as if the element had **position:** *fixed* set.

An element assigned **display:** *oeb-page-head* or **display:** *oeb-page-foot* **shall not** be considered in effect while any preceding content remains presented. For example, when rendered to a screen with appropriate style settings, the *myhead*-classed **div** element below would become the page header as soon as nothing preceding the containing **div** is displayed:

```
<div>
   <div class="myhead" style="display: oeb-page-head">
      The OEB Publication Structure: Introduction
   </div>
```

```
    <h2>Introduction</h2>
    <p>...</p>
</div>
```

Such a header (or footer) remains in effect until another header (or footer) is in effect instead, or until no part of its parent element remains presented (such as when the `div` is no longer visible in the above example), whichever occurs first.

[7] Reading System **may** map to top.

[8] Reading System **may** map to bottom.

[9] **Must not** be used within a style sheet whose `@media` value is other than *aural*.

[10] One-page Reading Systems **must** treat *left* and *right* as *always*.

[11] Reading Systems **need not** distinguish *italic* and *oblique* from each other.

[12] Number specifies the speaking rate in words per minute.

[13] Specifies a number of columns in which to render content; **may** be applied to all block level elements. Reading Systems are free to support integer values other than *1*, or **may** map them to *1*. Reading Systems may support column balancing. A value of *auto* allows the Reading System to decide on the optimal number of columns in which to render content, considering available width, font sizes, or any other metrics it considers relevant for readability.

3.4: Embedded Fonts

To provide authors with control over the appearance of the text, OPS supports the CSS2 font-face at-rule (@font-face). See section 15.3.1 of the CSS2 Recommendation. The following font descriptors **must** be supported:

- `font-family`
- `font-style`
- `font-variant`
- `font-weight`
- `font-size`
- `src`

For portability, authors **must not** use any other descriptors. Font files **must** carry all information needed for rendering Unicode characters. Fonts **must not** provide mappings for Unicode characters that would change the semantics of the text (e.g. mapping the letter "A" to a biohazard symbol). Content creators **must not** assume that any particular font format is supported. Fonts could be included in multiple formats by using a list of files for the `src` descriptor; the first supported format **should** be used. At least one file in OpenType format **should** always be included in the list. It is advisable for a Reading System to support the OpenType font format, but this is not a conformance requirement; a reading system **may** support no embedded font formats at all. Content creators **should** use comma-separated lists for font-family properties to specify fallback font choices.

Content creators **must** always honor usage restrictions that are encoded in OpenType fonts (and many other font formats). Fonts that are marked "no embedding" **must not** be included in OPS Publications.

Any font files included in an OPS Publication **must** be included in the OPF `manifest` with appropriate media type (*application/vnd.ms-opentype* for OpenType fonts).

Appendices

Appendix A: The NVDL Definition of OPS XHTML Content Documents

This NVDL specification references three XML schemas:

1. The XHTML DTD
2. The SVG RelaxNG schema
3. The OPS RelaxNG schema

The first two of these are referenced externally; the OPS schema is defined in Appendix B.

```
<?xml version="1.0" encoding="UTF-8"?>
<rules xmlns="http://purl.oclc.org/dsdl/nvdl/ns/structure/1.0"
       startMode="valid-ops" xmlns:html="http://www.w3.org/1999/xhtml">

<!-- mode that allows any elements -->
<mode name="anything">
   <anyNamespace>
      <allow/>
   </anyNamespace>
</mode>

<!-- mode that only allows either valid SVG fragment -->
<!-- or arbitrary XHTML fragment -->
<mode name="blessed-xhtml">
   <namespace ns="http://www.w3.org/2000/svg">
      <validate schema="http://www.w3.org/Graphics/SVG/1.1/rng/svg11.rng"
                useMode="svg-content"/>
   </namespace>
   <namespace ns="http://www.w3.org/1999/xhtml">
      <allow/>
   </namespace>
</mode>

<!-- mode that only allows arbitrary SVG fragment -->
<mode name="blessed-svg">
   <namespace ns="http://www.w3.org/2000/svg">
      <allow/>
   </namespace>
</mode>

<!-- mode that allows any elements -->
<!-- any XHTML fragments are attached in place of -->
<!-- the OPS fragment that invoked this mode -->
<mode name="xhtml-attach">
   <namespace ns="http://www.w3.org/1999/xhtml">
      <attach/>
   </namespace>
   <anyNamespace>
      <allow/>
   </anyNamespace>
</mode>

<!-- mode that allows any elements -->
<!-- any SVG fragments are attached in place of the OPS fragment -->
<!-- that invoked this mode -->
```

```
<mode name="svg-attach">
   <namespace ns="http://www.w3.org/2000/svg">
      <attach/>
   </namespace>
   <anyNamespace>
      <allow/>
   </anyNamespace>
</mode>

<!-- mode that allows any valid XHTML with SVG islands included -->
<!-- and OPS switch elements with valid default clause -->
<mode name="valid-ops">
   <namespace ns="http://www.w3.org/1999/xhtml">
      <validate schema="http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"
                schemaType="application/xml-dtd" useMode="xhtml-content"/>
   </namespace>
</mode>

<!-- mode that validates SVG and OPS islands inside XHTML -->
<mode name="xhtml-content">
   <namespace ns="http://www.w3.org/2000/svg">
      <validate schema="http://www.w3.org/Graphics/SVG/1.1/rng/svg11.rng"
                useMode="svg-content"/>
   </namespace>
   <namespace ns="http://www.w3.org/1999/xhtml">
      <attach/>
   </namespace>
   <namespace ns="http://www.idpf.org/2007/ops">
      <!-- doing two independent thing here -->
      <!-- first, attach XHTML islands from default element into the XHTML -->
      <!-- tree so that they can be validated in the right context -->
      <unwrap useMode="xhtml-attach">
         <context path="case" useMode="anything"/>
      </unwrap>
      <!-- next, validate OPS itself (switch element), paying attention to -->
      <!-- what can go inside case and default elements -->
      <validate schema="ops20.rng">
         <context path="case" useMode="anything"/>
         <context path="default" useMode="blessed-xhtml"/>
      </validate>
   </namespace>
</mode>

<!-- mode that validates OPS islands inside SVG -->
<mode name="svg-content">
   <namespace ns="http://www.w3.org/2000/svg">
      <attach/>
   </namespace>
   <namespace ns="http://www.idpf.org/2007/ops">
      <!-- doing two independent thing here -->
      <!-- first, attach SVG islands from default element into SVG tree, -->
      <!-- so that they can be validated in the right context -->
      <unwrap useMode="svg-attach">
         <context path="case" useMode="anything"/>
      </unwrap>
      <!-- next, validate OPS itself (switch element), paying attention -->
      <!-- to what can go inside case and default elements -->
      <validate schema="ops20.rng">
         <context path="case" useMode="anything"/>
         <context path="default" useMode="blessed-svg"/>
      </validate>
   </namespace>
```

```
</mode>
</rules>
```

Appendix B: The OPS Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar ns="http://www.idpf.org/2007/ops" xml:lang="en"
         xmlns="http://relaxng.org/ns/structure/1.0"
         datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">

<start>
   <ref name="OPS.switch"/>
</start>

<define name="OPS.switch">
   <element name="switch">
      <optional>
         <attribute name="id">
            <data type="ID"/>
         </attribute>
      </optional>
      <oneOrMore>
         <element name="case">
            <optional>
               <attribute name="id">
                  <data type="ID"/>
               </attribute>
            </optional>
            <attribute name="required-namespace">
               <text/>
            </attribute>
            <optional>
               <attribute name="required-modules">
                  <text/>
               </attribute>
            </optional>
            <zeroOrMore>
               <ref name="OPS.switch"/>
            </zeroOrMore>
         </element>
      </oneOrMore>
      <element name="default">
         <optional>
            <attribute name="id">
               <data type="ID"/>
            </attribute>
         </optional>
<zeroOrMore>
            <ref name="OPS.switch"/>
         </zeroOrMore>
      </element>
   </element>
</define>

</grammar>
```

Appendix C: Contributors

This specification has been developed through a cooperative effort, bringing together publishers, Reading System vendors, software developers, and experts in the relevant standards.

Version 2.0.1 of this specification was prepared by the International Digital Publishing Forum's EPUB Maintenance Working Group. Active members of the working group at the time of publication of revision 2.0.1 were:

Alexis Wiles (OverDrive)
Alicia Wise (Publishers Licensing Society)
Amir Moghaddami (National Library and Archives of Iran)
Andreas Gosling (Penguin UK)
Andy Williams (Cambridge University Press)
Anupam Jain (Innodata Isogen)
Ben Trafford (Invited Expert)
Benoit Larroque (Feedbooks)
Bill McCoy (Invited Expert)
Bill Kasdorf (Apex CoVantage)
Bob Kasher (The BookMasters Group)
Brady Duga (eBook Technologies)
Byron Laws (PreMedia Global)
Catherine Zekri (University of Montreal)
Chris Kennedy (Pearson Education)
Corey Podolsky (Entourage Systems Inc.)
Cristina Mussinelli (AIE)
Daihei Shiohama (Voyager Japan)
Dan Amos (DNAML)
Dan Galperin (Kobo)
Dan Kok (Crossway Books and Bibles)
Dave Cramer (Hachette Book Group USA)
Dave Gunn (RNIB Centre for Accessible Information)
David Mandelbaum (Barnes&Noble.com)
Deidra Roberts (World Health Organization)
Donald Goyette (McGraw-Hill Professional)
Eric Freese (Aptara)
Eric Gold (Digital Divide Data)
Eric Muller (Adobe)
Garth Conboy (eBook Technologies) - Working Group Vice-chair
George Kerscher (DAISY Consortium) - Working Group Chair
Gregory Shepherd (Cengage Learning)
Guy Fain (Crossway Books & Bibles)
Hadrien Gardeur (Feedbooks)
Hisashi Hoda (Voyager Japan)
Ignacio Fernández Galván

Israel Viente (Mendele He-Books)
Jim Link (Macmillan Publishing Solutions)
James MacFarlane (Easypress Technologies)
Jim Rura (Educational Testing Service)
John Crossman (Benetech)
John Prabhu (HOV Services)
John Rivlin (eBook Technologies)
John Wait (Pearson Education)
Jon Noring (Invited Expert)
Joshua Tallent (eBook Architects)
Karen Broome (Sony)
Keith Fahlgren (Threepress Consulting)
Kenny Johar (Vision Australia)
Laurie Casey (Pearson)
Lech Rzedzicki (Pearson UK)
Liisa McCloy-Kelley (Random House)
Lindy Humphreys (Wiley/ Blackwell Books)
Liza Daly (Theepress Consulting)
Makoto Murata (JEPA EPUB Study Group)
Marco Croella (Simplicissimus Book Farm)
Markus Gylling (DAISY Consortium)
Mattias Karlsson (Dolphin Computer Access AB)
Michael Smith (IDPF), Neil Soiffer (Design Science)
Noah Genner (BookNet Canada)
Pat Pagano (HarperCollins)
Patricia Payton (Bowker)
Patrick Barry (The Educational Company of Ireland)
Patrick Berube (LEARN)
Paul Durrant (Durrant Software Limited)
Paul Norton (Invited Expert)
Penelope Reid (EPUB User Group (UK))
Perce Huang (Far EasTone Telecommunications)
Peter Brantley (Internet Archive)
Peter Sorotokin (Adobe)
Richard Heiberger (HarperCollins Publishers)
Richard Kwan (Invited Expert)
Russell White (Random House)
Samir Kakar (Aptara)
Satya Pamarty (codeMantra)
Scott Cook (codeMantra)
Sean Ramsey (LibreDigital)

Siobahn Padgett (Hachette BG USA)
Steve Arany (John Wiley & Sons)
Takeshi Kanai (Sony)
Thad Swiderski (LibreDigital)
Tim Middleton (BookNet Canada)
Trudy Conti (Follett)
Tyler Ruse (LibreDigital)
William Howard (EasyPress Technologies)

Version 2.0 of this specification was prepared by the International Digital Publishing Forum (IDPF) Open eBook Publication Structure (OEBPS) Working Group. Members of the working group at the time of publication of revision 2.0 were:

Angel Ancin, iRex Technologies
Rick Bowes, Thomson Learning
Peter Brantley, Digital Library Federation
Clint Brauer, SONY
Ken Brooks, Thomson Learning
Janice Carter, Benetech
Richard Cohn, Adobe Systems, Inc.
Garth Conboy, eBook Technologies, Inc. (Working Group co-Chair)
Neil De Young, Hachette Book Group
Linh Do, Random House
Brady Duga, eBook Technologies, Inc.
Geoff Freed, WGBH
Markus Gylling, DAISY Consortium
Jonathan Hevenstone, Publishing Dimensions (Working Group Vice Chair)
Rick Johnson, VitalSource Technologies
George Kerscher, DAISY Consortium
Steve Kotrch, Simon & Schuster
Bill McCoy, Adobe Systems, Inc.
Jon Noring, Digital Pulp Publishing
Siobhan Padgett, Hachette Book Group
Mina Park, Holtzbrinck Publishers
John Rivlin, eBook Technologies, Inc. (Working Group co-Chair)
Tyler Ruse, CodeMantra
Mike Smith, Harlequin
Peter Sorotokin, Adobe Systems, Inc.
Jennifer Sutton, Benetech
Ben Trafford, Invited Expert
Gary Varnell, Osoft
Eli Willner, Green Point Technology Services
Ric Wright, Adobe Systems, Inc.

The prior version of the OPS specification, on which this effort is based, was OEBPS 1.2. OEBPS 1.2 was developed by the Open eBook Forum Publication Structure Working Group. Active members that that working group during development of OEBPS 1.2 were:

Jeff Alexander, Franklin Electronic Publishers

Garth Conboy, Gemstar eBook Group Limited (Version 1.2 Co-Editor)

Eva Conkright, Random House

Brady Duga, Gemstar eBook Group Limited (Version 1.2 Co-Editor)

Jerry Dunietz, Microsoft Corporation (Working Group Vice Chair)

David Goldstein, Versaware Inc.

Gene Golovchinsky, FX Palo Alto Laboratory, Inc. (Development Sub-Group Co-Chair)

Donald Goyette, McGraw-Hill

Rebecca Guenther, Library of Congress

Markku Hakkinen, isSound Corporation

Barbara Hammond, Nokia

Benjamin Jung, deepX

George Kerscher, DAISY Consortium

Christopher R, Maden, crism consulting

Bob Mathews, Adobe

Michael Moodie, National Library Service for the Blind and Physically Handicapped, Library of Congress

Jon Noring, Windspun Inc. (Maintenance Sub-Group Chair)

David Pawson, Royal National Institute for the Blind

Chris Powell, University of Michigan Library

James Pritchett, Recording for the Blind and Dyslexic

Allen Renear, Brown University Scholarly Technology Group (Working Group Chair)

Dorothea Salo, OverDrive, Inc. (Working Group Scribe)

Ilan Sender, Microsoft Corporation

Sheela Sethuraman, CAST

Roger Sperberg, e-bks, LLC

Patrick Stickler, Nokia

Lisa Rupe, Questia Media Inc.

Mikhail Vaysbukh, Data Conversion Laboratory

Garret Wilson, GlobalMentor, Inc. (Development Sub-Group Co-Chair)

Harrison Yee, Apex

Appendix D: Acknowledgements

The working group wishes to specifically acknowledge the contributions of the following individuals. **Peter Sorotokin** for authoring the OPS and OPF RelaxNG schemas, creation of the NVDL definition of OPS, and general technical acumen. **Ben Trafford** for the concept and drafting of XML Islands, as well as overall technical participation, and the XML templates used to produce the specifications. **George Kerscher** for drafting the OPF NCX section, providing consistent accessibility direction, and broad technical input. **Brady Duga** and **Jon Noring** for directional

contributions, specification editing, and providing continuity with the historic OEBPS 1.2 effort.
Garth Conboy for working group leadership and motivation, specification drafting, and technical contributions.

Appendix E: Supporting Information & Errata

For additional information about all IDPF specifications including sample files, specification implementations and other information, please visit our public forums at http://www.idpf.org/forums. If errors in the specifications are identified following publication, please post these errors to the forums. The responsible Working Group will review the errors and post pending corrections to the specifications if required or necessary. Corrections will be incorporated into subsequent versions of the specifications.